machine learning: A set of methods that can
- automatically learn regularities in observed data
- use them to predict & make decisions about/in unobserved data

Main Classes of Learning Problems
- Supervised (predictive) learning: given examples w/ labels, predict labels of unseen examples
  - classification: label = category
  - regression: label = quantity
  - ranking: label = order (relative position)
- Unsupervised (descriptive) learning: given unlabeled examples, discover hidden labels (structure) of all/each/unseen examples
  - (probability) density estimation: hidden variable = probability density
  - clustering: hidden variable = cluster index; examples within clusters are similar and differ from examples outside
  - Dimensionality Reduction: hidden variable = coordinates in low-dimensional space that preserve some properties of original space, e.g. angles

Other Learning Scenarios
- semi = supervised learning: given both labeled and unlabeled examples, predict label for all unseen examples
- Transductive inference: similar to semi-supervised, only used to predict labels for a fixed seen unlabeled examples, not unseen examples
- Active learning: sequentially choose unlabeled examples for labeling
  - for accurately & quickly predict labels for all unseen examples
- Online learning: (keep learning) multiple rounds of learning & decision-making; receive one unlabeled example at a time, predict its label, receive true label, incur loss
  - GOAL: minimize cumulative loss
- Reinforcement learning: multi-round learning like online, but...
  - no long-term reward
  - decisions can change environment
  - e.g. search, ads on internet & income as much/as possible

DEFINITIONS
- examples: data used for training/evaluation
- features: set of attributes represented as a vector $x \in \mathcal{X}$ (in some feature/input space, e.g. $\mathbb{R}^d$) associated w/ example
  - categorical/nominal: no intrinsic value or ordering, just a label
  - Quantitative: has intrinsic numerical value, e.g. counts, time, discrete or continuous
  - Ordinal: has intrinsic ordering, but no numerical value
- labels: categories, values, or ordering in output/label space assigned to examples
  - classification: label $y \in \mathcal{Y} = \{1, \dots m\}$, m = # of classes
  - Regression: label $y \in \mathcal{Y} = \mathbb{R}$, a real value
- Training examples: $\mathcal{D} \in \{z_1, \dots, z_n\}$, $z_i = (x_i, y_i)$
  - examples of (features & labels) used to train a learning algorithm
- Validation examples: examples used to tune parameters of learning algorithms, e.g. λ, polynomial degree (not coeffs) in reg.
- Test examples: examples (features & labels) used to evaluate performance of learning algorithm (once fixed, cheap test if train data separable)
- Hypothesis set: $\mathcal{H} = \{h_1, h_2, \dots\}$
  - set of functions/decision rules mapping feature vectors to labels
  - learning algorithm: training examples $\to h \in \mathcal{H}$
- Loss Function: $\ell(x, y, h) : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathbb{R}_+ \in [0, \infty)$
  - function that quantifies our degree of disagreement between our true and predicted label
  - 0-1 loss certain/exact: $\ell(x, y, h) = \mathbb{1}(h(x) \neq y)$
  - squared loss function regression: $\ell(x, y, h) = (h(x) - y)^2$
  - Lost Function really just a function of $y, h(x)$

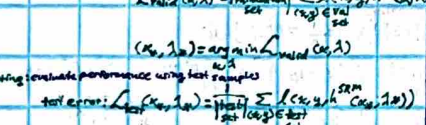
MAIN STAGES OF ANY LEARNING PROBLEM
- data partitioning / training, validation, test sets
  - # of validation examples $\gg$ # of parameters to be trained
  - # training examples $\gg$ # testing examples when total # labeled examples is small
- manual explore all possibilities: brute force computation
- preprocessing & feature selection: leverage prior domain knowledge & beliefs, often messy, heuristic, critical step: can dramatically affect performance
  - preprocessing: missing data, imbalanced data, outliers, profiling, normalizing, data augmentation
  - feature selection: identify most relevant subsets/functions of input data
- training a class of learning rules: $\mathcal{H} = \{z_1, \dots, z_n\}$, $z_i = (x_i, y_i)$, fix tuning parameters λ & find/select the best decision rule
- Overfitting: training error small, testing error significantly larger; caused when # free parameters is $\gg$ # examples, solutions based on empirical risk minimization (ERM) won't lead to nicely behaved (simple) models; starts fine-tuning to training data, instead of learning to generalize
- Structural Risk Minimization (SRM) principle:

$$\mathcal{L}_{train}(h, \mathcal{D}, \lambda) = \left[\frac{1}{n}\sum_{j=1}^{n} \ell(x_j, y_j, h)\right] + \frac{\lambda}{n}\,\text{complexity}(h)$$

(empirical risk / data term; structure penalty / constraints)

$$h^{SRM}(\mathcal{D}, \lambda) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \mathcal{L}_{train}(h, \mathcal{D}, \lambda)$$

- complexity regularization: 2nd term of $\mathcal{L}_{train}$, mitigates overfitting by steering solutions to be faithful to prior beliefs
- Bias-Variance tradeoff:
  - $\lambda = 0$: prior beliefs completely ignored; complex solutions have low bias, high variance
  - $\lambda \gg 1$: prior beliefs dominate. Solutions have low variance, high bias



- validation: learn the best values of $\mathcal{H}, \lambda$ using validation samples & tries to minimize the validation error:

$$\mathcal{L}_{valid}(\mathcal{D}, \lambda) = \frac{1}{|\text{validation set}|}\sum_{(x,y) \in \text{validation set}} \ell(x, y, h^{SRM}(\mathcal{D}, \lambda))$$

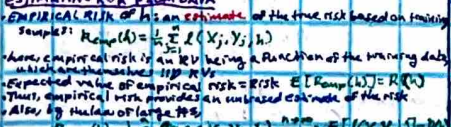
$$(\mathcal{H}_*, \lambda_*) = \underset{\mathcal{H}, \lambda}{\operatorname{argmin}} \mathcal{L}_{valid}(\mathcal{D}, \lambda)$$

- testing: evaluate performance using test samples

$$\text{test error: } \mathcal{L}_{test}(h^{SRM}(\mathcal{D}, \lambda_*)) = \frac{1}{|\text{test set}|}\sum_{(x,y) \in \text{test set}} \ell(x, y, h^{SRM}(\mathcal{D}, \lambda_*))$$

- other performance metrics: correct classification rate, confusion matrix, prob. of false alarm, missed detection, precision, recall, F-score, MSE, MAE
- small training error $\not\Rightarrow$ small validation error

CROSS-VALIDATION
not enough labeled data for creating separate validation samples
cycle samples through training & validation (can also do for testing if needed)
K-fold cross validation



$$\mathcal{L}_{cv}(\mathcal{D}, \lambda) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{L}_{valid\,fold_k}(\mathcal{D}, \lambda)$$

$$(\mathcal{H}_*, \lambda_*) = \underset{\mathcal{H}, \lambda}{\operatorname{argmin}} \mathcal{L}_{cv}(\mathcal{D}, \lambda)$$

- once you have $\mathcal{H}_*, \lambda_*$: train on all n samples to obtain $h^{SRM}(\mathcal{D}, \lambda_*)$
- evaluate performance on test samples!

- Holdout CV: $k = 2$, LOOCV: $k = n$, typical Fold & choice: $K = 5, 10$

ESTIMATING RISK OF h FROM DATA
- EMPIRICAL RISK of h: an estimate of the true risk based on training samples: $R_{emp}(h) = \frac{1}{n}\sum_{j=1}^{n} \ell(X_j, Y_j, h)$
- have, empirical risk is an RV being a function of the training data, whose randomness comes from RVs
- expected value of empirical risk = risk $\mathbb{E}[R_{emp}(h)] = R(h)$
- Thus, empirical risk provides an unbiased estimate of the risk
- Also, by the law of large #s:

$$R_{emp}(h) = \frac{1}{n}\sum_{j=1}^{n} \ell(X_j, Y_j, h) \xrightarrow{n \to \infty} \mathbb{E}[\ell(X, Y, h)] = R(h)$$

- conditioned on a specific set of training samples: $\{(x_1, y_1), \dots, (x_n, y_n)\} = \mathcal{D} = \{(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\}$

$$R_{emp}(h|\mathcal{D}) = \frac{1}{n}\sum_{j=1}^{n} \ell(x_j, y_j, h) = \mathcal{L}_{train}(h, \mathcal{D}, \lambda = 0)$$

the empirical risk is a deterministic number!
- USEFUL: If $x_{emp}, y_{emp} \sim$ Uniform(CD) then

$$R_{emp}(h|\mathcal{D}) = \frac{1}{n}\sum_{j=1}^{n} \ell(x_j, y_j, h) = \mathbb{E}[\ell(X_{emp}, Y_{emp}, h)]$$

- such

$$\left(\begin{matrix}\text{discrete} \\ \text{variable}\end{matrix}\right) \sim \left(\begin{matrix}x_1 \\ x_2 \\ \vdots \\ x_k\end{matrix}\right) \quad \begin{matrix}x_i \text{ tom, categorical} \\ x_i \text{ kinds of category} \\ x_i \text{ real val. math} \end{matrix}$$ Gauss

PROBABILISTIC FRAMEWORK
- View training data samples as an unordered collection of IID samples from unknown joint CDF/PMF/PDF

$$(X_1, Y_1), \dots (X_n, Y_n) \sim IID\, p(x,y)$$

- Why? Mathematical convenience & the Fundamental Theorem: unordered observations are conditionally independent given an underlying latent variable
- $(\vec{x}, y) \to RV(\vec{x}, y) \to$ specific realization
- Find h to minimize risk

STATISTICAL DECISION THEORY
- optimum decision making GIVEN
  1. Model: $p(\vec{x}, y)$ ← learning is when you don't have this!
  2. Loss function $\ell(y, \hat{y})$
- RISK (generalization error, expected loss) of decision rule $h(\cdot) =$ Expected loss $= R(h) = \mathbb{E}[\ell(y, h(\vec{x}))] = \int\int \ell(y, h(\vec{x}))\, p(\vec{x}, y)\, d\vec{x}\, dy$

Need $p(\vec{x}, y)$, $\ell(y, \hat{y})$, $h(\cdot)$ to compute this ← OPTIMUM RULE

BAYES DECISION RULE
- rule which minimizes risk: $h_{Bayes}(\cdot) = \underset{h}{\operatorname{argmin}} R(h)$
- $\mathcal{H}$ = the optimum decision rule/hypothesis given $p(\vec{x}, y)$ and $\ell(y, \hat{y})$
- Bayes risk: risk of Bayes decision rule = minimum achievable risk

$$R_{Bayes} = R(h_{Bayes}) = \min_h R(h)$$

for particular

POSTERIOR RISK
- Posterior Risk of decision rule = conditional expected loss given $X = x$

$$\mathbb{E}[\ell(y, h(x)) | X = x] = \int \ell(y, h(x))\, p(y|x)\, dy$$

Need POSTERIOR distribution $p(y|x)$, loss function $\ell(y, \hat{y})$ and decision rule $h(\cdot)$ to compute this
- $h_{Bayes}(x) = \underset{h}{\operatorname{argmin}} \mathbb{E}[\ell(y, \hat{y}) | X = x]$
- optimum label decision $h_{Bayes}(\vec{x})$ for feature $\vec{x}$ equals the value of $\hat{y}$ that minimizes the posterior risk
- this means we only need posterior $p(y|x)$ and loss function to compute Bayes decision rule

EXAMPLES OF BAYES DECISION RULES
- Minimum mean square error (MMSE) Estimate
  - $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}^m$, $\ell(x, y, h) = \|y - h(\vec{x})\|^2$
  - Result: $h_{Bayes}(\vec{x}) = h_{mmse}(\vec{x}) =$ posterior mean $= \mathbb{E}[Y|\vec{X} = x]$
- Minimum mean absolute error (MMAE) estimate
  - $\mathcal{X} = \mathcal{Y} = \mathbb{R}$, $\ell(x, y, h) = |y - h(\vec{x})|$
  - Result: $h_{Bayes}(\vec{x}) = h_{mmae}(\vec{x}) =$ posterior median, i.e. $\int_{-\infty}^{h_{mmae}(\vec{x})} p(y|\vec{x})\, dy = \int_{h_{mmae}(\vec{x})}^{\infty} p(y|\vec{x})\, dy$
- Minimum Probability of Error (MAPE) Estimate
  - $\mathcal{Y} = \{1, \dots m\}$, $\ell(y, h(\vec{x})) = \mathbb{1}(y \neq h(\vec{x}))$, unit cost for error

$$h_{Bayes}(\vec{x}) = h_{mpe}(\vec{x}) = \text{posterior mode} = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} p(y|\vec{x})$$

- Maximum Aposteriori Probability (MAP) Decision Rule
- $h_{map}(\vec{x}) =$ posterior mode $= \underset{y \in \mathcal{Y}}{\operatorname{argmax}} p(y|\vec{x})$

$$= \underset{y \in \mathcal{Y}}{\operatorname{argmax}} p(\vec{x}, y) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} p(y)\, p(\vec{x}|y)$$

$$= \underset{y}{\operatorname{argmax}}\, p(y) \cdot p(\vec{x}) = \underset{y}{\operatorname{argmax}}\, p(\vec{x}|y) \cdot p(y) = \underset{y}{\operatorname{argmax}} [\ln p(\vec{x}|y) + \ln p(y)]$$

$$= \underset{y \in \mathcal{Y}}{\operatorname{argmin}} [-\ln p(\vec{x}|y) - \ln p(y)]$$

- Maximum Likelihood (ML) Decision Rule
- $h_{ml}(\vec{x}) = \underset{y}{\operatorname{argmax}}\, p(\vec{x}|y)$
- RESULT: if $\mathcal{Y} \sim$ uniform then MAP rule reduces to the ML rule, i.e. $h_{map}(\vec{x}) = h_{ml}(\vec{x}) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} p(\vec{x}|y)$

PROBABILISTIC MODELS
- Family of probability models $\{p(\vec{x}, y)\}$
- Parametric vs. Nonparametric
- Parametric: Fixed number of parameters, e.g. Gaussian w/ mean vector & covariance matrix
- Non parametric: number of parameters not fixed in advance, but grows w/ the amount of training data, e.g. series expansions of pdfs, Fourier/wavelet, where # terms depends on training data
- Generative vs. Discriminative
- Generative: training $\to \hat{p}(\vec{x}, y)$ full model estimate; can use this to generate (sample) $(\vec{x}, y)$
- Discriminative: training data $\mathcal{D} \to \hat{p}(y|\vec{x})$ posterior model estimate. Cannot generate $(\vec{x}, y)$ since $p(\vec{x})$ is not learned. Recall that the posterior model is sufficient for optimum decision making.

STATISTICAL LEARNING
- Model-Based Methods
- select family of models, $p(\vec{x}, y)$, $p(y|\vec{x})$
  - parametric/nonparametric
  - generative/discriminative
- estimate model from data
  - Frequentist learning: via ML principle & IID data assumption
  - Bayesian learning: via MAP principle & IID data assumption → need to specify a prior $\Pi(\cdot)$
- Compute (bayes) decision rule for learned model → need to specify LOSS FUNCTION
- Provides a decision rule & confidence of decision under model
- FREQUENTIST & BAYESIAN GENERATIVE LEARNING
- FREQUENTIST: parameters θ viewed as deterministic unknowns and estimated from training data via ML

$$\hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmax}}\, p(\mathcal{D}|\theta) = \underset{\theta}{\operatorname{argmax}} \left[\prod_{j=1}^{n} p(\vec{x}_j, y_j|\theta)\right] = \underset{\theta}{\operatorname{argmax}} \left[\frac{1}{n}\sum_{j=1}^{n} -\ln p(\vec{x}_j, y_j|\theta)\right]$$

- analogous to ERM w/ a loss function defined by generative prob. model
- BAYESIAN: parameters θ viewed as random with given prior $\Pi(\theta)$ and estimated from $\mathcal{D}$ via MAP

$$\hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmax}}\, p(\theta|\mathcal{D}) = \underset{\theta}{\operatorname{argmax}} [p(\mathcal{D}|\theta)\pi(\theta)] = \underset{\theta}{\operatorname{argmin}} \left[\frac{1}{n}\sum -\ln p(\vec{x}_j, y_j|\theta) - \frac{1}{n}\ln \pi(\theta)\right]$$

- analogous to SRM with a loss function defined by a generative probability model and a structure penalty (complexity regularization) term determined by the prior
- BAYESIAN GENERATIVE LEARNING AS STRUCTURAL RISK MINIMIZATION
- BAYESIAN GENERATIVE LEARNING:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[\frac{1}{n}\sum_{j=1}^{n} -\ln p(\vec{x}_j, y_j|\theta) - \frac{1}{n}\ln \pi(\theta)\right]$$

- STRUCTURAL RISK MINIMIZATION:

$$h^{SRM}(\mathcal{D}, \lambda) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \left[\frac{1}{n}\sum_{j=1}^{n} \ell(\vec{x}_j, y_j, h) + \frac{\lambda}{n}\,\text{complexity}(h)\right]$$

(empirical risk / data term; structure penalty / prior beliefs)

FREQUENTIST AND BAYESIAN DISCRIMINATIVE LEARNING
- FREQUENTIST: parameters θ viewed as deterministic unknowns and p estimated from training data via conditional ML

$$\hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmax}} \left[\prod_{j=1}^{n} p(y_j|\vec{x}_j, \theta)\right] = \underset{\theta}{\operatorname{argmin}} \left[\frac{1}{n}\sum_{j=1}^{n} -\ln p(y_j|\vec{x}_j, \theta)\right]$$

(analogous to ERM w/ loss function!)
- BAYESIAN: parameters θ viewed as random w/ given prior $\Pi(\theta)$ and estimated from training data $\mathcal{D}$ via conditional MAP

$$\hat{\theta}_{map} = \underset{\theta}{\operatorname{argmax}} \left[\left(\prod_{j=1}^{n} p(y_j|\vec{x}_j, \theta)\right)\pi(\theta)\right] = \underset{\theta}{\operatorname{argmin}} \left[\frac{1}{n}\sum -\ln p(y_j|\vec{x}_j, \theta) - \frac{1}{n}\ln \pi(\theta)\right]$$

- analogous to SRM w/ loss function defined by a discriminative probability model and a structure penalty (complexity regularization) term determined by the prior
- BAYESIAN DISCRIMINATIVE LEARNING AS STRUCTURAL RISK MIN.
- BAYESIAN DISCRIMINATIVE LEARNING

$$\hat{\theta}_{map} = \underset{\theta}{\operatorname{argmin}} \left[\frac{1}{n}\sum -\ln p(y_j|\vec{x}_j, \theta) - \frac{1}{n}\ln \pi(\theta)\right]$$

- STRUCTURAL RISK MINIMIZATION

$$h^{SRM}(\mathcal{D}, \lambda) = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \left[\frac{1}{n}\sum_{j=1}^{n} \ell(\vec{x}_j, y_j, h) + \frac{\lambda}{n}\,\text{complexity}(h)\right]$$

(emp. risk / data term; structure penalty / prior beliefs; size of loss function)

- Any algorithm which learns a decision rule from data, e.g. $h^{ERM}(\mathcal{D}, \lambda)$:
  - Asymptotic consistency: as $n \to \infty$ will the learned rule converge to the best rule in some family of rules, even if not in the family? Desirable property of the learning rule. Does not imply that the learned rule is any good for small n.
  - Sample complexity: how many training samples n does one need in order to guarantee that the estimated risk with the learned rule will be within some ε neighborhood of the best-in-class risk with confidence at least $1 - \delta$?
  - Computational complexity: how many time/storage/processor units are needed for training and testing
  - Complexity (capacity) of hypothesis set: VC, doubling, Natarajan, ... dimensions
- UNIVERSAL ASYMPTOTIC CONSISTENCY THEOREM: For any joint distribution $p(\vec{x}, y)$ on features $\vec{x}$ & labels y, there exists a decision rule whose risk converges to the Bayes risk as training set size $n \to \infty$
- Rate of convergence can be arbitrarily slow
  - "No free lunch theorem"
  - Each $h_n(\cdot)$ is a binary classification rule learned from a training set $\mathcal{D}$ of size n
  - For every n and every $\varepsilon \in (0, 1/2)$, there exists a joint distribution $p(\vec{x}, y)$ on features $\vec{x}$ and labels y and a training set $\mathcal{D}$ of size n such that

$$P(L(h_n(\vec{x}), y)) \geq \varepsilon$$

  - and yet with probability at least $1/2$ over the training set,

$$P(L_n(\vec{x}) \pm \varepsilon) > 1/2 - \varepsilon$$

, i.e., only slightly better than random guessing
  - similar result for regression
- No Free Lunch Theorem:
  - there is no universally best model
  - assumption that does well in one domain may work poorly in another
  - no most popular w/ wide variety of models to consider variety of real-world data

GAUSSIAN DISCRIMINANT ANALYSIS
- Classification: label = category, $y \in \mathcal{Y} = \{1, \dots m\}$, m = # of classes, $\ell(x, y, h) = \mathbb{1}(h(x) \neq y)$, Risk = $P(\mathcal{Y} \neq h(\vec{x})) = P(\text{Error})$
- Gauss. Discr. Anal.
  - parametric model for data; generative learning despite "discriminant" in name
  - feature vectors: $x \in \mathcal{X} = \mathbb{R}^d$
  - Model: $p(x, y|\theta) = p(y|\theta)\, p(x|y, \theta)$

$$= p(y) \cdot \mathcal{N}(x; \mu_{X|Y}, \Sigma_{X|Y})(x) = p(y) \cdot \frac{1}{\sqrt{(2\pi)^d}\sqrt{\det(\Sigma_{X|Y})}} \exp\left[-\frac{1}{2}(x - \mu_{X|Y})^T \Sigma_{X|Y}^{-1}(x - \mu_{X|Y})\right]$$

  - parameters: $\theta = \{(p(y), \mu_{X|Y}, \Sigma_{X|Y}),\ y = 1, \dots, m\}$
  - total # of scalar parameters: $(1 + d + d(d+1)/2)\,m$
- BAYESIAN CLASSIFIER (θ known): 0-1 loss, MPE rule = MAP rule
- Quadratic Discriminant Analysis (QDA)

$$h_{map}(x) = \underset{y=1,\dots m}{\operatorname{argmax}} p(y|x) = \underset{y}{\operatorname{argmax}} [\ln p(x|y, \theta) - \ln p(y)]$$

$$= \underset{y}{\operatorname{argmin}} \left[\frac{1}{2}(x - \mu_{X|Y})^T \Sigma_{X|Y}^{-1}(x - \mu_{X|Y}) + \frac{1}{2}\ln\det(\Sigma_{X|Y}) - \ln p(y)\right]$$

(class dependent quadratic; class dependent scale offset)

$$= h_{QDA}(x)$$

- $\sqrt{z^T \Sigma^{-1} z} = 1 \Rightarrow \|z\| = 1$ shows that length of z weighted by Σ
- thus, $(x - \mu_y)^T \Sigma^{-1}(x - \mu_y)$ is like the squared (unweighted) distance between x & μ_y
- Mahalanobis distance btw x and μ: $\sqrt{(x-\mu)^T \Sigma^{-1}(x-\mu)}$, reduces to Euclidean distance when Σ is identity matrix
- Mahalanobis: general notion of measuring # of std. devs a value is from the mean!
- Set of all x s.t. $p(x|y, \theta) =$ constant: $(x - \mu_{X|Y})^T \Sigma_{X|Y}^{-1}(x - \mu_{X|Y}) =$ constant (c) is the equation of the d-dimensional ellipsoid centered at $\mu_{X|Y}$
- principle directions of ellipsoid given by orthonormal eigenvectors of $\Sigma_{X|Y}$
- QDA in words: map x to class whose mean vector $\mu_{X|Y}$ is manually correlated with it. Correlation measured by weighted (inverse) squared distance ($\Sigma_{X|Y}$ is closer to it (closeness measured by weighted squared distances)
- LINEAR Discriminant Analysis (LDA)
- special case of QDA: all classes have same covariance matrix $\Sigma_Y = \Sigma\ \forall y$
- quadratic rule becomes linear:

$$h_{LDA}(x) = \underset{y}{\operatorname{argmin}} \left[\frac{1}{2}(x - \mu_{X|Y})^T \Sigma^{-1}(x - \mu_{X|Y}) + \frac{1}{2}\ln\det(\Sigma) - \ln p(y)\right]$$

$$= \underset{y}{\operatorname{argmin}} \left[\frac{1}{2}x^T \Sigma^{-1} x - \mu_{X|Y}^T \Sigma^{-1} x + \frac{1}{2}\mu_{X|Y}^T \Sigma^{-1} \mu_{X|Y} - \ln p(y)\right]$$

$$= \underset{y}{\operatorname{argmax}} \left[\left(\mu_{X|Y}^T \Sigma^{-1}\right)\vec{x} - \frac{1}{2}\mu_{X|Y}^T \Sigma^{-1} \mu_{X|Y} + \ln p(y)\right]$$

$\beta_y^T x$ projection; γ_y offset

$$= \underset{y}{\operatorname{argmax}} \left[\beta_y^T \vec{x} + \gamma_y\right]$$

- LDA rule in words: map x to class whose mean vector $\mu_{X|Y}$ is maximally correlated with it. Correlation measured by weighted (inverse) squared distance (Mahalanobis) using shared covariance matrix Σ for all classes
- Decision boundary between classes i, j: $(\beta_i - \beta_j)^T x + (\gamma_i - \gamma_j) = 0$
- 2-Class LDA (will make this assumption ← alternative by posterior)
- given classes $y = 0$ & $y = 1$:

$$\vec{\omega} := (\beta_1 - \beta_0) = \Sigma^{-1}(\mu_{X|1} - \mu_{X|0})$$

$$\vec{\mu} := \tfrac{1}{2}(\mu_{X|1} - \mu_{X|0}) - (\mu_{X|1} - \mu_{X|0}) \frac{(\ln p_Y(1) - \ln p_Y(0))}{(\mu_{X|1} - \mu_{X|0})^T \Sigma^{-1}(\mu_{X|1} - \mu_{X|0})}$$

$$\vec{\omega}^T(x - \vec{\mu}) \gtrless 0 \Rightarrow h_{LDA}(x) = \mathbb{1}(\vec{\omega}^T(x - \vec{\mu}) > 0)$$

LEARNING θ from DATA
- Empirical parameter estimates: For all $y = 1, \dots m$

$$n_y = |\{j \in [1:n] : y_j = y\}| = \sum_{j=1}^{n} \mathbb{1}(y_j = y)$$

$$\hat{p}(y) = \frac{n_y}{n}$$

$$QDA: \hat{\mu}_{X|Y} = \frac{1}{n_y}\sum_{j \in [1:n]: y_j = y} x_j \qquad QDA: \hat{\Sigma}_{X|Y} = \frac{1}{n_y}\sum_{j \in [1:n]: y_j = y} (x_j - \hat{\mu}_{X|Y})(x_j - \hat{\mu}_{X|Y})^T$$

$$LDA: \hat{\Sigma} = \frac{1}{n}\sum_{j=1}^{n} (x_j - \hat{\mu}_{X|Y_j})(x_j - \hat{\mu}_{X|Y_j})^T = \sum_{y=1}^{m} \hat{p}(y)\, \hat{\Sigma}_{X|Y}$$

- These are also ML (frequentist) estimates of θ based on data & they are asymptotically consistent!
- Expected values of parameter estimates: $\forall\, y = 1, \dots m$

$$\mathbb{E}[\hat{p}(y)] = p(y) \qquad \mathbb{E}[\hat{\mu}_{X|Y}\,|\,y_1 = \dots y_n = y > 0] = \mu_{X|Y}$$

$$\mathbb{E}[\hat{\Sigma}_{X|Y}|y_1, \dots y_n, n_y > 0] = \frac{(n_y - 1)}{n_y}\Sigma_{X|Y} \xrightarrow{\text{unbiased: }} \hat{\Sigma}_{X|Y} \to \frac{n_y}{n_y - 1}\hat{\Sigma}_{X|Y}$$

$$\mathbb{E}[\hat{\Sigma}|y_1, \dots y_n, n_y > 0] = \frac{n - m}{n}\Sigma \xrightarrow{\text{unbiased: }} \hat{\Sigma} \to \frac{n}{n - m}\hat{\Sigma}$$

FINAL LEARNED QDA/LDA RULE:
- plug-in decision rule: plug in the estimates of all the means & covariances into the Bayes decision rule, i.e. set $\theta \to \hat{\theta}_{ML}$ in Bayes rule
- NOTE: QDA & LDA use inverse of estimated covariance matrices
- Comp. complexity: (assuming $d \leq n$)

Method	Resource	Training	Testing (per sample)
LDA	time	$O(nd^2 + d^3) = O(nd^2)$	$O(md)$
	memory	$\Theta(md + d^2) = \Theta(d^2)$	$O(md)$
QDA	time	$O(nd^2 + md^3) = O(md^3)$	$O(md^2)$
	memory	$O(md + md^2) = O(md^2)$	$O(md^2)$

m = # classes, n = # examples, d = dimension of $\vec{x}$

NON-INVERTIBILITY OF COVARIANCE ESTIMATES
- ML estimates of covariance matrices may not be invertible if
  - $d >$ by be sum of my rank 1 matrices $\Rightarrow$ can have rank at most n
  - this is the overfitting problem again — complexity of model (# parameters) is greater than the amount of data
  - Even when $d \leq n$, ML can result in ill-conditioned (close to singular/singular) — may be due to limited numerical precision + strong deterministic (linear) dependency + redundant features
- SOLUTIONS:
- use pseudo inverse $(\cdot)^+$ instead of inverse whenever needed
- Formal square root of Σ: SVD $A = U\begin{bmatrix}\sigma_1 & & \\ & \ddots & \\ & & \sigma_d\end{bmatrix}V^T$, $A^T = V\begin{bmatrix}\sigma_1 & & \\ & \ddots & \\ & & \sigma_d\end{bmatrix}U^T$ where $\sigma^+ = \sigma^{-1}$ if $\sigma \neq 0$, 0 else
- this is equivalent to reducing the dimensionality of the feature space by eliminating / masking differentiating dimensions
- Modeling assumption: take Σ to be diagonal, i.e. make it invertible and ignore all correlations to reduce # degrees of freedom in estimate → ignore all feature dimensions, but still include weighted contributions to decision-making via prior beliefs
- KEY IDEA: Let $\hat{\Sigma}$ be a symmetric positive semi-definite matrix (not necessarily invertible, let δ be a small symmetric positive definite matrix (e.g. $\varepsilon \cdot$ identity). Then $\hat{\Sigma} + \delta$ is positive definite and invertible
- Regularization strategy 1: $\hat{\Sigma}_{reg} = \lambda\hat{\Sigma} + (1 - \lambda)(\hat{\Sigma}_{ML})$, $\lambda \in [0, 1]$, $\hat{\Sigma}$ = shrinkage estimate (towards diagonal) (towards identity) — choose λ using cross-validation
- Regularization strategy 2: $\hat{\Sigma}_{reg} = \lambda\,\text{diag}(\hat{\Sigma}_{ML}) + (1 - \lambda)(\hat{\Sigma}_{ML})$, $\lambda \in [0, 1]$ — choose λ using cross-validation
- This works because we know $\text{diag}(\hat{\Sigma}_{ML})$ is invertible since all rows should have non-zero variance!
- It can be proved that this is the MAP (Bayesian) estimate of Σ with an inverse Wishart prior $\Pi(\Sigma)$ & λ a tuning parameter
- SPECIAL CASE w $\lambda = 1$: $\hat{\Sigma}_{reg} = \text{diag}(\hat{\Sigma}_{ML}) \to$ equivalent to assuming entirely all d components are independent → "Gaussian Naive Bayes"

$y =$ likes style C, prefers P, shops at S

$$p(x|y, \theta) = p(x_1|y, \theta_{1|y}) \cdot p(x_2|y, \theta_{2|y}) \cdots p(x_d|y, \theta_{d|y})$$

$$h_{NBC}(x) = \underset{y}{\operatorname{argmax}}\, \hat{p}(y) \cdot p(x_1 = x_1|y, \hat{\theta}_{1|y}) \cdots$$

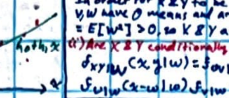
$$S = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}} \qquad \epsilon = \sqrt{\frac{\sum(x_i - \bar{x})^2}{N}}$$

e.g. POLYNOMIAL REGRESSION
- $X = Y = \mathbb{R}$
- $\mathcal{H} = \{h(x) = \sum_{k=0}^{\kappa} h_k x^k\}$
- $\ell(y, h(x)) = (y - h(x))^2$

Training: Find best $h \to h_0, h_1, \dots h_\kappa$ by MINIMIZING (empirical) MSE
- $(h_0^{best}, \dots h_\kappa^{best}) = \arg\min_{(h_0, \dots h_\kappa)} \frac{1}{n} \sum_{j=1}^{n} \left(\sum_{k=0}^{\kappa} h_k (x_j)^k - y_j \right)^2$

HOW DO WE SELECT κ? VALIDATION

Given training set $D = \{(x_1, y_1), \dots (x_n, y_n)\}$, Find best predictor $h \in \mathcal{H}$



Validation set $= \{(x_{1v}, y_{1v}), \dots (x_{mv}, y_{mv})\}$

$MSE_{valid}(\kappa) = \frac{1}{m} \sum_{j=1}^{m} \left(\sum_{k=0}^{\kappa} h_k^{best} (x_{jv})^k - y_{jv} \right)^2$

- once we get κ_*, retrain on full training set to find best h_k's

e.g. Parametric model (generative)

$Y = \{0, 1\}, X = \mathbb{R}$

$p(y) = \begin{cases} \lambda, & y = 1 \\ 1-\lambda, & y = 0 \end{cases}$ $(0 \le \lambda \le 1)$

$= \lambda^y (1-\lambda)^{1-y}$

$p(x|y=0) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu_0)^2}{2}} \to \mathcal{N}(\mu_0, 1)(x)$

$p(x|y=1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu_1)^2}{2}} \to \mathcal{N}(\mu_1, 1)(x)$

So, $p(x, y) = p(y) \cdot p(x|y) = \lambda^y (1-\lambda)^{1-y} \frac{e^{-\frac{(x-\mu_y)^2}{2}}}{\sqrt{2\pi}}$

Unknowns: $(\lambda, \mu_0, \mu_1) = \theta$, 3 real parameters

When IID examples $(x_1, y_1), \dots (x_n, y_n) \sim p(x, y|\theta)$

If you know λ, μ_0, μ_1:

$h_{MMSE}(x) = E[Y|X=x]$

$= 0 \cdot p(0|X=x) + 1 \cdot p(1|X=x)$

e.g. Non-parametric model

$p(z) = $ fn. of z

w/ Taylor series, $p(z) = a_0 + a_1 z + a_2 z^2 + \dots$ no series

↳ number of parameters not fixed!... where do we stop adding terms

$\theta = m, (a_0, a_1, \dots a_m)$

Fisher's LDA for m = 2
- similar to LDA, gaussian assumption not made, optimality criterion changed
- Model: Given $Y = y$, $X = \mu_{X|Y} + Z_y$ ← signal, zero-mean "noise" w/ cov $\Sigma_{X|Y}$
- Idea: project $X = x$ along a direction w and threshold the projection $w^T x$ to make class decision
- Best direction? How to choose threshold?
- Given $Y = y$, projection $w^T X$ is a scalar RV w/ mean $w^T \mu_{X|y}$ and variance $w^T \Sigma_{X|y} w$
- Between-class variance (signal power) of projection: $\sigma^2_{between}(\vec{w}) = (w^T \mu_{X|1} - w^T \mu_{X|0})^2$
- Within-class variance (noise power) of projection as: $\sigma^2_{within}(\vec{w}) = p_0 w^T \Sigma_{X|0} w + p_1 w^T \Sigma_{X|1} w = w^T (p_0 \Sigma_{X|0} + p_1 \Sigma_{X|1}) \vec{w}$
- Measure of separation between classes: $SNR(\vec{w}) = \frac{\sigma^2_{between}(w)}{\sigma^2_{within}(w)}$, signal-to-noise ratio (SNR)
- Best w is the one that maximizes the SNR

$w_{Fisher} = \arg\max_w SNR(\vec{w})$

MAIN RESULT: If Σ is invertible, $w_{Fisher} = \Sigma^{-1}(\mu_{X|1} - \mu_{X|0})$

Final decision rule: $h_{Fisher}(x) = 1(\vec{w}_{Fisher}^T x > \tau)$

τ can be chosen as in LDA, but in practice it is treated as a tuning parameter and its value is set via CV and...

- To calculate eigenvalues, solve equation $\det(B - \lambda I) = 0$
- orthonormal: mutually orthogonal & unit norm length
- If $D = ABC$, $tr(D)$ is symmetric w/ respect to circular reordering of A-B-C
- Conditional probability: $P(A|B) = \frac{P(A \cap B)}{P(B)}$
- Bayes Rule: $P(B|A) = \frac{P(A|B)P(B)}{P(A)}$
- Law of total probability: if $B_1, \dots B_n$ form a partition of Ω, $P(A) = \sum_{i=1}^{n} P(A|B_i) P(B_i)$
- Independence of Random variables: $P(x|y) = P(x)$ ∀ X and Y; $P(x \cap y) = P(x) \cdot P(y)$
- Conditional Independence: $P(A_1 \cap A_2 | B) = P(A_1|B) P(A_2|B)$
- Expectation: $E[X] = \sum x p(x) = \int x f(x) dx$

$E[g(X)] = \sum g(x) p(x) = \int g(x) f(x) dx$
- Variance: $var(X) = E[(X - E[X])^2] = \sigma^2$

$= E[X^2] - (E[X])^2$

★ $Var[aX + b] = a^2 Var(X)$
- Gaussian distribution:

$\mathcal{N}(\mu, \sigma^2), \mu \in \mathbb{R}, \sigma \ge 0$

$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$
- Q function:

$Q(x) := \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt = 1 - F_X(x), X \sim \mathcal{N}(0, 1)$

↳ probability that $X > x$

If Y is a Gaussian RV w/ mean μ, variance σ^2, $X = \frac{Y - \mu}{\sigma}$ is standard normal and $P(Y > y) = P(X > x) = Q(x)$ where $x = \frac{y - \mu}{\sigma}$
- Conditional density: $f_{X|Y}(x|y) = \frac{f_{XY}(x, y)}{f_Y(y)}$

★ If x, y jointly continuous & independent, $f_{X|Y}(x|y) = f_X(x)$ for all $y: f_Y(y) > 0$

conditional expectation of X given Y = y:

$E[X|Y=y] = \int_{-\infty}^{\infty} x f_{X|Y}(x|y) dx$

$g(y) := E[X|Y=y]$ is a deterministic function of Y

Law of iterated expectations:

$E[E[X|Y]] = E[X]$ inner w.r.t. $f_{X|Y}$, outer w.r.t. f_Y

$E[X|Z] = E[E[X|Y, Z]|Z]$

Correlation: $R_{XY} := E[XY] = \iint xy f_{XY}(x, y) dx dy$

$= \sum\sum xy f_{XY}(x, y)$

★ If X, Y indep, $E[XY] = E[X]E[Y]$

↳ note this can also be true if X, Y dependent!

- Covariance: $cov(X, Y) = E[(X - E[X])(Y - E[Y])]$ → $Var(X) = Cov(X, X)$

$= R_{XY} - \mu_X \mu_Y$

$Cov(X, Y) = E[X(Y - E[Y])]$

$Cov(X+Y, U+V) = cov(X, U) + cov(X, V) + cov(Y, U) + cov(Y, V)$

$Cov(aX + b, cY + d) = ac Cov[X, Y]$

$Var[X + Y] = Var[X] + Var[Y] + 2Cov[X, Y]$

$Var[aX + bY + c] = a^2 Var[X] + b^2 Var[Y] + 2ab\, cov(X, Y)$ ($2ab \rho \sigma_x \sigma_y$)

- Correlation coefficient: $\rho_{XY} = \frac{cov(X, Y)}{\sqrt{var(X) var(Y)}}$ (≤ 1)
- Cauchy-Schwarz inequality:

$|E[XY]| \le \sqrt{E[X^2] E[Y^2]}$
- Triangle Inequality:

$\sqrt{E[(X+Y)^2]} \le \sqrt{E[X^2]} + \sqrt{E[Y^2]}$

Let U, V, W be IID Unif[-0.5, 0.5] RVs. Let $X := W + U$ and $Y := W + V$.

i) Are X and Y independent? NO

In order for X & Y to be independent, they must be uncorrelated. Since U, V, W have 0 means and are independent: $cov(X, Y) = E[XY] = E[(W+U)(W+V)] = E[W^2] > 0$, so X & Y are not uncorrelated

ii) Are X & Y conditionally independent given W? YES

$f_{XY|W}(x, y|w) = f_{UV|W}(x-w, y-w|w) = f_U(x-w) f_V(y-w) = f_{U|W}(x-w|w) f_{V|W}(y-w|w) = f_{X|W}(x|w) f_{Y|W}(y|w)$

Let $W = Y + U$, where label Y and noise U are independent, $P(Y=1) = 3/4$, $P(Y=-1) = 1/4$, $U \sim$ Uniform[-3, 3]. Let observation $X = sign(W)$

a) Compute $p(x|y)$ for all $x, y \in \{-1, +1\}$

$P(X=+1|Y=+1) = P(W>0|Y=+1) = 4/6 = 2/3$

$p(x|y) =$

	x = -1	x = 1
y = 1	1/3	2/3
y = -1	2/3	1/3

c) are X & Y independent?

NO. $P(X=1|Y=1) \ne P(X=1|Y=-1)$

d) Compute $E[Y|X=x]$ as a function of x

$E[Y|X=x] = 1 \cdot P(Y=1|X=x) - 1 \cdot P(Y=-1|X=x)$

$= 2P(Y=1|X=x) - 1 = \frac{2P(X=x|Y=1) \cdot P(Y=1)}{P(X=x)} - 1$

$= \begin{cases} 5/7, & x = 1 \\ 1/5, & x = -1 \end{cases}$

- X, Y can only be independent if their graph is a rectangle (possible values of Y should not be dependent on X and vice versa)

Let $X = Y + Z$ where Y and Z are independent Gaussians w/ 0 means and variances σ_Y^2 and σ_Z^2 respectively which are KNOWN. Derive MMSE, MMAE, MAP, and ML decision rules for estimating y given a test point x. For MMSE, also derive an expression for the Bayes Risk $E[(Y - h_{MMSE}(X))^2]$.

Since Y, Z are independent Gaussians and X is their sum, all 3 are jointly Gaussian. So,

$Y|X = x \sim \mathcal{N}(E[Y|X=x], Cov(Y|X=x))$, where

$E[Y|X=x] = E[Y] + Cov(Y, X) Cov^{-1}(X)(x - E[X]) = 0 + Cov(Y, Y+Z)Cov^{-1}(Y+Z, Y+Z) x$

$= \frac{\sigma_Y^2}{\sigma_Y^2 + \sigma_Z^2} x$ (since $Cov(Y, Z) = 0$)

$Cov(Y|X=x) = Cov(Y) - Cov(Y, X) Cov^{-1}(X) Cov(X, Y)$

Thus, $Y|X = x \sim \mathcal{N}\left(\frac{\sigma_Y^2}{\sigma_Y^2 + \sigma_Z^2} x, \frac{\sigma_Y^2 \sigma_Z^2}{\sigma_Y^2 + \sigma_Z^2}\right)$

The posterior pdf of Y conditioned on X = x is a scalar Gaussian which is symmetric about the posterior mean $E[Y|X=x]$ and also reaches its maximum value there.

Thus, the posterior mean, median, & mode all coincide and the MMSE, MMAE, and MAP decision rules are all identical

$h_{MMSE}(x) = h_{MMAE}(x) = h_{MAP}(x) = E[Y|X=x] = \frac{\sigma_Y^2}{\sigma_Y^2 + \sigma_Z^2} x$

The associated MSE (Bayes risk for squared loss) is given by:

$E[(Y - h_{MMSE}(X))^2] = E[E[(Y - E[Y|X])^2|X]] = E[Cov(Y|X)]$

$= E\left[\frac{\sigma_Y^2 \sigma_Z^2}{\sigma_Y^2 + \sigma_Z^2}\right] = \frac{\sigma_Y^2 \sigma_Z^2}{\sigma_Y^2 + \sigma_Z^2}$

To derive ML decision rule, first note that $p(x|y) = \mathcal{N}(y, \sigma_Z^2)(x)$, i.e., a Gaussian w/ mean at y and variance σ_Z^2. The Gaussian attains its unique maximum value at x only if $x = y$. Thus, $h_{ML}(x) = x$

(MSE of ML would be $E[(X - Y)^2] = E[Z^2] = \sigma_Z^2$)

Bag-of-Words! (6.3)

Consider a "bag of words" Naive Bayes model for spam classification w/ the vocabulary: {customer, dollar, for, healthy, is, low, million, offer, pizza, play, price, secret, sports, today, valued}

spam: "million dollar offer for secret", "secret for price offer today", "million for dollar price"

normal: "low offer for valued customer", "play secret today", "healthy sports", "dollar pizza offer today"

a) Compute ML estimates of following params: P(spam), P(word|spam), P(word|normal) for each word in message "secret dollar offer today"

$P(spam) = \frac{3}{3+4} = 3/7 = 6/14$, $P(normal) = 8/14$

P("secret"|spam) $= \frac{3}{14}$, P("dollar"|spam) $= \frac{2}{14}$, P("offer"|spam) = 3/14

P("today"|spam) = 1/14

P("secret"|norm) = 1/14, P("dollar"|norm) $= \frac{1}{14}$, P("offer"|norm) = 2/14

P("today"|normal) = 3/14

b) How would "secret dollar offer today" be classified?

★ $P(Y_j = c | x_j) = P(Y_j = c) \prod_{i=1}^{d_j} P(w_{i,j} | Y = c)$ ★

P(sentence|spam) · P(spam) = P("secret"|spam) · P("dollar"|spam) · P("offer"|spam) · P("today"|spam) · P(spam) $= \frac{3 \cdot 2 \cdot 2 \cdot 1 \cdot 6}{14}$

P(sentence|norm) · P(norm)
= p("secret"|norm) ·
p("dollar"|norm) ·
p("offer"|norm) ·
p("today"|norm) · P(norm)
$= \frac{1 \cdot 1 \cdot 2 \cdot 3 \cdot 8}{14}$

P(spam|sentence) > p(norm|sentence), so **spam**

- X and Y are orthogonal if their correlation $R_{XY} = 0$
- X and Y are uncorrelated if their covariance = 0

★ independence ⇒ $Cov(x, y) = 0$ BUT $Cov(x, y) = 0$ ⇏

PROPERTIES OF JOINT GAUSSIAN RVS:
- any linear function of X, Y is Gaussian:

If $Z = \alpha X + bY + \gamma$, Z is Gaussian$(\mu_Z, \sigma_Z)$

$\mu_Z = \alpha\mu_X + b\mu_Y + \gamma$

$\sigma_Z^2 = \alpha^2 \sigma_X^2 + b^2 \sigma_Y^2 + 2\alpha b \rho_{X,Y} \sigma_X \sigma_Y$
- conditional expected value: $E[X|Y=y] = \mu_X + \rho_{X,Y} \frac{\sigma_X}{\sigma_Y}(y - \mu_Y)$

$Var[X|Y=y] = (1 - \rho_{X,Y}^2) \sigma_X^2$
- conditional pdf Gaussian

Uncorrelated Gaussian (jointly) RVs → independent GRVs

$E[X|Y] = E[X] + cov(X, Y) cov(Y, Y)^{-1}(y - E[Y])$

var $= cov(X, X) - cov(X, Y) cov(Y, Y)^{-1} cov(Y, X)$

Nearest Neighbor

Let $S := \{x \in \mathbb{R}^d : \|x\|_2 \le 1\}$ be a spherical ball of radius 1 in $\mathbb{R}^d$. Consider a binary classification problem in which the class labels are equally likely, i.e. $p(Y=0) = P(Y=1) = 0.5$, and the feature vector $x \in \mathbb{R}^d$ with the following class-conditional densities:

$p(x|y) = Uniform(\mu_y + S)$, $y = 0, 1$

with $\|\mu_1 - \mu_0\| > 4$. Let $D = \{(X_j, Y_j), j = 1 \dots n\}$ be n IID labeled training examples w/ joint distribution $\frac{1}{2}p(x|y)$. Let (X_{test}, Y_{test}) be a test pair drawn independently of D according to the same joint distribution $0.5p(x|y)$. Let $h_{k\text{-}NN}(x; D)$ denote the k-NN decision rule based on D where k is an odd positive integer. Since D is random, the decision rule $h_{k\text{-}NN}(x; D)$ evaluated at any point x is a random variable.

a) Compute the MAP rule and its misclassification probability.

Let $S_y = \mu_y + S = \ell_2$ ball of radius 1 in $\mathbb{R}^d$ centered at μ_y. Since the distance b/w the centers of the class 0 and class 1 balls $\|\mu_1 - \mu_0\| > 4$, any two points in the same ball S_y are closer to each other than any point in S_y is to any point in S_{1-y}

If $a, b \in S_y$, then triangle inequality says $\|a - b\|_2 \le \|a - \mu_y\|_2 + \|\mu_y - b\|_2 \le 1 + 1 = 2$

On the other hand, if $a \in S_0$, $b \in S_1$, $\|a - b\|^2 = \|(\mu_0 - \mu_1) + (a - \mu_0 + \mu_1 - b)\|^2$ ($c = a - \mu_0 + \mu_1 - b$)

$= \|\mu_0 - \mu_1\|^2 + \|c\|^2 + 2c^T(\mu_0 - \mu_1)$

$\ge \|\mu_0 - \mu_1\|^2 + \|c\|^2 - 2\|c\| \|\mu_0 - \mu_1\|$ ... CS

$= (\|\mu_0 - \mu_1\| - \|c\|)^2$

$> (4 - 2)^2$ (since $\|c\| \le \|a - \mu_0\| + \|\mu_1 - b\| \le 1 + 1 = 2$)

$\Rightarrow \|a - b\|_2 > 2$

Since class 1 & class 0 spheres have no overlap, classes perfectly separable

$h_{MAP} = \begin{cases} 0 & \|x - \mu_0\|_2 \le 1 \\ 1 & \text{otherwise} \end{cases}$, misclass. prob = 0

Not unique since decision outside support of probability distr. of x can be arbitrary

Given $E[X|Y=y] = \rho y$, $cov(X|Y=y) = (1 - \rho^2)$, express $E[(X-Y)^2|Y=y]$ in terms of ρ and y

$E[(X-Y)^2|Y=y] = E[((X - \rho y) + (\rho y - y))^2 | Y = y]$

$= E[(X - \rho y)^2 | Y = y] + (1-\rho)^2 y^2$

$= cov(X|Y=y) + (1-\rho)^2 y^2$

$= (1 - \rho^2) + (1-\rho)^2 y^2$

Let Z_1, Z_2 be IID scalar $\mathcal{N}(0, 1)$, $Z = (Z_1, Z_2)^T$, and $(X, Y)^T = AZ + (-3, 2)^T$ where $A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$

a) Compute the covariance matrix for $(X, Y)^T$

$Cov((X, Y)^T) = A \cdot Cov(Z) \cdot A^T = A \cdot I_2 \cdot A^T = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$

b) Are X and Y independent? No, X, Y correlated

c) Compute $g(x) := E[Y|X=x]$ as a function of x and $E[(Y - g(X))^2]$

$X = Z_1 - 3$, $Y = Z_1 + Z_2 + 2$, so $Y = X + Z_2 + 5$, X and Z_2 are independent.

$g(x) := E[Y|X=x] = E[X + Z_2 + 5 | X = x] = (x + 5)$

$E[(Y - g(X))^2] = E[Y - X - 5]^2 = E[Z_2^2] = 1$

Y is continuous w/ posterior pdf

$p(y|X=3) = \begin{cases} 1/8 & 0 \le y \le 4 \\ 1/2 & 5 \le y \le 6 \\ 0 & \text{otherwise} \end{cases}$

a) Compute MAP estimate of Y given x = 3

$h_{MAP}(3)$ = mode of $p(y|X=3) = [5, 6]$, not unique

b) Compute MMAE estimate of Y given X = 3

$h_{MMAE}(3)$ = median of $p(y|X=3) = [4, 5]$, not unique

c) Compute MMSE estimate of Y given X = 3

Let $Z := 1(Y \in [5, 6])$. Then,

$h_{MMSE}(3) = E[Y|X=3] = E[E[Y|X=3, Z]]$

$= P(Z=0) E[Y|X=3, Y \notin [5,6]] + P(Z=1) E[Y|X=3, Y \in [5,6]]$

$= 1/2 * 2 + 1/2 * 5.5 = \frac{7.5}{2} = 3.75$, unique

d) Let prior $p(y) = \begin{cases} 0.05 & 0 \le y \le 2 \\ 0.2 & 2 < y \le 4 \\ 0.5 & 5 \le y \le 6 \\ 0 & \text{otherwise} \end{cases}$. Compute ML estimate of Y given X = 3

Since $p(x|y) = \frac{p(y|x) \cdot p(x)}{p(y)}$, $h_{ML}(3) = \arg\max_y \frac{p(y|X=3)}{p(y)}$

$\frac{p(y|X=3)}{p(y)} = \begin{cases} \frac{1/8}{0.05} = 2.5 & y \in [0, 2] \\ \frac{1/8}{0.2} = 5/8 & y \in (2, 4] \\ .5/.5 = 1 & y \in [5, 6] \end{cases}$ So, $h_{ML}(3) = [0, 2]$, not unique

Let $z_1 = -3.1$, $z_2 = -2.9$, $z_3 = 5.3$, $z_4 = 6.2$, $z_5 = 7.5$ be 5 IID observations of a Laplacian RV Z w/ pdf $p(z) = 0.5e^{-|z - \theta|}$, where θ is an unknown mean parameter. Use ML (frequentist) learning principle to estimate θ.

$\hat{\theta}_{ML} = \arg\max_\theta p(z_1, z_2, \dots z_5 | \theta) = \arg\max_\theta \prod_{j=1}^{5} p(z_j|\theta) = \arg\max_\theta \prod_{j=1}^{5} \frac{1}{2} e^{-|z_j - \theta|}$

$= \arg\max_\theta e^{-\sum_{j=1}^{5} |z_j - \theta|}$

So, $\hat{\theta}_{ML} = \arg\min_\theta \sum_{j=1}^{5} |z_j - \theta| = \arg\min_\theta \sum_{j=1}^{5} \frac{1}{5} |z_j - \theta| = \arg\min_\theta E|Z_{emp} - \theta|$

where $Z_{emp} \sim$ Uniform$\{z_1, z_2, \dots z_5\}$. Thus $\hat{\theta}_{ML}$ is the MMAE estimate of Z_{emp}, which is equal to the median value of Z_{emp}'s distribution which is $z_3 = 5.3$

Let $\hat{\mu}_{ML}$ and $\hat{\sigma}^2_{ML}$ denote the ML estimates of the mean and variance parameters of a scalar Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ based on observing n IID training samples $x_1, \dots x_n$.

a) Derive the expressions for $\hat{\mu}_{ML}$, $\hat{\sigma}^2_{ML}$ in terms of training samples & n.

Let $\theta = (\mu, \sigma^2)$ and $D := \{x_1, \dots x_n\}$. Then $p(x|\theta) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$

For convenience, define $X_{emp} \sim$ Uniform$\{x_1, \dots x_n\}$. Then

$E[X_{emp}] = \frac{1}{n} \sum_{j=1}^{n} x_j$ $\quad Var(X_{emp}) = \frac{1}{n} \sum_{j=1}^{n} (x_j - E[X_{emp}])^2$

With these preliminaries,

$\hat{\theta}_{ML}(x_1, \dots x_n) = \arg\max_\theta p(D|\theta) = \arg\min_\theta \sum_{j=1}^{n} -\frac{1}{n} \ln p(x_j|\theta)$

$= \arg\min_\theta E[-\ln p(X_{emp}|\theta)]$

$= \arg\min_\theta E\left[\frac{(X_{emp} - \mu)^2}{2\sigma^2} + \frac{1}{2} \ln \sigma^2\right]$

$= \arg\min_{\mu, \sigma^2} \left[\frac{1}{\sigma^2}(E[X_{emp}] - \mu)^2 + \frac{Var(X_{emp})}{\sigma^2} + \ln \sigma^2\right]$

For any choice of σ^2, expression will be minimized when

$\mu = \hat{\mu}_{ML} = E[X_{emp}] = \frac{1}{n} \sum_{j=1}^{n} x_j$

Now, determine optimum value for $\hat{\sigma}^2$

$\hat{\sigma}^2_{ML} = \arg\min_{\sigma^2} \left[\frac{Var(X_{emp})}{\sigma^2} + \ln \sigma^2\right]$, set $\sigma^2 = t \cdot Var(X_{emp})$, $t \ge 0$

$= Var(X_{emp}) \cdot \arg\min_{t \ge 0} \left[\frac{1}{t} + \ln t\right]$ ↝ minimized at t = 1

So, $\hat{\sigma}^2_{ML} = Var(X_{emp}) = \frac{1}{n} \sum_{j=1}^{n} (x_j - \hat{\mu}_{ML})^2$

b) Compute $E[\hat{\mu}_{ML}]$, $E[\hat{\sigma}^2_{ML}]$ of the ML estimates w/ respect to distribution on training set. Are they unbiased?

$E[\hat{\mu}_{ML}] = E[\frac{1}{n} X_j] = \frac{1}{n} \sum E[X_j] = \mu$

$E[\hat{\sigma}^2_{ML}] = E\left[\frac{1}{n} \sum_{j=1}^{n} (X_j - \hat{\mu}_{ML})^2\right] = \frac{1}{n} \sum_{j=1}^{n} E[(X_j - \hat{\mu}_{ML})^2] = E[(X_1 - \hat{\mu}_{ML})^2]$

$= \frac{1}{n^2} E\left[(n-1)X_1 - \sum_{j=2}^{n} X_j\right]^2 = \frac{1}{n^2} E\left[E\left[(n-1)X_1 - \sum_{j=2}^{n} X_j\right]^2 \Big| X_1\right]$

$= \frac{1}{n^2} E[((n-1)X_1 - (n-1)\mu)^2 + (n-1)\sigma^2]$

$= \frac{1}{n^2}[(n-1)^2 \sigma^2 + (n-1)\sigma^2] = \frac{n-1}{n} \sigma^2 \ne \sigma^2$

So, biased estimate. Unbiased when n = 1 or if you scale by $\frac{n}{n-1}$

Unbiased estimate: $\frac{1}{n-1} \sum_{j=1}^{n} (x_j - \hat{\mu}_{ML})^2$

b) Compute $P(h_{k\text{-}NN}(X_{test}; D) \ne Y_{test})$ for k = 1, 3, 5, ...

k-NN only meaningful if n ≥ k. For k = 1:

$P(h_{1\text{-}NN}(X_{test}; D) \ne Y_{test}) = P(\text{No training feature vector} \in S_{Y_{test}})$

$= P(\{Y_1 = 0 \dots Y_n = 0, Y_{test} = 1\}$ OR $\{Y_1 = 1 \dots Y_n = 1, Y_{test} = 0\})$

$= \frac{1}{2^{n+1}} + \frac{1}{2^{n+1}} = \frac{1}{2^n}$

For k = 3: $P(h_{3\text{-}NN} \dots) = P(\text{at most 1 training feature} \in S_{Y_{test}})$

$P(\{\text{No training points} \in S_{Y_{test}}\}$ or $\{1 \text{ training point} \in S_{Y_{test}}\})$

$= \frac{1}{2^n} + \frac{\binom{n}{1}}{2^n}$

In general for k, $P(h_{k\text{-}NN}(X_{test}; D) \ne Y_{test}) = \frac{1}{2^n} \sum_{j=0}^{\lfloor k/2 \rfloor} \binom{n}{j}$

c) Compare performances of k-NN for k = 1, 3, 5 ... from best to worst

If $P_{e,k\text{-}NN}$ denotes misclassification probability of k-NN rule,

$P_{e,1\text{-}NN} < P_{e,3\text{-}NN} < \dots < P_{e,k\text{-}NN}$. 1-NN performs the best, can't exclude it from consideration!

d) Evaluate misclassification probability of kNN for each k as $n \to \infty$

$P_{e,kNN} \le \frac{1}{2^n} \binom{n}{\lfloor k/2 \rfloor} (1 + \lfloor k/2 \rfloor) \le \frac{n^{\lfloor k/2 \rfloor}}{2^n} (1 + \lfloor k/2 \rfloor) \to 0$

K-Nearest Neighbor (NN) Classifier

- **discriminative classifier**: only $p(y|\vec{x})$ estimated
- **non-parametric**: no parametric model for $p(y|\vec{x})$
- To specify classifier:
  - K: # of nearest neighbors, typically chosen to be not a multiple of m (the # of classifiers)
  - $dist: \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$, a measure of "nearness"; $dist(\vec{x}, \vec{x}')$ between $\vec{x}$ and $\vec{x}'$
- Typically, $dist$ is a metric:
  1. Positive definite: $dist(\vec{x}, \vec{x}') \geq 0$
  2. Symmetric: $dist(\vec{x}, \vec{y}) = dist(\vec{y}, \vec{x})$
  3. Triangle inequality: $dist(\vec{x}_1, \vec{x}_3) \leq dist(\vec{x}_1, \vec{x}_2) + dist(\vec{x}_2, \vec{x}_3)$
- k-NN $(\vec{x})$ = most abundant label (majority vote) among the labels of the k nearest training examples of $\vec{x}$, according to the metric

Formal description:

Given labeled training data $D = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m)\}$ and a test point $\vec{x}$.

Let $\{(\vec{x}_{(1)}, y_{(1)}), \ldots, (\vec{x}_{(m)}, y_{(m)})\}$ be a reordering of training data s.t. $d(\vec{x}, \vec{x}_{(1)}) \leq d(\vec{x}, \vec{x}_{(2)}) \leq \ldots \leq d(\vec{x}, \vec{x}_{(m)})$

THEN

$$h_{k\text{-NN}}(\vec{x}) = \text{any max}_{y \in \{1,\ldots,m\}} \sum_{j=1}^{k} I(y_{(j)} = y)$$

(= # of k NN of $\vec{x}$ w/ label $= y$)

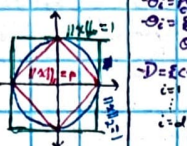
k-NN classifier = MPE/MAP rule with the above non-parametric estimate of $p(y|\vec{x})$

Common Distance Functions

- L_p distance:

$$\|\vec{x}_a - \vec{x}'\|_p = \left(\sum_{i=1}^{d} |x_i - x_i'|^p\right)^{1/p}$$

- compare this to a norm: a distance $\forall\, p \geq 1$
- Euclidean/L_2 distance: L_p distance w/ $p = 2$
- Manhattan/city-block distance: L_p distance w/ $p = 1$
- max norm/L_∞ distance: $\max_i |x_i - x_i'|$



- Mahalanobis distance: $\sqrt{(\vec{x}_a - \vec{x}')^T \Sigma^{-1} (\vec{x}_a - \vec{x}')}$, Σ a positive def. square matrix

Voronoi Regions: 1-NN classifier

- **Voronoi region**: of a training point $\vec{x}_i$ = all points in the feature space closer to $\vec{x}_i$ than to any other training point $\vec{x}_j$

$$VR(\vec{x}_i) = \{\vec{x} : d(\vec{x}, \vec{x}_i) < d(\vec{x}, \vec{x}_j),\ \forall j \neq i\}$$

- the 1-NN classifier will assign all points in $VR(\vec{x}_i)$ to $\vec{x}_i$'s class
- the Voronoi tessellation is a partition of the feature space into the Voronoi regions of the training points
- if the distance function is a norm induced by an inner product, e.g. $\|\vec{x}\| = \sqrt{\vec{x} \cdot \vec{x}}$, then the Voronoi regions are CONVEX

Universal consistance & choosing k

- Asymptotic performance guarantees:
  - as $n \to \infty$, risk of 1-NN classifier $(\text{for } 0\text{-}1 \text{ loss}) \leq 2R_{Bayes}$ for all "nice" data distributions and distance functions
  - if as $n \to \infty$:
    - $k_n \to \infty$ ... ensures increasing # of examples in majority vote)
    - $\frac{k_n}{n} \to 0$ ... ensures the k_n NN get closer to any test point)

  then, risk of k_n-NN classifiers (for 0-1 loss) $\to R_{Bayes}$ for all "nice" data
- Best value of k chosen by cross-validation (e.g. LOOCV), best choice can be $k = 1$

Computational Complexity

with k, d fixed

- Direct: $O(cnd)$
- Using multidimensional search trees:
  - $O(c\, k d \log n)$ worst case
  - $O(c \log n)$ expected time
- Approximate search:
  - locality-sensitive hashing
  - kd-trees, ball trees
  - dataset reduction (select subset of training set)
  - dimensionality reduction (select subset of features)

$$P(y_{test} \neq h_{k\text{-NN}}(\vec{x}_{test})) \leq 2R(y_{test}, h_{Bayes}(\vec{x}_{test}))$$

Euclidean Diagonal Matrix

- Let $\vec{x}$ be observation, $\vec{x}_i$ and $\vec{x}_j$ (training) can be calculated via kernel products:

$$\|\vec{x} - \vec{x}_i\|_2^2 = \langle \vec{x}, \vec{x} \rangle + \langle \vec{x}_i, \vec{x}_i \rangle - 2\langle \vec{x}, \vec{x}_i \rangle$$

- Let $\vec{x}_i = [x_{i1}, \ldots, x_{in}]$, $diag2vec(A)$ = column vector of the main diagonal elements of square matrix A, and 1 = column vector of all 1's. Then the mean square Euclidean distance matrix (EDM) is given by:

$$EDM(\vec{x}) = 1\, diag2vec(\vec{x}^T\vec{x})^T - 2\vec{x}^T\vec{x} + diag2vec(\vec{x}^T\vec{x})\, 1^T$$

$$EDM(\vec{x})(i,j) = \|\vec{x}_i - \vec{x}_j\|_2^2$$

NOTE: rank(EDM($\vec{x}$)) $\leq d+2$

Weighted Nearest Neighbor Rules

- observation: $h_{k\text{-NN}}(\vec{x}) = \text{any}\arg\max_y \sum_{j=1}^{k} I(y_{(j)} = y)$

$$= \text{any}\arg\max_y \sum_{j=1}^{k} \frac{1}{k} I(y_{(j)} = y)$$

$$= \text{any}\arg\max_y \sum_{i=1}^{m} \omega(\vec{x}, \vec{x}_i)\, I(y_i = y)$$

where $\omega(\vec{x}, \vec{x}_i) = \begin{cases} 1/k & \text{if } \vec{x}_i \in \{k\text{NNs of } \vec{x}\} \\ 0 & \text{otherwise} \end{cases}$

- Idea: use a alternative weight function $\omega(\vec{x}, \vec{x}')$ = nonnegative and decreases with increasing distance $dist(\vec{x}, \vec{x}')$. Can be interpreted as a "similarity score". Typically, $\omega(\vec{x}, \vec{x}')$ be $e^{-dist^2(\vec{x}, \vec{x}')}$ or $\frac{1}{dist(\vec{x}, \vec{x}')}$
- Example: $\omega(\vec{x}_i, \vec{x}') = \frac{1}{(\sigma\sqrt{2\pi})^d} e^{\frac{1}{2\sigma^2}\|\vec{x}_i - \vec{x}'\|^2} = \frac{1}{\alpha^d}\psi\left(\frac{\vec{x} - \vec{x}'}{\alpha}\right)$

$$\psi(\vec{x}) = NCO, I(\vec{x}_{NN})$$

- Here, α is a bandwidth parameter which controls the neighborhood size (like K in k-NN. (small $\alpha \Leftrightarrow$ small neighborhood)

$$p(\vec{x}|y) = \frac{1}{n_y} \sum_{j:y_j = y} \frac{1}{(\alpha\sqrt{2\pi})^d} e^{-\frac{1}{2\alpha^2}\|\vec{x} - \vec{x}_j\|^2} I(y_j = y)$$

$$n_y = |\{j : y_j = y\}|$$

- is a non-parametric estimate of the conditional density $p(\vec{x}|y)$ called **kernel density estimate** with the Gaussian "kernel" of bandwidth α
- The weighted nearest neighbor rule can be interpreted as implementing the MPE/MAP rule on the above kernel non-parametric estimate of the joint distribution $p(\vec{x}, y)$:

$$\hat{p}(\vec{x}, y) = \hat{p}(\vec{x}|y)\hat{p}(y) = \frac{1}{n_y} \frac{1}{(\alpha\sqrt{2\pi})^d} e^{-\frac{1}{2\alpha^2}\|\vec{x} - \vec{x}_i\|^2} I(y_i = y)$$

n_y/n

- other kernel functions: $\psi(\vec{x})$ any "nice" symmetric pdf maximizing at $\vec{x} = \vec{0}$

Kernel Density Estimator (KDE)

- given $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$, a non-parametric estimate of $p(\vec{x}) \in \mathbb{R}^d$ is given by $\hat{p}(\vec{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\alpha^d} \psi\left(\frac{\vec{x} - \vec{x}_i}{\alpha}\right)$
- $\alpha > 0$, bandwidth parameter = local neighborhood size
- $\psi(\vec{x})$: kernel function
  - pdf: ψ nonnegative & integrates to 1 over $\mathbb{R}^d$
  - typically: ψ centered at $\vec{0}$, is smooth, is symmetric around the origin, and decreases as $\|\vec{x}\| \to \infty$ (e.g. Gaussian kernel)
- As $n \to \infty$ if $\alpha \to 0$ and $n\alpha^d \to \infty$, then $\hat{p}(\vec{x}) \to p(\vec{x})$ in various senses (may need additional conditions on $\psi(\vec{x})$ or $p(\vec{x})$)

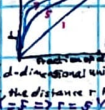
K-nearest Neighbor Regression

- Formal description: Given labeled training data $D = \{(\vec{x}_i, y_i), \ldots, (\vec{x}_m, y_m)\}$ and a test point $\vec{x}$
- Let $\{(\vec{x}_{(1)}, y_{(1)}), \ldots, (\vec{x}_{(m)}, y_{(m)})\}$ be a reordering of training data such that $d(\vec{x}_{(1)}, \vec{x}) \leq d(\vec{x}_{(2)}, \vec{x}) \leq \ldots \leq d(\vec{x}_{(m)}, \vec{x})$
- Best constant estimate (regression) is local neighborhood is:

$$h_{k\text{-NN}}(\vec{x}) = \frac{1}{k} \sum_{j=1}^{k} y_{(j)}$$

- kernel regression:

$$h_{k\text{-NN}}(\vec{x}) = \frac{\sum_{j=1}^{m} y_j \psi\left(\frac{\vec{x} - \vec{x}_j}{\alpha}\right)}{\sum_{j=1}^{m} \psi\left(\frac{\vec{x} - \vec{x}_j}{\alpha}\right)}$$



Curse of Dimensionality

- NN-classifiers (may) break down if d is large
- Suppose training points uniformly distributed in d-dimensional unit space
- To capture a fraction f of the training points, the distance r from the test point is expected to be $r = f^{1/d}$
- e.g. to capture 1% of data:
  - $d = 2$: $r = 0.1$, i.e. need to traverse 10% of range to capture 10% of examples (examples to far from a point are usually not relevant)
  - $d = 10$: $r = 0.63$
  - $d = 20$: $r = 0.89$
- To keep decision-making local for $r = 0.1 \Rightarrow$ # of points needed for classification error $= n = 10^{10}$ (!) need a lot of data to make a reliable decision
- Things get a lot worse in high dimensions, most points are far from each other and on the boundary, i.e. nearest neighbor is not necessarily near
- The curse of dimensionality refers to the phenomenon where the # of examples required for reliable decision making grows exponentially with the dimensionality of the feature space
- In LDA, we saw that a good estimate of covariance matrix Σ requires $n \gg d$
- Thus, LDA, need many more than d examples
- In practice, the n in LDA would have to be much larger than d, but unlike k-NN classification, not exponentially large in d
- This is b/c LDA is a parametric model w/ a fixed # of free parameters that does not increase with d
- Despite curse of dimensionality, k-NN may work well for large d if intrinsic dimensionality of feature space small compared to ambient dimension d, e.g. MNIST data set

Naive Bayes

- Generative/Learning: all features are conditionally independent given the class label
- ASSUMPTION: all features are conditionally independent given the class label

$$\forall y,\ p(\vec{x}_i|y, \vec{\theta}) = p(x_{i1}|y, \theta_1) \ldots p(x_{id}|y, \theta_d) = \prod_{j=1}^{d} p(x_{ij}|y, \theta_j)$$

$$\vec{\theta} = (\vec{\theta}_1, \vec{\theta}_2, \ldots \vec{\theta}_d)^T \quad \vec{\theta}_j = (p(x_{ij}|1), \ldots p(x_{ij}|m))^T$$

- Resulting MPE decision rule: Naive Bayes classifier (spite, since we do not expect features to be independent even conditioned on class label)
- Although naive, works surprisingly well in practice
- Has an over-order of $O(cnd)$ parameters to learn (cnd) $\to$ relatively immune to overfitting
- E.g. suppose all features binary, only taking vals $\in \{1, \text{true}\}$:
  - for NB model, need only 1 scalar parameter per feature & class, namely $p(x_{ij} = 1|y) = \theta_{jy}$ and $\theta_{jy} \in [0,1]$
  - Specifying the joint pdf of all binary features requires $(2^d - 1)$ scalar parameters per class or $n(2^d - 1)$ scalar parameters in total
- class conditional likelihood functions: $p(x_{ij}|y, \theta_{jy})$, $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, d\}$
  - may either be easily parametric or complicated, non-parametric for some components may be modeled parametrically and others non-parametrically
  - some/all feature components may be discrete, continuous, or mixed
- Gaussian Naive Bayes: $\forall i, y$, $p(x_{ij}|y, \theta_j) \sim N(\mu_{jy}, \sigma_{jy}^2)$
- Categorical (Naive Bayes): all features are discrete and take only a finite number of possible values

Notation summary

- y = class label $\in \{1, \ldots, m\}$ - i = sample index $\in \{1, \ldots, n\}$
- j = feature component/# index $\in \{1, \ldots, d\}$
- θ_j = parameters for class-conditional pdf/pmf of feature j across all classes
- θ_{jy} = $\{\theta_{jy} | y = 1, \ldots, m\}$, where θ_{jy} = parameters for class-conditional pdf/pmf of feature j for class y and $p(x_{ij}|y, \theta_{jy}) = p(x_{ij}|y, \theta_j)$ $\forall i, y$
- $D = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\}$ (training set)

$$i = 1 \quad \begin{matrix} x_{11} & \ldots & x_{1d} \\ \vdots & & \vdots \\ x_{i1} & \ldots & x_{id} \\ \vdots & & \vdots \\ x_{n1} & \ldots & x_{nd} \end{matrix} = \begin{pmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_i^T \\ \vdots \\ \vec{x}_n^T \end{pmatrix} \quad \begin{matrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{matrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix}$$

$$j = 1 \quad \ldots \quad d$$

- Training set: $D = \{(\vec{x}_i, y_i) : j = 1, \ldots, n\}$
- Training set for "feature j": $D_j = \{(x_{ij}, y_i), i = 1, \ldots, n\}$
- Training set for feature j & class y:

$$D_{jy} = \{(x_{ij}, y_i), i : y_i = y\}$$

- Key Result: for each feature j and each class y, the ML estimate of θ_{jy} based on D is the same as the ML estimate of θ_{jy} based on D_{jy}
- Intuition: features independent, so if each class y has d parameters of feature distribution, can be estimated independently for d features in each class

$$p(D|\vec{\theta}) = \prod_{j=1}^{d} p(x_{ij}|y, \theta_j)$$

$$\hat{\vec{\theta}}_{ML}(D) = \text{any}\arg\max_{\vec{\theta}} \frac{1}{n} \ln p(D|\vec{\theta}),$$

$$\vec{\theta} = (\theta_{11}, \theta_{21}, \ldots \theta_{d1}, \theta_{12}, \ldots \theta_{dm})$$

$$\hat{\theta}_{jy,ML}(D) = \text{any}\arg\max_{\theta_{jy}} \left[\sum_{y=1}^{m} \left(\frac{n_y}{n}\right) \ln \theta_{jy}\right]$$

$$\forall i, y,\ \hat{\theta}_{ij,ML}(D) = \text{any}\arg\max_{\theta_{ij}} \left[\frac{1}{n} \sum_{i:y_i = y} \ln p(x_{ij}|y, \theta_{ij})\right]$$

$$= \hat{\theta}_{ij,ML}(D_{jy})$$

Categorical Naive Bayes

- All features categorical: $\omega_{ij} \in \{1, \ldots, k_j\}$, $\forall i, \vec{x}_i \in \{1, \ldots, k\}$
- Notation summary:
  - y = class label $\in \{1, \ldots, m\}$
  - i = sample index $\in \{1, \ldots, n\}$
  - j = feature index $\in \{1, \ldots, d\}$
  - ω = feature value $\in \{1, \ldots, k\}$
- Model: $\forall y, j, \omega$, $P(x_{ij} = \omega | y = y) = \theta_{jy,\omega}$, where $\theta_{jy,\omega} \geq 0$, $\sum_{\omega=1}^{k} \theta_{jy,\omega} = 1$
- Parameters:

$$\theta = \{\theta_{jy,\omega} : j = 1, \ldots, d,\ y = 1, \ldots, m,\ \omega = 1, \ldots, k\}$$

$$\theta_{jy} = \{\theta_{jy,\omega} : \omega = 1, \ldots, k\}$$

- Total # of scalar parameters: mkd
- Bayes classifier for 0-1 loss (MPE/MAP rule)

$$h_{MPE}(\vec{x}) = \text{any}\arg\max_{y \in \{1,\ldots,m\}} p(y)p(\vec{x}|y, \theta)$$

$$= \text{any}\arg\max_{y} p(y) \prod_{j=1}^{d} \theta_{jx_j,y}$$

- ML estimate (frequentist) of θ: let
  - n_y = # of class y training examples
  - $n_{\omega,y,i}$ = # class y training examples with ith feature val $= \omega$
- Then $\sum_{y=1}^{m} n_y = n$ and $\forall i$, $\sum_{\omega=1}^{k} n_{\omega,y,i} = n_y$
- Also, let $n_{\omega,y}$ = # of times ω occurs across all of components in class y examples. So $\sum_{\omega=1}^{k} n_{\omega,y} = dn_y$

$$\forall y, \hat{\theta}_{0y,ML} = \hat{p}(y) = \frac{n_y}{n}$$

$$\forall \omega, y, i \quad \hat{\theta}_{\omega,y,i} = \frac{n_{\omega,y,i}}{n_y}$$

- IF $\forall \omega, y, i$ $\hat{\theta}_{\omega,y,i} = \hat{\theta}_{\omega,y,}$ then $\hat{\theta}_{\omega,y,i} = \frac{\sum_{i=1}^{d} n_{\omega,y,i}}{\sum_{i=1}^{d} n_y} = \frac{n_{\omega,y}}{dn_y}$
- Let $p_1, \ldots p_L, q_1, \ldots q_L$ be two pmfs over L items, $q_i > 0$ $\forall i$, $\sum_{i=1}^{L} q_i = 1$

$$\text{argmax}_{p} \sum_{i=1}^{L} p_i \ln q_i = \{p_i, i = 1, \ldots, L\}$$

Overfitting problem

- IF $\forall \omega > n_y$ (likely, but $\exists$ a value ω_0 which never occurs in training set but occurs in a test example, then
- For such a value, $n_{\omega,y,i} = n_{\omega,y} i = 0$ $\forall y, i$, and $\hat{p}(x_{test}|y) \hat{\theta}_{\omega,y} = 0$ for all y, and the decision reduces to random guessing. Ignores info in all values seen in training set
- SOLUTION: remove values not seen in training
- still possible $p(x_{test}|y) = 0$ by, e.g., for each class y, only have a value (ω, y) appearing in the test example but not training examples of class y
- SOLUTION: regularize estimation of θ by incorporating prior beliefs in the pdf $\pi(\vec{\theta})$

Bayesian Naive Bayes w/ Dirichlet prior

- Dirichlet distribution $Dir(\vec{q}|\vec{\alpha})$: a family of continuous multivariate probability distributions parameterized by a vector $\vec{\alpha}$ of positive reals called concentration parameters
- It is a pdf over all pmfs over W items, i.e., pdf over $(W-1)$-dim. probability simplex:

$$S_W = \{\vec{q} : 0 \leq q_i \leq 1, i = 1, \ldots, W, \sum_{i=1}^{W} q_i = 1\}$$

$$Dir(\vec{q}|\vec{\alpha}) = \frac{\{\text{if } \vec{q} \in S_W\}}{Z(\vec{\alpha})} \prod_{i=1}^{W} q_i^{\alpha_i - 1}$$

$$Z(\vec{\alpha}) = \text{normalization constant} = \frac{\prod_{i=1}^{W} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{W} \alpha_i\right)}$$

- $\alpha_0 = \sum_{i=1}^{W} \alpha_i$ controls how peaked the distribution is for a fixed $\vec{\alpha}$ (larger $\alpha_0 \Rightarrow$ more peaked)
- $Dir(1, 1)$ uniform over probs. simplex
- $Dir(3, 3, 3)$ a broad distribution centered at $(1/3, 1/3, 1/3)$
- $Dir(30, 30, 30)$ narrow distribution centered at $(1/3, 1/3, 1/3)$
- IF $\forall i < 1$ $\forall i$, we get "spikes" at the corners of the probability simplex
- $Dir(1, 1, \ldots, 1) = \frac{1}{Z(\vec{\alpha})} \prod_{i=1}^{W} (p_{w,y,i})^{\alpha_{w,y,i} - 1}$
- MAP (Bayesian) estimate of $\vec{\theta}$:

$$\hat{\vec{\theta}}_{MAP} = \text{argmax}_{\vec{\theta}}\, p(\vec{\theta}|D) = \text{argmax}_{\vec{\theta}}\, p(D|\vec{\theta})\pi(\vec{\theta})$$

$$= \text{argmax}_{\vec{\theta}} \left[\prod_{j=1}^{n} p(y_j)\right] \cdot \left[\prod_{\omega=1}^{W} \prod_{y=1}^{m} \prod_{i=1}^{d} (p_{w,y,i})^{n_{w,y,i} + \alpha_{w,y,i} - 1}\right]$$

- MAP (Bayes) estimate of $\vec{\theta}$ solution:

$$\forall y, j, \hat{\theta}_{y,j} = \frac{n_y}{n}, \quad \forall \omega, y, i, \hat{\theta}_{\omega,y,i} = \frac{n_{w,y,i} + \alpha_{w,y,i} - 1}{n_y + \sum_{\omega=1}^{W} (\alpha_{w,y,i} - 1)}$$

- IF $\forall \omega, y, i$, $\alpha_{w,y,i} = \alpha_{w,y}$

$$\forall \omega, y, i \quad \hat{\theta}_{\omega,y,i} = \frac{n_{w,y} + \alpha_{w,y} - 1}{n_y + \sum_{\omega=1}^{W} (\alpha_{w,y} - 1)} \quad (\alpha_{w,y,i} = \alpha_{w,y})$$

REMARKS

- α's can be interpreted as "prior" counts and the MAP solution as updating these prior counts w/ empirical counts from the training data
- IF all $\alpha_{\omega} = 2$, prior counts = 1, referred to as add-one smoothing / Laplace's rule of succession of the maximum likelihood
- Can incorporate a separate Dirichlet prior for $p(y)$ in a similar way
- ML estimates of $\vec{\theta}$ asymptotically consistent!

Logistic Regression

- Discriminative/Learning: only $p(y|\vec{x})$ estimated
- Parametric $p(y|\vec{x})$: LDA posterior

$$p(y|\vec{x}, \vec{\theta}) = e^{\vec{w}_y^T \vec{x}_{ext} + y_y} = \frac{e^{\vec{w}_y^T \vec{x}_{ext}}}{\sum_{k=1}^{m} e^{\vec{w}_k^T \vec{x}_{ext} + y_k}} = \frac{e^{\vec{w}_y^T \vec{x}_{ext}}}{\sum_{k=1}^{m} e^{\vec{w}_k^T \vec{x}_{ext}}} = \text{softmax}(y_y | \vec{w}_{ext}) \quad (\text{def'n})$$

$$y \in \{1, \ldots, m\}$$

where $\forall k$, $\vec{w}_k^T = [\vec{\beta}_k^T\ y_k]$, $\vec{x}_{ext} = [\vec{x}^T\ 1]^T$, $\vec{\theta} = (w_1, \ldots w_m) \in \mathbb{R}$

- NOTE: $\vec{\theta} = [w_1, \ldots, w_m]$ and $\vec{\theta}' = [w_1 - w_m, \ldots, w_{m-1} - w_m]$ are EQUIVALENT PARAMETERIZATIONS of the exact same model, i.e. a model $w\vec{\theta}$ and another $\vec{\theta}'$ are indistinguishable
  - to ensure identifiability (uniqueness), we may assume that $w_m = \vec{0}$ ($\Rightarrow$ there are only (dm) $\vec{\theta}$ or $(m-1)$ scalar parameters
- MPE/MAP classifier:

$$h_{MPE}(\vec{x}) = \text{any}\max_{y \in \{1,\ldots,m\}} p(y|\vec{x}, \vec{\theta}) = \text{any}\max_{y \in \{1,\ldots,m\}} \vec{w}_y^T \vec{x}_{ext}$$

- a linear classifier as in LDA, but the weights are learned by maximizing the conditional likelihood
- ESTIMATION OF PARAMETERS

$$\hat{\vec{\theta}}_{ML}(D) = \text{argmax}_{\vec{\theta}} \prod_{j=1}^{n} p(y_j|\vec{x}_j, \vec{\theta})$$

$$= \text{argmax}_{\vec{\theta}} \left\{\sum_{j=1}^{n} \ln\left(\sum_{k=1}^{m} e^{\vec{w}_k^T \vec{x}_{ext}}\right) - \sum_{k=1}^{m} \vec{w}_k^T \left(\sum_{j:y_j=k}^{n} \vec{x}_{ext}\right)\right\}$$

- Negative Log Likelihood (NLL)

$$NLL(w_1, \ldots w_m) = \sum_{j=1}^{n} \ln\left(\sum_{k=1}^{m} e^{\vec{w}_k^T \vec{x}_{ext}}\right) - \sum_{k=1}^{m} \vec{w}_k^T \left(\sum_{j:y_j=k}^{n} \vec{x}_{ext}\right)$$

- Gradient of NLL w.r. respect to $w_y, y = 1, \ldots, m$ is given by

$$\nabla_{w_y} NLL(w_1, \ldots w_m) = \sum_{j=1}^{n} \left(\frac{e^{\vec{w}_y^T \vec{x}_{ext}}}{\sum_{k=1}^{m} e^{\vec{w}_k^T \vec{x}_{ext}}} - I(y_j = y)\right) \vec{x}_{ext}^T$$

- The Hessian $\nabla_{\vec{\theta}}^2 NLL(\vec{\theta})$ can be shown to be a positive semi-definite matrix $\Rightarrow$ $NLL(\vec{\theta})$ is a convex differentiable function of $\vec{\theta}$
- $\Rightarrow$ local minimum of $NLL(\vec{\theta})$ = global minimum
- Gradient descent algorithm can find a global minimizer

Gradient (Steepest) Descent Algorithm

- Initialization: $\vec{\theta}_0 = (w_1^{(0)}, \ldots, w_m^{(0)})$ = all 0's or random
- For $t = 1, 2, \ldots$, until Stopping Criterion is met do:
  - Evaluate gradients: $\nabla_{w_1} NLL(\vec{\theta}_t), \ldots, \nabla_{w_m} NLL(\vec{\theta}_t)$
  - Update weights: $w_y^{(t+1)} = w_y^{(t)} - \eta_t \nabla_{w_y} NLL(\vec{\theta}_t)$, $y = 1, \ldots, m$
- end For

Stopping criterion: e.g. # iterations $\geq t_{max}$, $\|\vec{\theta}_{t+1} - \vec{\theta}_t\| \leq \epsilon$, NLL does not change much, Gradients almost 0

- Step size:
  - Fixed: $\eta_t = \eta$, $\forall t$: too small $\to$ slow convergence, too large $\to$ fail to converge
  - Adaptive (Line search): $\eta_t = \text{argmin}_{\eta \geq 0} NLL(\vec{\theta}_t - \eta \nabla NLL(\vec{\theta}_t))$
    - scalar optimization problem
    - Derivative with η: $\Rightarrow \eta = \eta_t$ satisfies following equation:

$$\nabla^T NLL(\vec{\theta}_t - \eta_t \nabla NLL(\vec{\theta}_t)) \cdot \nabla NLL(\vec{\theta}_t) = 0$$

gradient at end of step = gradient at beginning of step

- Either gradient at end of step $= 0$ $\Rightarrow$ global minimum found or gradient at end of step $\perp$ gradient at beginning of step: ZIG-ZAG path (see picture on next slide)

- Other Algorithms: Stochastic gradient descent, batch gradient descent, method of conjugate gradients, Newton's method, Quasi-Newton method

REGULARIZATION

- is generally needed when $n \ll d$, in logistic regression may be needed when $n \gg d$
- suppose classes are perfectly linearly separable for some $\vec{\theta} \Rightarrow$ also linearly separable for $\frac{1}{\epsilon}\vec{\theta}$ for any $\epsilon > 0$
- think of the limit which is given by softmax parameterized when $T \to 0$ which converges to the softmax function as the max function and assigns the max probability mass to the true class.
- $\Rightarrow$ unbounded minimizers of NLL when $\|\vec{w}_y\| \to \infty$ for at least one y, making gradient descents unstable $\to$ reg. needed

L_2 regularization

- IID Gaussian prior on weights: $\pi(\vec{\theta}) = \pi(w_1, \ldots w_m) = \prod_{k=1}^{m} N(\vec{0}, \Sigma_{reg})(w_k)$, $\Sigma_{reg} > 0$

$$\hat{\vec{\theta}}_{Bayes}(D) = \text{argmax}_{\vec{\theta}} (-\ln p(D|\vec{\theta}) - \ln \pi(\vec{\theta}))$$

$$= \text{argmin}_{\vec{\theta}} J(\vec{\theta})$$

$$J(\vec{\theta}) = NLL(\vec{\theta}) + \frac{1}{2} \sum_{k=1}^{m} w_k^T \Sigma_{reg}^{-1} w_k$$

$$\nabla_{w_y} J(\vec{\theta}) = \nabla_{w_y} NLL(\vec{\theta}) + \Sigma_{reg}^{-1} w_y$$

- $J(\vec{\theta})$ strictly convex and differentiable since for each k, $w_k^T \Sigma_{reg}^{-1} w_k$'s strictly convex & differentiable, and Σ convex $\Rightarrow$ convex
- So, $J(\vec{\theta})$ has unique global min.

Naive Bayes (Generative)	Logistic Regression (Discriminative)
• typically fewer parameters	
• stronger modeling assumption $\Rightarrow$ less robust to model mismatch	• weaker modeling assumption $\to$ more robust to model mismatch
• converges faster w/ increasing data to a less-accurate classifier	• converges slower w/ increasing data to more accurate classifier
• easier to learn/update since each class's parameters optimized separately, but each class has only a fraction of the total # of training examples	• joint optimization: harder to update, because all training examples in optimization

Convexity

- Line segment: $\forall \vec{x}_a, \vec{x}_b \in \mathbb{R}^d$ joining $\vec{x}_a$ and $\vec{x}_b$ in $\mathbb{R}^d$ given by: $\{\vec{x} = (1-\lambda)\vec{x}_a + \lambda\vec{x}_b : \lambda \in [0,1]\}$ (intersection of convex sets convex)
- convex set: a set $C \subseteq \mathbb{R}^d$ is called convex if for any two points $\vec{x}_a, \vec{x}_b \in C$, $\{(1-\lambda)\vec{x}_a + \lambda\vec{x}_b : \lambda \in [0,1]\} \subseteq C$
- Convex function: let C be a convex set, $f: C \to \mathbb{R}$ a real-valued function over C. f is a convex function over $C \Leftrightarrow$ the region above the graph of f over C (epi(f)) is a convex set
  - $\Rightarrow$ a chord joining any two points on the graph never goes below the graph
  - $\forall \vec{x}_a, \vec{x}_b \in C$, $\forall \lambda \in [0,1]$, line $f((1-\lambda)\vec{x}_a + \lambda\vec{x}_b) \leq (1-\lambda)f(\vec{x}_a) + \lambda f(\vec{x}_b)$
  - if C convex, f convex $\Leftrightarrow$ any line segment in C lies above the graph of f
  - if equality only when $\lambda = 0$ or 1 or $\vec{x}_a = \vec{x}_b$, then f is strictly convex
  - Real-valued affine function: $f(\vec{x}) = \vec{a}^T\vec{x} + b$, $\forall \vec{x} \in \mathbb{R}^d$, $b \in \mathbb{R}$, $\vec{a} \in \mathbb{R}^d$ CONVEX but not strictly
  - Subtract sets of a convex function: set of points $\exists\, x \in C : f(x) \leq s$, s convex
- Jensen's inequality: If f convex over C, x a random vector w/ support contained in C, $E[x] \in C$, $E[f(x)]$ has all means & weights
  - if f strictly convex, equality attained iff x constrained w/ probability 1
- Concave: f concave $\Leftrightarrow$ $-f$ convex, f strictly concave $\Leftrightarrow$ $-f$ strictly convex
- Operations preserving convexity:
  - f convex $\Leftrightarrow$ $sa + b$, $a \geq 0$
  - sum of convex functions convex
  - max of convex functions is convex
  - If f convex, g non-decreasing & convex, $h(\vec{x}) = g(f(\vec{x}))$ convex
  - If $f(x)$, $x \in \mathbb{R}^d$ convex, so is $h(z) = f(Az + b)$ for any $d \times d$ matrix A, $b \in \mathbb{R}^d$
- Continuity: If f convex over C, it is continuous over C's interior (at most countably many discontinuity at C's boundary)
- Differentiability: f differentiable over C if gradient vector $\nabla f(\vec{x}) = \left(\frac{\partial f(\vec{x})}{\partial x_1}, \ldots\right)^T$ exists $\forall \vec{x} \in C$
  - f convex over $C \Leftrightarrow$ its graph never goes below tangent plane at any point: $\forall \vec{x}_a, \vec{x} \in C$, $f(\vec{x}) + (\vec{x}_a - \vec{x})^T \nabla f(\vec{x}) \leq f(\vec{x}_a)$
- Twice differentiable: if f twice diff'ble, its partial derivs:

$$\nabla^2 f(\vec{x}) = \begin{pmatrix} \frac{\partial^2 f(\vec{x})}{\partial x_1^2} & \ldots & \frac{\partial^2 f(\vec{x})}{\partial x_1 \partial x_d} \\ \vdots & & \vdots \\ \frac{\partial^2 f(\vec{x})}{\partial x_d \partial x_1} & \ldots & \frac{\partial^2 f(\vec{x})}{\partial x_d^2} \end{pmatrix}$$

f convex over $C \Leftrightarrow \nabla^2 f(\vec{x})$ positive semi-def. for all $\vec{x} \in C$

$\Leftarrow$ if $\nabla^2 f(\vec{x}) > 0$ $\forall \vec{x} \in C$, f strictly convex

$\nabla^2 f(\vec{x}) \geq 0$ $\forall \vec{x} \in C$

Convex Optimization

- An optimization problem = the form $\min_{\vec{x} \in C} f(\vec{x})$ in which
  1. C is a convex set
  2. $f(\vec{x})$ a convex function over C
- If C closed, minimum over C guaranteed to exist
- If C closed & f strictly convex over C the minimizer exists and is unique
- If C closed & $f_1, f_2, \ldots, f_n$ be convex functions over $C \subseteq \mathbb{R}^d$

$$\min_{\vec{x} \in C} f(\vec{x}) \text{ subject to } g_j(\vec{x}) \leq 0,\ j \in \{1, \ldots, n\}$$

is a convex optimization problem

- Lagrange function (or saddle point): $\forall \vec{x} \in C$, $\forall \lambda_1, \ldots \lambda_n \geq 0$

$$L(\vec{x}, \vec{\lambda}) = f(\vec{x}) + \sum_{j=1}^{n} \lambda_j g_j(\vec{x})$$

- Slater's condition (weak constraint qualification): $\exists$ $\vec{x}_0 \in C$ s.t. $\forall j \in \{1, \ldots, n\}$ either $g_j(\vec{x}_0) < 0$ or $g_j(\vec{x}_0) = 0$ and g_j is affine

Constrained Opt. - Lagrangian Duality / KKT Theorem

- $\min_{x \in C} f(x)$ s.t. $g_i(x) \le 0$, $i = 1,\dots,m$, $h_j(x) = 0$, $j = 1,\dots,r$
- f convex & differentiable, $C \subseteq \mathbb{R}^n$ convex, g_i convex & differentiable, h_j affine
- Slater's condition: $\exists \tilde{x} \in C$ s.t. $g_i(\tilde{x}) < 0$ $\forall i$
- Lagrangian: $L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{r} \mu_j h_j(x)$
- Lagrangian dual function: $\theta(\lambda, \mu) = \min_{x \in C} L(x, \lambda, \mu)$
- Dual problem: $\max_{\lambda \ge 0, \mu} \theta(\lambda, \mu)$
- Strong duality: $\min_{x \in C} \max_{\lambda \ge 0, \mu} L(x, \lambda, \mu) = \max_{\lambda \ge 0, \mu} \min_{x \in C} L(x, \lambda, \mu)$
- Complementary slackness: $\lambda_i g_i(x) = 0$ $\forall i$
- KKT conditions: $\nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x) + \sum_{j=1}^{r} \mu_j \nabla h_j(x) = 0$, $\lambda_i g_i(x) = 0$, $g_i(x) \le 0$, $h_j(x) = 0$, $\lambda_i \ge 0$

Duality

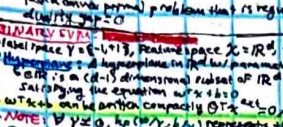
- Lagrange dual function associated to the primal optimization problem defined by $F(\lambda_1,\dots,\lambda_m) = \min_{x \in C} F(x, \lambda)$, i.e. $\forall \lambda_1,\dots,\lambda_m \ge 0$, $F(\lambda) = \min_{x \in C} \left(f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \right)$
- If $\lambda = (\lambda_1,\dots,\lambda_m)^T$, then $F(\lambda)$ a concave function of λ
- The optimization problem $\max_{\lambda \ge 0} F(\lambda)$ is called the dual opt. problem associated to the primal optimization problem
- If Karush-Kuhn-Tucker theorem conditions satisfied, $\max_{\lambda \ge 0} F(\lambda) = \min_{x \in C} f(x)$ subject to $g_i(x) \le 0$, $i \in [1,m]$

Remarks: even if the primal problem is non-convex
- dual function is concave
- dual function never above primal minimum $\Rightarrow$
- dual max never above primal min
- duality gap = primal min − dual max ≥ 0
- for a convex primal problem that is regular (Slater's condition), $d(w) = p^* = 0$

BINARY SVM

Input space $\mathcal{X} = \mathbb{R}^d$, Label space $\mathcal{Y} = \{-1, +1\}$, Feature space $\mathcal{X} = \mathbb{R}^d$, Training set $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$

- Hyperplane: a hyperplane in the d parameters are $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ and h is a $(d-1)$ dimensional subset of $\mathbb{R}^d$ consisting of all points $x \in \mathbb{R}^d$ satisfying the equation $w^T x + b = 0$
- $w^T x + b$ can be written compactly $\theta^T x' = 0$, where $\theta = (w, b)$ and $x' = (x, 1)$
- NOTE: $\forall \gamma \ne 0$, $hp(w, b) = hp(\gamma w, \gamma b)$ represents the same hyperplane
- Euclidean distance of $x \in \mathbb{R}^d$ from $hp(w, b)$: $\frac{|w^T x + b|}{\|w\|}$
- Hyperplane classifier: a binary classifier of the form $h(x) = \text{sign}(w^T x + b)$ for parameters $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$



- Linearly separable training set: a training set is linearly separable if there is a hyperplane classifier $hp(w, b)$ which classifies all training samples w/ error: $\forall i = 1,\dots,n$, $\text{sign}(w^T x_i + b) = y_i$, i.e. $y_i (w^T x_i + b) > 0$
- Canonical form of separating hyperplanes: let $hp(w, b)$ linearly separate D. We say $hp(w, b)$ is in canonical form w.r.t. D if $\min_{i=1,\dots,n} |w^T x_i + b| = 1$
- If $\gamma = \min_{i=1,\dots,n} y_i (w^T x_i + b) > 0$, then $hp(w/\gamma, b/\gamma)$ is canonical
- If hp not in canonical form, $\min_{i=1,\dots,n} |w^T x_i + b| = \gamma$
- Perceptron Algorithm to determine θ (Batch)

Input: Training set $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$
Initialize: $\theta_1 = (0, \dots, 0)^T$
For $t = 1, 2, \dots$
  If $y_i (\theta_t^T x_i) \le 0$ for some $(x_i, y_i) \in S$ then
    $\theta_{t+1} = \theta_t + y_i x_i$
  else
    output θ_t
  end if
end for

Hard Margin SVM (Linearly Separable Case)

- Geometric margin: the geometric margin $\rho(w, b, D)$ of a hyperplane $hp(w, b)$ which linearly separates a training set D is defined as its distance to the closest example: $\rho(w, b, D) = \min_{i=1,\dots,n} \frac{|w^T x_i + b|}{\|w\|}$
- Hard-margin SVM: the hard-margin SVM hyperplane $hp(w_{SVM}, b_{SVM})$ is the hyperplane which maximizes the geometric margin $\rho(w, b, D)$ among all classifiers which linearly separate the training set, i.e. $\forall i$, $y_i (w^T x_i + b) > 0$
- $(w_{SVM}, b_{SVM}) = \arg\max_{(w,b):\, \forall i,\, y_i (w^T x_i + b) > 0} \rho(w, b, D) = \arg\max_{(w,b):\, \forall i,\, y_i (w^T x_i + b) > 0} \frac{1}{\|w\|}$
- $(w_{SVM}, b_{SVM}) = \arg\min_{(w,b):\, \forall i,\, y_i (w^T x_i + b) \ge 1} \frac{1}{2}\|w\|^2$
- Convex Hull: the convex hull of a set of vectors $U = \{u_1, \dots, u_k\}$ in $\mathbb{R}^d$, denoted by $\text{conv}(U)$, is the smallest convex set containing them

PROPOSITION: the hard-margin SVM is the unique canonical hyperplane which bisects the line segments joining the closest pair of points from the positive and negative convex hulls

$(x_+, x_-) = \arg\min_{(u,v) \in C_+ \times C_-} \|u - v\|$, $w_{SVM} = \frac{2(x_+ - x_-)}{\|x_+ - x_-\|^2}$ UNIQUE, $b = -w_{SVM}^T \frac{(x_+ + x_-)}{2}$

- since convex weights are nonnegative and add to 1, $w_{SVM} = \sum_{j=1}^{n} \alpha_j^{SVM} y_j x_j$, $\alpha_j^{SVM} \ge 0$ $\forall j$, $\sum_{j=1}^{n} \alpha_j^{SVM} y_j = 0$

Hard Margin Observations

- Even though w_{SVM} is a linear combination of feature vectors, not all of them participate in the linear combination that defines w_{SVM}. The ones for which $\alpha_j > 0$ and such feature vectors in the training set are called "support vectors"
- All support vectors must lie on the same marginal hyperplanes and there is at least one support vector on each marginal hyperplane
- Even though w_{SVM} is unique, the support vectors for a hard margin SVM, which characterize it, need not be unique

Summary SVM

$(w_{SVM}, b_{SVM}) = \arg\min_{(w,b)} \left[\frac{1}{2}\|w\|^2 + \text{penalty}(\xi_1, \dots, \xi_n) \right]$
$\forall i,\ y_i (w^T x_i + b) \ge 1 - \xi_i$
$\xi_i \ge 0$

- penalty $(\xi_1, \dots, \xi_n)$ is a nonnegative function which is nondecreasing with respect to each ξ_i; i.e. if for all i, $\xi_i' \ge \xi_i$ then penalty$(\xi_1', \dots, \xi_n') \ge$ penalty$(\xi_1, \dots, \xi_n)$
- Typically, the penalty function is also convex and differentiable
- For instance, we can recover hard-margin SVM solution by setting the penalty function
1. Proportional Penalty: $C \sum_{j=1}^{n} \xi_j$, $C > 0$
2. Balanced and proportional penalty: $C_+ (\sum_{j:y_j=+1} \xi_j) + C_- (\sum_{j:y_j=-1} \xi_j)$, $C_+, C_- > 0$
   Good for unbalanced datasets
   Can set $C_+ n_+ = C_- n_- = C/2$ for some $C > 0$
3. General proportional penalty: $\sum_{j=1}^{n} C_j \xi_j$, $C_j \ge 0$ $\forall j$
4. ℓ^p penalty: $C \sum_{j=1}^{n} \xi_j^p$, $C > 0$, $p \ge 1$

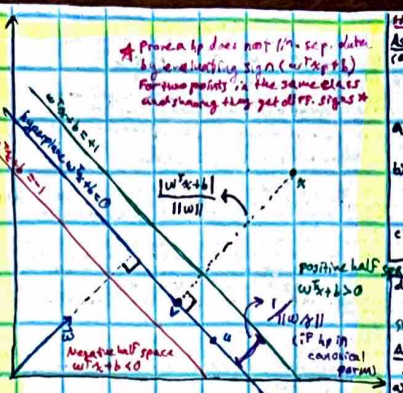
PROPOSITION: If the penalty function is a nondecreasing function of each ξ_j, then $(w_{SVM}, b_{SVM}) = \arg\min_{(w,b)} \left[\frac{1}{2}\|w\|^2 + \text{penalty}(\text{hinge}(y_1 (w^T x_1 + b)), \dots, \text{hinge}(y_n (w^T x_n + b))) \right]$
where $\text{hinge}(\epsilon) := \max(0, 1 - \epsilon)$
$\text{hinge}(\epsilon) = 1 - \epsilon$ if $\epsilon \le 1$, 0 otherwise

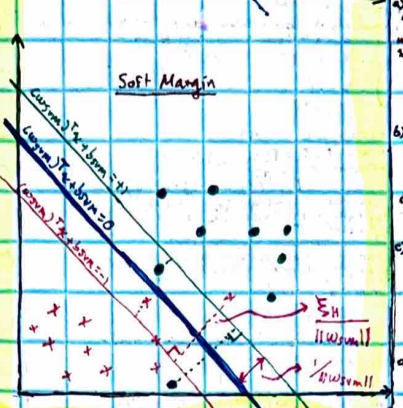
- Soft-Margin SVM: Allowing misclassified points, allowing for the general proportional penalty, the solution to the soft margin optimization problem exists. Solution for w_{SVM} is unique and has the following form:
$w_{SVM} = \sum_{j=1}^{n} \alpha_j^{SVM} y_j x_j$
where $\alpha^{SVM} = (\alpha_1^{SVM}, \dots, \alpha_n^{SVM})^T$ is the solution to the dual optimization problem

$\max_{\alpha_1,\dots,\alpha_n} \sum_{j=1}^{n} \alpha_j - \frac{1}{2} \sum_{k=1}^{n} \sum_{l=1}^{n} \alpha_k \alpha_l y_k y_l (x_k^T x_l)$
$0 \le \alpha_j \le C_j$, $\sum_{j=1}^{n} \alpha_j y_j = 0$

For each j, $\alpha_j^{SVM} (1 - \xi_j^{SVM} - y_j (w_{SVM}^T x_j + b)) = 0$ and $b_{SVM} = \arg\min_{b} \sum_{j=1}^{n} C_j \text{hinge}(y_j (w_{SVM}^T x_j + b))$

$(C_j - \alpha_j^{SVM}) \cdot \xi_j^{SVM} = 0$





Exam problems

1. Let $Z_1, Z_2 \sim$ i.i.d. scalar vars $N(0,1)$. Let $X = 3Z_1 + 2Z_2 - 1$, $Y = Z_2 + 4Z_2 + 1$
   a) Compute cov. matrix: $\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} + \begin{pmatrix} -1 \\ 1 \end{pmatrix}$
   $\text{Cov}(X,Y) = \text{Cov}(3Z_1 + 2Z_2 - 1, Z_2 + 4Z_2 + 1) = \text{Cov}(3Z_1 + 2Z_2, Z_2 + 4Z_2) = AA^T = \begin{pmatrix} 13 & 11 \\ 11 & 17 \end{pmatrix}$
   b) Are X and Y dependent? YES, Correlation $\ne 0$
   c) Compute MLE, MAP, PEL estimate of Y based on observing $X = x$
   Since X, Y jointly Gaussian, Y conditioned on X Gaussian:
   $E[Y|X=x] = E[Y] + \text{Cov}(Y,X)\text{Cov}^{-1}(X)(x - E[X]) = 0 + \frac{11}{13}(x+1)$
   $\text{Cov}(Y|X=x) = \text{Cov}(Y,Y) - \text{Cov}(Y,X)\text{Cov}^{-1}(X)\text{Cov}(X,Y)$
   $= 16.9$
   So, $Y|X=x \sim N(0.1+\frac{11}{13}x, 16.9)$
   MMSE: $h_{MMSE}(x) = E[Y|X=x] = 0.1 + \frac{11}{13}x$
   MAP: posterior mode coincides w/ posterior mean since $Y|X=x$ is Gaussian, so $h_{MAP}(x) = h_{MMSE}(x)$
   ML: estimate of Y given x is the value of y which maximizes the likelihood $p(x|y)$. Since $f(x|y)$ is scalar Gaussian w/
   $E[X|Y=y] = E[X] + \text{Cov}(X,Y)\text{Cov}^{-1}(Y)(y - E[Y])$
   $= \frac{11}{17}(y - 1) + 1$
   Since pdf of a Gaussian is maximum at the mean, the likelihood is maximized when y is chosen s.t. $E[X|Y=y] = x$ coincides as a function of x.
   i.e. $x = \frac{11}{17}y - \frac{11}{17} + 1 \Rightarrow \hat{y}_{ML}(x) = \frac{17}{11}x + 1$
   d) Compute MSE of MMSE estimator
   $MSE = E[(Y - h_{MMSE}(X))^2] = E[E[(Y - E[Y|X])^2 | X]]$
   $= E[\text{Cov}(Y|X)] = E[16.9] = 16.9$
2. We are given the following labeled dataset for a binary class. problem

i	1	2	3	4	5	6	7	8	9	10
x_i	−7	−1	+2	+2	−3	−2	1	+1	+2	6
y_i	−1	−1	−1	+1	+1	+1	−1	−1	+1	+1

   Use the following parametric generative model $p(x, y | \theta_1, \theta_2)$ where $P(Y=1) = P(Y=2) = 1/2$, θ_1, θ_2 are scalar parameters, and x is independent of y and x and y are conditionally independent given θ_1, θ_2. Assume $p(x|\theta_1) = \frac{1}{2} e^{-|x - \theta_1|}$, labels/data are assumed to be IID $\sim p(x, y | \theta_1, \theta_2)$.
   a) Compute Bayes decision rule for the model for 0-1 loss function in terms of θ_1, θ_2.
   $h_{MAP}(x) = \arg\max_{y \in \{1,2\}} p(y|x, \theta_1, \theta_2) = \arg\max_{y \in \{1,2\}} p(x|y, \theta_1, \theta_2) p(y)$
   $= \arg\max_{y \in \{1,2\}} 0.5\, p(x|y, \theta_1, \theta_2) = \arg\max_{y \in \{1,2\}} 0.25\, e^{-|x - \theta_y|} = \arg\min_{y \in \{1,2\}} |x - \theta_y|$
   $h_{MAP}(x) = \begin{cases} 1 & \text{if } |x - \theta_1| \le |x - \theta_2| \\ 2 & \text{otherwise} \end{cases}$
   b) Use ML (Principle of) learning principle to estimate θ_1, θ_2 based on training data $(\theta_{1,ML}, \theta_{2,ML}) = \arg\max_{\theta_1, \theta_2} p((x_1, y_1), \dots, (x_{10}, y_{10}) | \theta_1, \theta_2)$
   $= \arg\max_{\theta_1, \theta_2} \prod_{i=1}^{10} p(x_i, y_i | \theta_1, \theta_2)$
   $= \arg\max_{\theta_1, \theta_2} \left(\prod_{j:y_j=1} \frac{1}{2} e^{-|x_j - \theta_1|} \right) \left(\prod_{j:y_j=2} \frac{1}{2} e^{-|x_j - \theta_2|} \right)$
   $= \arg\max_{\theta_1, \theta_2} e^{-\sum_{j:y_j=1} |x_j - \theta_1| - \sum_{j:y_j=2} |x_j - \theta_2|}$
   Thus, $\hat{\theta}_{i,ML} = \arg\min_{\theta_i} \sum_{j:y_j=i} |x_j - \theta_i| = \arg\min_{\theta_i} \frac{1}{5} \sum_{j:y_j=i} |x_j - \theta_i| = \arg\min_{\theta_i} E[|Z_{emp} - \theta_i|]$
   where $Z_{emp} \sim \text{Uniform}\{x_j, \dots, x_k\}$ i.e. $\hat{\theta}_{i,ML}$ is MMAE estimate of Z_{emp} which is equal to the median value of Z_{emp} distribution ($\theta_{1,ML} = x_3 = -2$, $\theta_{2,ML} = \frac{x_9 + x_{10}}{2} = 0.4$)
   c) Consider the following prior on the pdf on the parameters:
   $\pi(\theta_1, \theta_2) = e^{-2|\theta_1| - 2|\theta_2|} \mathbb{1}_{\{\theta_1 = \theta_2\}}$ MAP (Bayesian) learning principle to estimate θ_1, θ_2 using given data & the error.
   $(\theta_{1,MAP}, \theta_{2,MAP}) = \arg\max_{\theta_1, \theta_2} p(\theta_1, \theta_2 | D)$
   $= \arg\max_{\theta_1, \theta_2} p(D|\theta_1, \theta_2) \cdot \pi(\theta_1, \theta_2)$
   $= \arg\max_{\theta_1, \theta_2} e^{-\sum_{j:y_j=1} |x_j - \theta_1| - \sum_{j:y_j=2} |x_j - \theta_2|} \cdot e^{-2|\theta_1| - 2|\theta_2|} \mathbb{1}_{\{\theta_1 = \theta_2\}}$
   $= \arg\max_{\theta_1, \theta_2} e^{-(10-\theta_1+1+0-\theta_2) + \sum_{j:1<x_j<\theta_1} (x_j - \theta_1) - (1-2-\theta_2+1+2-\theta_2) + \sum_{j:\theta_2<x_j<1} (x_j - \theta_2)}$
   Observe that the effect of the prior is equivalent to doing ML (Principle) learning w/ 2 additional examples at 0 for class 1 and two additional examples at -2 for class 2 & 2 shifts median!
   $\hat{\theta}_{1,MAP} = x_4 = -2.7$, $\hat{\theta}_{2,MAP} = x_7 = -1.3$

Kernel SVM

Replace inner products by a real valued function K of two vector variables (symmetric, pos. def.)
$(w_{SVM}, b_{SVM}) = \arg\min_{(w,b)} \left[\frac{1}{2} w^T w + \sum_{j=1}^{n} \text{hinge}(y_j (w^T x_j + b)) \right]$
$k = x^T x$ Decision: $h_{SVM}(x_{test}) = \text{sign}(a_{SVM}^T (x^T x_{test}) + b_{SVM})$
GOAL: KERNEL!

Soft Margin Observations
1. All outliers are support vectors
2. A support vector need not be an outlier
3. w_{SVM} is unique & a lin. combo of supp. vecs
4. Support vectors need not be unique
5. In the hard margin SVM, for positive class and negative class and equal
6. As in hard margin, at least 1 support vector from each class

MAP Naive Bayes Learning
$p(x_i = c) = \lambda_i$
Need prior on θ_i's
Common: $\pi(\lambda_1, \dots, \lambda_k) = (\lambda_1)^{\alpha_1 - 1} (\lambda_2)^{\alpha_2 - 1} \cdots (\lambda_k)^{\alpha_k - 1}$
$(c(a_1), \dots, c(a_k)) \sim \text{Multinomial}(\lambda_1, \dots, \lambda_k)$
$\pi(\lambda_1, \dots, \lambda_k) \cdot (\lambda_1)^{c(a_1)} \cdots (\lambda_k)^{c(a_k)} \propto (\lambda_1)^{c(a_1) + \alpha_1 - 1} \cdots (\lambda_k)^{c(a_k) + \alpha_k - 1}$
$= \arg\max_{\lambda_1, \dots, \lambda_k} \pi(\lambda_1, \dots, \lambda_k) \cdot \text{const}$
$= \arg\max_{\lambda_1, \dots, \lambda_k} \left[\prod_{i=1}^{k} \lambda_i^{c(a_i) + \alpha_i - 1} \right] \cdot \frac{\text{const}}{\text{const}} = \prod_{i=1}^{k} \lambda_i^{(c(a_i) + \alpha_i - 1)}$

EMPIRICAL Covariance: $\hat{\Sigma} = \frac{1}{n} \sum_{j=1}^{n} (x_j - \bar{x})(x_j - \bar{x})^T$ So, $\lambda_{i,MAP} = \frac{n_i + (\alpha_i - 1)}{n + (\alpha_1 + \dots + \alpha_k) - k}$ NAIVE BAYES

HW Problems

Assignment 6 Problem: Let $D = \{(x_1 = -3, y_1 = -1), (x_2 = 5, y_2 = +1)\}$ be a set of ID (number, label) pairs for training a binary classifier. Let
$p(y=-1 | x, \omega) = \frac{1}{1 + e^{\omega x}}$ be a Logistic deterministic model w/ single scalar parameter ω.
a) Compute $p(y=+1 | x, \omega)$ as a function of ω, $p(y=+1 | x, \omega) = 1 - p(y=-1 | x, \omega) = \frac{e^{\omega x}}{1 + e^{\omega x}} = \frac{1}{1 + e^{-\omega x}}$
b) Let $NLL(\omega) = -\ln(\prod_{j=1}^{2} p(y_j | x_j, \omega))$ be the negative log-likelihood. Compute $\frac{\partial NLL}{\partial \omega}(\omega)$
$NLL(\omega) = \ln(1 + e^{-5\omega}) + \ln(1 + e^{+5\omega}) \Rightarrow \frac{\partial NLL}{\partial \omega} = \frac{-5}{1 + e^{5\omega}} + \frac{+5}{1 + e^{-5\omega}}$
c) w/ initialization $\omega_0 = 0$, compute ω_1, value of ω after one iteration of the gradient descent algorithm for minimizing NLL
$\omega_1 = \omega_0 - \eta \frac{\partial NLL}{\partial \omega}|_{\omega=0} = 0 - \eta(-5 - 5) = 4\eta$
d) Compute ω_{ML}, MLE estimate of ω based on D.
likelihood function of ω is: $\frac{1}{1 + e^{5\omega}} \cdot \frac{1}{1 + e^{-5\omega}}$
(P hp in canonical form)

SVM

This is a strictly increasing function of $w, s.t.$ it is maximized at $\hat{\omega}_{ML} = +\infty$

Assignment 7 Problem: Two-class classification
- 1 points: $x_{\oplus} = (2,0)$, $x_{\ominus} = (0,4)$ +1 points: $x_{\oplus} = (3,3)$ $x_{\ominus} = (7,5)$

a) Given the --- hyperplane $\omega = (3,0)$, $b = -3$, compute parameters in canonical form.
$\min_{x \in \{x_{\oplus}, x_{\ominus}, x_3, x_4\}} \frac{|\omega^T x + b|}{\|\omega\|} = \min\{\frac{|6-3|}{3}, \frac{|0-3|}{3}, \frac{|14-3|}{3}, \frac{|21-3|}{3}\} = 1$
$\omega_{canonical} = \frac{\omega}{3} = (1,0)^T$
$b_{canonical} = \frac{b}{3} = -1$

b) Compute the parameters of the hyperplane passing through $x_{\oplus}$ and $x_{\ominus}$.
Slope: $\frac{(0-4)}{(2-0)} = -2$, y intercept $= 4$, so $x_2 = -2x_1 + 4$, which can be written as $(2,1)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 4 = 0$
$\omega = (2,1)^T$, $b = -4$ (--- hp)

c) Compute the orthogonal projection of $x_{\ominus}$ onto the hyperplane in part b) If $x_{\perp}$ denotes the orthogonal projection, then $(x_{\ominus} - x_{\perp}) \parallel \omega$ for some γ since $\omega \perp$ to hyperplane. So, $x_{\perp} = x_{\ominus} - \gamma\omega$. Since $x_{\perp}$ lies on the hyperplane, it must satisfy $\omega^T x_{\perp} + b = 0$.
$\omega^T x_{\perp} + b = \omega^T x_{\ominus} - \gamma\omega^T\omega + b = 0$
So, $\gamma = \frac{\omega^T x_{\ominus} + b}{\omega^T \omega} = \frac{(2,1)\begin{pmatrix} 3 \\ 2 \end{pmatrix} + (3-1) - 4}{6 - 4 + 1} = \frac{3}{3} = 1$
So, $x_{\perp} = \begin{pmatrix} 3 \\ 2 \end{pmatrix} - 1\begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$
$x_{\perp} = (1,1)^T$

d) Compute the parameters of the maximum margin linear separating hyperplane (SVM hyperplane) in canonical form.
From the figure it is clear that a linearly separating hyperplane exists
Convex hull C_- is the line segment joining $x_{\ominus}$ & x_3
Convex hull C_+ is the line segment joining $x_{\oplus}$ and x_4
Since $x_{\ominus} - x_{\perp} = (2,1)^T = \frac{1}{2}\omega$ i.e. $x_{\perp}$ is $\perp$ to hyperplane (ω, b) and is longer than $x_3 - x_{\perp} = (2,1)^T$, it follows that among all pairs of points, $(x_{\ominus}, x_3)$ is the unique closest pair.
So $\omega_{SVM} = (x_{\ominus} - x_{\perp})^T$, the midpoint is $(x_{\ominus} - x_{\perp})/2 = (2,1)^T$
The SVM hyperplane passes through the midpoint, so
$\omega_{SVM}^T (2,1)^T + b_{SVM} = 0 \Rightarrow b_{SVM} = -\gamma(2,1)(2,1)^T = -6.5\gamma$
Since the hyperplane is in canonical form and $x_{\oplus}$ is a point closest to it,
$\omega_{SVM}^T x_{\oplus} + b_{SVM} = +1 \Rightarrow \gamma \|(2,1)(3,3)^T - 6.5\| = 1 \Rightarrow \gamma = 2/5$, so
$\omega_{SVM} = (4/5, 2/5)^T$, $b_{SVM} = -13/5$

e) Let $\omega_{SVM} = \alpha_{\oplus} x_{\oplus} - \alpha_{\ominus} x_{\ominus} + \alpha_3 x_3 - \alpha_4 x_4$. Hand-compute the α's. Hard margin SVM $\Rightarrow \alpha_j^{SVM} > 0$ for support vectors, $\alpha_j^{SVM} = 0$ otherwise.
$\sum_j \alpha_j^{SVM} y_j = 0$
$\alpha_3 = \alpha_4 = 0$, so: $\begin{pmatrix} 4/5 \\ 2/5 \end{pmatrix} = \alpha_{\oplus}\begin{pmatrix} 2 \\ 1 \end{pmatrix} + (\frac{3}{2})(\alpha_{\oplus} - \alpha_{\ominus})\begin{pmatrix} 2 \\ 1 \end{pmatrix}$
$4/5 = 2\alpha_{\oplus} + 3\alpha_{\oplus} - 3\alpha_{\ominus}$
$2/5 = \alpha_{\oplus} + \frac{3}{2}\alpha_{\oplus} - \frac{3}{2}\alpha_{\ominus}$
$\alpha_{\ominus} = 4\alpha_{\oplus} - 1$
$\alpha_{\ominus} = \frac{7}{3}\alpha_{\oplus} - \frac{4}{3}$
$\alpha_{\oplus} = 7$, $\alpha_{\ominus} = 27$ UNIQUE: one way to express $x_{\oplus}$ as a linear combination of $x_{\ominus}, x_3, x_4$

Assignment 5 Problem 1: Let $Z_1, Z_2 \sim$ IID w/ pmf Uniform$[-0.5, 0.5]$, $X_1 = Z_1$, $X_2 = Z_1 + Z_2$ and $Y = X_1 X_2$
a) Compute $p(y)$, $p(x_1|y)$, $p(x_2|y)$
X_1 deterministic function of Z_1, Y a deterministic function of Z_1. Since Z_1, Z_2 are independent, it follows that X_1 is independent of Y.
So, $p(x_1, 1 | y=-1) = p(x_1 | y=+1) = p(x_1) = \text{Uniform}[-0.5, 0.5]$
Since $Z_2 \sim \text{Uniform}[-0.5, 0.5]$ is symmetric around 0 and $Y = X_1 (X_1 + Z_2)$,
$p(y=-1) = p(y=+1) = 1/2$
Since X_1 independent of Y, $X_2 = X_1 + Z_2$ given $Y=1$, $X_2 = -X_1 + Z_2$ given $Y=-1$.
Since X_1 and $-X_1$ have the same distribution $p(x_2 | y=1) = p(x_2 | y=-1)$
$= p(x_2) = \text{Uniform}[-0.5, 0.5]$, X_2 also independent of Y.
b) Compute Naive Bayes classifier given by Chrome + Tex w/ cost $= \text{FNR} + \text{FPR}$:
$h_{NB}(x_1, x_2) = \arg\max_{y \in \{-1,+1\}} [p(y) \cdot p(x_1|y) \cdot p(x_2|y)]$
Since $p(x_1|y) = p(x_1) = 0.5 \cdot p(x_1) + 0.5 \cdot p(x_1)$, i.e. both classes tied for all values of x_1, x_2, y. Decision rule reduces to a coin flip!
c) Compute misclassification probability of the Naive Bayes classifier in part b) $= 1/2$.
d) Compute MPE classifier given x_1, x_2, and its misclassification probability
Observe that Y is completely determined by X_1 and X_2:
$X_2 = Y \cdot X_1 \to Y = X_2 / X_1 \Rightarrow P(Y \ne X_2 / X_1) = 0$
So, $h(x_1, x_2) = x_2 / x_1$ is the best classifier since misclassification probability can be < 0. It is a Bayes MPE classifier.
X_1, X_2 are individually independent of $Y \to Y$ is dependent on PAIR

Assignment 5 Problem 3: Let $x_1, \dots, x_n$ be a IID training samples from a categorical (multinoulli) distribution, $p(x_i | \theta) = \theta_{x_i}$ where $\theta = (\theta_1, \dots, \theta_k)^T$ an unknown pmf over m possible categories. Define expression for θ_{ML}. For $c \in \{1, \dots, m\}$, let $N_c = \sum_{i=1}^{n} \mathbb{1}(x_i = c)$ denote the # of training samples that correspond to c and $\hat{p}_c = N_c / n$ the corresponding fraction of training samples. Noting $n_c = n \hat{p}_c$,

$\theta_{ML}(x_1, \dots, x_n) = \arg\max_{\theta} p(x_1, \dots, x_n | \theta)$
$= \arg\max_{\theta} \prod_{j=1}^{n} p(x_j | \theta) = \arg\max_{\theta} \prod_{i=1}^{m} \theta_i^{N_i}$
$= \arg\max_{\theta} \sum_{i=1}^{m} N_i \ln \theta_i = \arg\max_{\theta} \sum_{i=1}^{m} \hat{p}_i \ln \theta_i$
$= \arg\min_{\theta} \sum_{i=1}^{m} \hat{p}_i \ln(\frac{1}{\theta_i}) = \arg\min_{\theta} \sum_{i=1}^{m} \hat{p}_i \ln(\frac{\hat{p}_i}{\theta_i})$ not a func of θ
$= \arg\min_{\theta} \left[\sum_{i=1}^{m} \hat{p}_i \ln(\frac{\hat{p}_i}{\theta_i}) + \sum_{i=1}^{m} \hat{p}_i \ln(\frac{1}{\hat{p}_i}) \right]$
$= \arg\min_{\theta} D(\hat{p} \| \theta)$
$= \hat{p}$ = Empirical pmf of categories
$\theta_{ML,i} = \hat{p}_i = \frac{N_i}{n}$

$\log t \le t - 1$ $\forall t \ge 0$
$\sum \hat{p}_i \log \frac{\theta_i}{\hat{p}_i} \le \sum \frac{\hat{p}_i}{\hat{p}_i} - 1$ $\forall i$
$\sum \hat{p}_i (\log \frac{\theta_i}{\hat{p}_i}) \le \sum \theta_i - \sum \hat{p}_i$
$\sum \hat{p}_i \log \frac{\theta_i}{\hat{p}_i} \le 0$
$\sum \hat{p}_i \log \theta_i - \sum \hat{p}_i \log \hat{p}_i \le 0$
$\sum \hat{p}_i \log \theta_i \le \sum \hat{p}_i \log \hat{p}_i$
EQUALITY when $\theta_i = \hat{p}_i$ $\forall i$ ✓

ML Categorical

MAP Naive Bayes

$\text{CCA}(\alpha_1, \dots, \alpha_k) \cdot \text{normalizer} = \int \prod_{i=1}^{k} \lambda_i^{\alpha_i - 1} d\lambda_1 \cdots d\lambda_{k-1} = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{k} \alpha_i)}$
$\pi(\lambda_1, \dots, \lambda_k) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} \lambda_i^{\alpha_i - 1}$
$\text{CCA}(\alpha_1 + c(a_1), \dots, \alpha_k + c(a_k)) \cdot \text{normalizer}$
$= \int \prod_{i=1}^{k} \lambda_i^{\alpha_i + c(a_i) - 1} d\lambda_1 \cdots d\lambda_{k-1} = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i + c(a_i))}{\Gamma(\sum_{i=1}^{k} \alpha_i + n)}$
$\pi(\lambda_1, \dots, \lambda_k) \cdot (\lambda_1)^{c(a_1)} \cdots (\lambda_k)^{c(a_k)} \propto \prod_{i=1}^{k} \lambda_i^{\alpha_i + c(a_i) - 1}$
$\lambda_{i,MAP} = \frac{n_i + (\alpha_i - 1)}{n + (\alpha_1 + \dots + \alpha_k) - k}$

Kernels

- **Kernel Function**: $k(\cdot,\cdot)$ = function of 2 vector variables $k(x,y) \in \mathbb{R}$, $k(\cdot,\cdot): \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, $x, y \in \mathbb{R}^d$
- **Symmetric (kernel)**: $k(x,y) = k(y,x)$ $\forall x, y$
- **Positive definite kernel**: $k(x,y) \geq 0$, $k(x,y) = 0 \Leftrightarrow x = y$ $\forall x_1, \forall x_2, \ldots \forall x_n \in \mathbb{R}^d$, $n \in \mathbb{N}$

$$\begin{bmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) \\ \vdots & & \vdots \\ k(x_n,x_1) & \cdots & k(x_n,x_n) \end{bmatrix}$$

is positive semi-definite (& nonnegative eigenvalues)

- **Kernel examples**:
  1) Polynomial: $(c + x^T y)^k = k(x,y)$, $c > 0$
  2) Radial basis function (RBF): $k(x,y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$
- **Mercer's Theorem**: If $k(\cdot,\cdot)$ is a symm. pos def kernel, then $\exists$ a function $x \to \phi(x)$ from $\mathbb{R}^d$ to a suitable vector space s.t. $k(x,y) = \langle \phi(x), \phi(y) \rangle$ $\forall x, y$

Multiclass vs. Binary Classification

1) **One vs. All (OVA) / One vs. Rest**
Given binary classification algorithm A: $D = \{(x_1, y_1), \ldots (x_n, y_n)\} \to \text{score}(x)$ $y_i \in \{1, \ldots, m\}$
Input: $D = \{x_1, \ldots, x_n\}$, $y_i \in \{1, \ldots, m\}$
Binary algorithm A on classes k to train models for each $k \in Y$
Let $D_k = \{(x_1, (-1)^{(y_1 \neq k)}), \ldots (x_n, (-1)^{(y_n \neq k)})\}$
Let $\text{score}_k(\cdot) = A(D_k)$
end for
$\text{Hova}(x_{\text{test}}) = \arg\max_{k \in Y} (\text{score}_k(x_{\text{test}}))$

2) **One vs. one (OVO)** → decision combining
Given: binary classification algorithm A
$A: \mathcal{D} \to h(\cdot)$ $h(\cdot): \mathcal{X} \to \{+1, -1\}$
Input: $\mathcal{D}$ for all $k \neq y \in Y$ (all classes)
$\mathcal{D}_{(k,l)} = \{(x_i, -1) : y_i = k\} \cup \{(x_i, +1) : y_i = l\}$
$h_{(k,l)} = A(\mathcal{D}_{(k,l)})$
end for
$\text{Hovo}(x_{\text{test}}) = \arg\max_{k \in Y} \left[\sum_{k<l<k} \frac{1 + h_{(k,l)}(x)}{2} + \sum_{l<k} \frac{1 - h_{(l,k)}(x)}{2} \right]$
(1 vote against class k; 1 vote against class l; # wins against classes $k+1, \ldots, m$)

Training complexity = dependent on binary classifier alg
A: $O(n^k)$
OVA: train m classifiers, still n data points, so $O(mn^k)$
OVO: train $\binom{m}{2}$ classifiers (order of m^2), n/m points trained each time, $O(m^2 (\frac{n}{m})^k) = O(m^{2-k} n^k)$
- If $m > n^{k-2}$, OVO better → OVO better when $k > 1$

Regression

$y = R, l(y, \hat{y}) = (y - \hat{y})^2$

$(\hat{w}_{\text{OLS}}, \hat{b}_{\text{OLS}}) = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T y$

Ordinary Least Squares (OLS)
$\mathcal{D} = \{(x_1, y_1), \ldots (x_n, y_n)\}$
$h_{OLS}(x) = w^T x + b = \tilde{w}^T \tilde{x}$
$(\hat{w}_{OLS}, \hat{b}_{OLS}) = \arg\min_{w,b} \sum_{j=1}^n (y_j - w^T x_j + b)^2$
$\hat{w}_{OLS} = (\tilde{X} \tilde{X}^T)^{-1} (\tilde{X} y)$
$\hat{b}_{OLS} = \hat{\mu}_y - (\hat{w}_{OLS})^T \hat{\mu}_X$
Issue if $\tilde{X} \tilde{X}^T$ is not invertible? (Features linearly related, $d > n$)
- USE PSEUDO INVERSE
$A = U \Lambda V$, $U^T U = I$, Λ square, pos def
$A^{-1} = V (\Lambda^{-1}) U^T$ (if $\lambda_i \geq 0$, just use $\frac{1}{\lambda_i}$)
OR
$\hat{X} = \tilde{X} + \lambda I_{d+1}$, $\lambda > 0$ (regularized least squares)

Regularized Least-Squares / Ridge Regression

$(\hat{w}_{\text{ridge}}, \hat{b}_{\text{ridge}}) = \arg\min_{w,b} \left[\sum_{j=1}^n (y_j - w^T x_j - b)^2 + \lambda \|w\|^2 \right]$
Let $b: b(w) = \bar{y}_n - w^T \bar{x}_n$ as in OLS, so:
$\hat{w}_{\text{ridge}} = \arg\min_w \left[\sum_{j=1}^n (y_j - w^T x_j - \bar{y}_n + w^T \bar{x}_n)^2 + \lambda \|w\|^2 \right]$
$\tilde{y}_j = y_j - \bar{y}_n$, $\tilde{x}_j = x_j - \bar{x}_n$
$\hat{w}_{\text{ridge}} = (\tilde{X} \tilde{X}^T + \lambda I_d)^{-1} (\tilde{X} \tilde{Y}^T)$
$X = \frac{1}{\sqrt{n}} [x_1 \cdots x_n]_{d \times n}$, $Y = [y_1 \cdots y_n]_{1 \times n}$
$\tilde{X} = X - [\bar{x}_n \cdots \bar{x}_n]_{d \times n}$, $\tilde{Y} = Y - \bar{y}_n$ (subtract scalar from each y term)

Representer Theorem

$w = \arg\min_w [S_{y_1}(\|w\|^2) + S_{y_2}(w^T x_1, \ldots w^T x_n)$
Then $w \in w = \sum_{j=1}^n \alpha_j x_j = X \alpha \Rightarrow w^T x_j = a^T (X^T X_j)$
$\|w\|^2 = a^T (X^T X) a$

Cool Matrix Identity

$X^T (X X^T + \lambda I_d)^{-1} = (X^T X + \lambda I_n)^{-1} X^T$

Kernel Ridge Regression

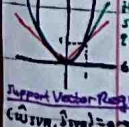
$h(x_{\text{test}}) = Y B (B^T K B + \lambda I_n)^{-1} B^T K (x_{\text{test}}) + \hat{\mu}_y$
where $K_{ij} \in K(x_i, x_j)$, $K(x_{\text{test}}) = \begin{pmatrix} K(x_1, x_{\text{test}}) \\ \vdots \\ K(x_n, x_{\text{test}}) \end{pmatrix}$
$B = (I_n - \frac{1}{n} 1_n 1_n^T)$, 1_n = unit vector of n
RIDGE REGRESSION PREDICTION:
$h_{\text{ridge}}(x_{\text{test}}) = Y B (B^T (X^T X) B + \lambda I_n)^{-1} B^T (X^T x_{\text{test}}) B + \hat{\mu}_y$

Polynomial Kernel

$\text{poly}(x) = b + w_1 x + w_2 x^2 + \ldots w_d x^d$
$= (w_1, w_2, \ldots w_d) \begin{pmatrix} x \\ x^2 \\ \vdots \\ x^d \end{pmatrix} + b$

Robust Linear Regression

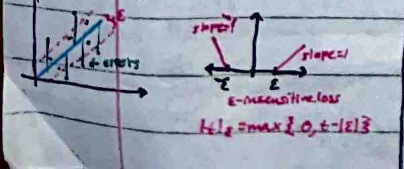
$(\hat{w}_{\text{robust}}, \hat{b}_{\text{robust}}) = \arg\min_{w,b} \left[\sum_{j=1}^n \rho(y_j - w^T x_j - b) + \lambda \|w\|^2 \right]$



* Often use centered data so solution not dependent on coordinate system → "SHIFT INVARIANT"

Support Vector Regression

$(\hat{w}_{SVR}, \hat{b}_{SVR}) = \arg\min_{w,b} \frac{1}{2} \|w\|^2 + \frac{1}{n} \sum_{j=1}^n |y_j - w^T x_j - b|_\varepsilon$
Intuition: if error very small ($< \varepsilon$), do not add to loss



Sparse Linear Regression

- **Goal #1**: Label prediction
- **Goal #2**: Feature selection → make preds w/ fewest features possible

Least-Squares Regression on max limit of # of features used

$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}$ $y = w^T x + b$ # of non-zero components of w is $\|w\|_0$

$y = w^T x + b = w_1 x_1 + w_2 x_2 + \ldots w_d x_d + b$
$\text{cur}_{\text{sparse}, \text{least squares}} = \arg\min_{w,b} \sum_{j=1}^n (y_j - w^T x_j + b)^2$
$\|w\|_0 \leq T$ (easy case → $T = 1, 2$ (small))
NP-hard!
$\|w\|_0 = \sum_{k=1}^d 1(w_k \neq 0)$ = # of non-zero components in w
↳ zero "norm"
$\|w\|_p \triangleq (\sum_{k=1}^d |w_k|^p)^{1/p}$
Meanwhile..... L_1-regularized least-squares
$(\hat{w}_{\text{lasso}}, \hat{b}_{\text{lasso}}) = \arg\min_{w,b} \left[\sum_{j=1}^n (y_j - w^T x_j - b)^2 + \lambda \|w\|_1 \right]$
L_1/taki norm

Tibshirani; Hastie et... Least absolute selection & shrinkage operator → LASSO

Δw_j LASSO
- w_j less significant
- more significant

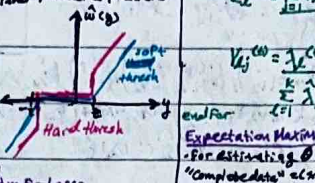
λ

$\sum_{j=1}^n (y_j - w_1 x_{j,1} - w_2 x_{j,2} - w_3 x_{j,3})^2 + \lambda [|w_1| + |w_2| + |w_3|]$

SGD coordinate descent!
optimize 1 coordinate at a time, keeping others fixed (converges always for LASSO!)

Compressive / Compressed Sensing

something about sparsity, small sample, linear projection of product z
δ-function
$\arg\min_w \left[\frac{(y_1 - w)^2}{2} + \tau |w| \right]$
$\arg\min_w \left[(y_1 - w)^2 + 2\tau |w| = \hat{w}(y_1) \right]$
$= \text{sign}(y_1) \cdot \max(0, |y_1| - \tau)$
$= \text{soft-thld}(y_1, \tau)$



ASSUME ALL DATA CENTERED
$(\bar{x}, \bar{y}, \bar{b} = 0)$
$\hat{b}_{\text{lasso}} = \bar{y} - (\hat{w}_{\text{lasso}})^T \bar{x}_n$

Coordinate Descent Algorithm for Lasso

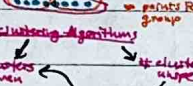
1) Initialization: $w = \hat{w}_{\text{ridge}}(\lambda_1)$
2) Update steps
for $t = 1 \to t_{\max}$ OR convergence
For $k = 1 \ldots d$
$a_k = 2 \sum_{j=1}^n (\tilde{x}_{kj})^2$
$c_k = 2 \sum_{j=1}^n \tilde{x}_{kj} (y_j - w^T x_j + w_k \tilde{x}_{kj})$
$w_k = \text{soft-thld}(\frac{c_k}{a_k}, \frac{\lambda}{a_k})$ ← k updates!
end
end

Iter
cost
$a_k \neq 0$, 0 else

Unsupervised Learning

Given: unlabeled dataset $\mathcal{D} = \{x_1, \ldots x_n\}$
Goal: learn a mapping from $\mathcal{X}$ to some label set $\mathcal{Y}$
$\mathcal{Y}$ discrete set, no intrinsic quantitative value, $|y| = k$ = # of clusters
→ CLUSTERING PROBLEM
→ Group feature vectors in non-overlapping groups s.t.
1) Examples in same group are "similar"
2) Examples in different groups are "dissimilar"

SIMILARITY is non-transitive, i.e. $A \approx B$, $B \approx C$ $\not\Rightarrow A \approx C$



Clustering Algorithms

clusters given → estimate it
clusters unspecified → K-E clustering above jointly

- **Hard Clustering**: Each x belongs to exactly one cluster
- **Soft Clustering**: Every x belongs to all clusters to some degree (probability)
- **Centroid-Based Methods**: K-means & variants (K-means++, K-medians, K-medoids) } k specified
- **Probabilistic Model**: Mixture of Gaussians (GMG), Gaussian mixture model (GMM) } k specified
- **Graph-Partitioning methods**: spectral clustering
- **Hierarchical Clustering**: agglomerative, divisive } k not specified
- **Density-based clustering**: DBScan, mean shift algo } k not specified

K means

Given $x_1, \ldots x_n \in \mathbb{R}^d$, k (# of clusters)
Find centroids $\mu_1, \ldots \mu_k \in \mathbb{R}^d$ and labels $y_1, \ldots y_n \in \{1, \ldots k\}$
To minimize the within-cluster sum of squares (WCSS)
$(y_1, \ldots y_n, \mu_1, \ldots \mu_k) = \arg\min_{\mu_1, \ldots \mu_k \in \mathbb{R}^d} \sum_{j=1}^n \|x_j - \mu_{y_j}\|^2$
$y_1, \ldots y_n \in \{1, \ldots k\}$
WCSS
Can be solved (worst case) complexity $O(n^{dk+1})$
exp hard if k, d given (P time)
algorithm: (initialize centroids first)
Input: $\mathcal{D}$, k
Initialize: $\mu_1^{(0)}, \ldots \mu_k^{(0)} \in \mathbb{R}^d$
For $t = 1, 2, \ldots$ some stop condition
Assign labels: $\forall j$, $y_j^{(t)} = \arg\min_{l \in \{1, \ldots k\}} \|x_j - \mu_l^{(t-1)}\|^2$
Update centroids: $n_l = \sum_{j=1}^n 1(y_j(t) = l)$
$\forall l \in \{1, \ldots k\}$, $\mu_l^{(t)} = \frac{1}{n_l} \sum_{j: y_j = l} x_j$
end for

* Initialization Strategies: *
1) Randomly pick centroids $\mu_1, \ldots \mu_k \overset{iid}{\sim} \mathcal{N}(0, I_d)$
2) K-mean ++
- pick first centroid from $x_1, \ldots x_n$ uniformly randomly ($\mu_1^{(0)}$)
- compute distances of all other points from $\{\mu_1^{(0)}\}$
- pick next centroid μ_2 at random w/ prob $\propto$ distance^2
current centroid set = $\{\mu_1^{(0)}, \ldots \mu_t^{(0)}\} = C_{(t)}$
$\text{prob}(x_j) \propto \min_{\mu \in C} \|x_j - \mu\|^2$
$\mu \in C$
distance $(x_j, C_{(t)})^2$

K medoids

- usually: Centroid of $x_1, \ldots x_n = \mu = \frac{1}{n} \sum_{j=1}^n x_j$ need not be one of $x_1, \ldots x_n$
- IF you want it to be part of the set, called medoid
- medoid of $x_1, \ldots x_n = \arg\min_{x \in \{x_1, \ldots x_n\}} \sum_{j=1}^n \|x_j - a\|^2$ so $\in \{x_1, \ldots x_n\}$
- medoid (general case) $\text{Medoid} = \arg\min_{x \in \{x_1, \ldots x_n\}} \sum_{j=1}^n d(x_j, a)$, $d(\cdot,\cdot)$ general distance function

Soft Clustering

Gaussian Mixture Model
label: $y \in \{1, \ldots k\}$, feature: $x \in \mathbb{R}^d$
$p(x|y) \sim \mathcal{N}(\mu_y, \Sigma_y)$ ← QDA model
$p(y) = \lambda_y$
Parameters: $\theta = (\mu_1, \Sigma_1, \lambda_1), L_1, \ldots k$
- Soft decisions: $\gamma_{ij}^{(t)} = p(y = l | x_j, \theta^t)$

$p(y|x, \theta) = \frac{p(x|y) p(y)}{p(x|\theta)}$
$p(x|y) = \lambda_y \cdot p(x|y)$
$= \lambda_y \mathcal{N}(\mu_y, \Sigma_y)(x) = \frac{\lambda_y e^{-(x-\mu_y)^T \Sigma_y^{-1} (x-\mu_y)}}{(\sqrt{2\pi})^d \sqrt{\det(\Sigma_y)}}$
$p(x) = \sum_{y=1}^k p(x,y) = \sum_{l=1}^k \lambda_l \mathcal{N}(\mu_l, \Sigma_l) x$
$p(y|x, \theta) = \frac{p(x,y,\theta) p(y|\theta)}{p(x|\theta)} = \frac{\lambda_y \mathcal{N}(\mu_y, \Sigma_y)(x)}{\sum_{l=1}^k \lambda_l \mathcal{N}(\mu_l, \Sigma_l)(x)}$

Initialize: $(\hat{\lambda}_l^{(0)}, \hat{\Sigma}_l^{(0)}, \hat{\mu}_l^{(0)})$, $l = 1, \ldots k$
For $t = 1, \ldots$ stop criterion
$\forall l = 1, \ldots k$
$\hat{\lambda}_l^{(t)} = \frac{1}{n} \sum_{j=1}^n \gamma_{lj}^{(t-1)}$
$\hat{\mu}_l^{(t)} = \frac{\sum_{j=1}^n \gamma_{lj}^{(t-1)} x_j}{\sum_{j=1}^n \gamma_{lj}^{(t-1)}}$
$\hat{\Sigma}_l^{(t)} = \frac{\sum_{j=1}^n \gamma_{lj}^{(t-1)} (x_j - \hat{\mu}_l^{(t)})(x_j - \hat{\mu}_l^{(t)})^T}{\sum_{j=1}^n \gamma_{lj}^{(t-1)}}$
$\gamma_{lj}^{(t)} = \frac{\lambda_l^{(t)} \cdot \mathcal{N}(\hat{\mu}_l^{(t)}, \hat{\Sigma}_l^{(t)})(x_j)}{\sum_{l=1}^k \lambda_l^{(t)} \mathcal{N}(\hat{\mu}_l^{(t)}, \hat{\Sigma}_l^{(t)})(x_j)}$
end for

Expectation Maximization Algorithm

- For estimating θ from $\mathcal{D} = \{x_1, \ldots x_n\}$
"Complete data" = $(x_1, y_1), (x_n, y_n) \to (X, Y)$
"Incomplete data" = $x_1, \ldots x_n \to X$
If complete data available, estimate θ by:
$\hat{\theta}_{ML} = \arg\max_\theta L_{\log} p(X, Y | \theta)$
Y not observed, so reasonable approach: average out cost w/ respect to random variable Y and then maximize likelihood: given θ, X
Replace: $L_{\log} p(X, Y|\theta)$ to be estimated
$p(y|x, \theta) \to p(Y|X, \theta) = \frac{p(X,Y|\theta)}{p(X|\theta)}$ ← w.r.t. $p(Y|X, \theta^{(t-1)})$
$= E[L_{\log} p(X, Y|\theta) | X, \theta] = Q(\theta, \theta^{(t-1)})$ ← $\psi_{X,Y}$
Initialize $\hat{\theta}^{(0)}$
For $t = 1, \ldots$ convergence
E step: $Q(\theta, \theta^{(t-1)}) = E[L_{\log} p(X, Y|\theta) | X, \theta^{(t-1)}]$
M step: $\theta^{(t)} = \arg\max_\theta Q(\theta, \theta^{(t-1)})$
end for

Theorem: $p(X|\theta^{(t)}) \leq p(X|\theta^{(t+1)}) \leq p(X|\theta^{(t)})$

When applied to GMM model for clustering
Set GMM algorithm!
In GMM alg:
1) Set $\lambda_l^{(t)} = 1/n$ $\forall l$ and all t
2) Set $\Sigma_l^{(t)} = \sigma^2 I_d$ $\forall l, t$
3) Set $\sigma^2 \to 0$
then $p(y_j | x_j, \theta) \to \begin{cases} 1, & \text{if cluster} \\ 0, & \text{other clusters} \end{cases}$
and GMM = K-means (closest to x_j)

SPECTRAL CLUSTERING

- emphasis: put observations/samples in different clusters (as opposed to K-means, which keeps similar items together)
- examples, $x_1, \ldots x_n \in \mathbb{R}^d$ "nodes" in a graph
- Define similarity function
$w_{ij} = d(x_i, x_j)$
e.g. $e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$
Degree (node i) = $\sum_{j=1}^n w_{ij}$ (assume weight function symmetric)
W = weighted adjacency matrix $= [-w_{ij}-]$
D = degree matrix $= \begin{bmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{bmatrix}$
$d_i = \sum_{j=1}^n w_{ij} = (W 1_n)_i$

Graph Laplacian L
$L = D - W$
$L_{sym} = D^{-1/2} L D^{-1/2}$
$L_{RW} = I - D^{-1} W$

Theorem: $\sum_{i,j=1}^n w_{ij} (f_i - f_j)^2 = 2 f^T L f$ for any $f \in \mathbb{R}^n$
- $L = L^T$ (symmetric)
- $L \geq 0$ → non-neg eigenvals
- IF graph has k connected components $A_1, \ldots A_k$, the eigenspace of eigenvalue 0 is spanned by the indicator functions of $A_1, \ldots A_k$

Algorithm
1) Compute graph laplacian $L_{sym} = \ldots$
2) Compute "first" k eigenvectors $v_1, \ldots v_k$ of L (= smallest eigenvals)
3) Let $V = [v_1 \, v_2 \ldots v_k]_{n \times k}$
4) Cluster rows of V using k-means

Rayleigh-Ritz Theorem
Let A be symmetric, then columns of P are given by orthonormal eigenvecs of L corresponding to k smallest eigs.

Original Ratio-Cut Minimization
$\text{argmin}_{A \subset V} (P | L | P^T)$ (P of Perron-Frobenius)
Relaxed Ratio-Cut minimization
$\text{argmin}_{F \in \mathbb{R}^{n \times k}} (F^T L F)$
s.t. $F^T F = I_k$

$f_j = \frac{1_{A_j}}{\sqrt{|A_j|}} = \frac{1}{\sqrt{|A_j|}} \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$
$\text{Ratio-Cut}(A_1, \ldots A_k) = \sum_{i=1}^k \frac{\text{cut}(A_i, A_i^c)}{|A_i|}$
$\text{cut}(A, B) = \sum_{i \in A} \sum_{j \in B} w_{ij}$
$F_{n \times k} = \begin{bmatrix} \frac{1}{\sqrt{|A_1|}} f_{1,1} & \cdots & \frac{1}{\sqrt{|A_k|}} f_{1,k} \\ \vdots & & \vdots \\ \frac{1}{\sqrt{|A_1|}} f_{n,1} & \cdots & \frac{1}{\sqrt{|A_k|}} f_{n,k} \end{bmatrix}$
$F^T F = I_k$

Kernel K-means

$\Sigma = (x_1 \ldots x_n)_{d \times n}$ $\forall j = 1 \ldots n$: $x_j = \begin{pmatrix} x_{j,1} \\ \vdots \\ x_{j,d} \end{pmatrix} = \Sigma e_j$
$\|x\|^2 = x^T x$
for dec. k, λ_l can be assumed to belong to the span of $x_1, \ldots x_n$
Why? Any component of μ_l which is $\perp \text{span}(x_1, \ldots x_n)$ would only increase $\|x_j - \mu_l\|^2$

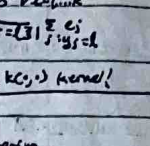
Express $\mu_l = \Sigma u_l$, $u_l \in \mathbb{R}^n$
$\|x_j - \mu_l\|^2 = \|\Sigma e_j - \Sigma u_l\|^2 = \|\Sigma(e_j - u_l)\|^2 = (e_j - u_l)^T \Sigma^T \Sigma (e_j - u_l)$
$k = \Sigma^T \Sigma$

algorithm
Init $\mu_1, \ldots \mu_k \in \mathbb{R}^n$
for $t = 1 \ldots$ stop
update labels: $\forall j = 1 \ldots n$
$y_j = \arg\min_{l=1 \ldots k} (e_j - u_l)^T k (e_j - u_l)$
update centroids: $\forall l = 1 \ldots k$
$u_l = \frac{1}{|\{j : y_j = l\}|} \sum_{j: y_j = l} e_j$
end for
$k(x_i, x_j) = k(x_i, x_j)$, $k(\cdot,\cdot)$ kernel!

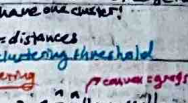
Hierarchical Clustering

→ recursive cluster points
→ creates "nested" clustering
1) Agglomerative (bottom up)
2) Divisive (top-down)

Distance b/w clusters
- Single linkage: $d_{SL}(A, B) = \min_{x \in A, x' \in B} d(x, x')$
- Complete linkage: $d_{CL}(A, B) = \max_{x \in A, x' \in B} d(x, x')$
- Average linkage: $d_{AVG}(A, B) = \frac{\sum_{x \in A} \sum_{x' \in B} d(x, x')}{|A| \cdot |B|}$



Agglomerative Clustering
1) Initialize clusters to $\{x_1\}, \{x_2\}, \ldots \{x_n\}$
2) Find two most similar clusters according to some notion of distance b/w clusters & merge them
3) Repeat until you have one cluster!



Sum of norms clustering
- k not specified
$\arg\min \sum_{j=1}^n \|x_j - \mu_{y_j}\|_p + \lambda \sum_{l=1}^k \|\mu_l\|_p$
$\mu_1 \ldots \mu_k \in \mathbb{R}^n$ → get n centroids

Dimensionality Reduction

- Like compression
$\mathbb{R}^d \to \mathbb{R}^k$, $k < d$

Motivation:
- reduced storage & complexity of algs
- visualization
- noise removal

k-means-based compression
$x \to y = \arg\min_{1 \leq j \leq k} \|x - \mu_j\|^2$
where $\mu_1, \ldots \mu_k$ found via clustering
use only $\log k$ bits to represent each x!

Euclidean space "stashing"
$x \in \mathbb{R}^d$, $y \in \mathbb{R}^k$ $F = d \times k$ mat
$y = F^T x$

Johnson-Lindenstrauss Lemma
Let $f_{d,k}$ be a random matrix w/ entries $\sim \mathcal{N}(0, 1/k)$, let $\varepsilon, \beta \in (0, 1)$ and n a positive integer
If $k \geq \frac{6}{\varepsilon^2} \log(\frac{n^2}{\beta})$
then $\forall \varepsilon, \beta, n$, datasets $\mathcal{D} \in \mathbb{R}^d$ w/ $|\mathcal{D}| = n$
$P(\forall x_i, x_j \in \mathcal{D}, (1 - \varepsilon)\|x_i - x_j\|^2 \leq \|F x_i^T - F x_j^T\|^2 \leq (1 + \varepsilon)\|x_i - x_j\|^2) \geq 1 - \beta$
$X \to Y$ DISTANCES PRESERVED

Let G be an undir. graph w/ nonneg weights. The multiplicity k of the eigenval 0 of L = # of connected components $A_1, \ldots A_k$ in the graph.
Eigenspace of eigenval 0 is spanned by the indicator vectors $1_{A_1}, \ldots 1_{A_k}$

eigenspace = nullspace of $A - \lambda I_n$
all vectors B s.t. $AB = 0$, $B \neq 0$

$1_{A_1} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ $1_{A_2} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$

typical vec in eigenspace:
$a 1_{A_1} + b 1_{A_2} + c 1_{A_3}$

PCA

Given $\mathbb{X} = (x_1^1 \cdots x_n^1)$, $k \le \text{rank}(\mathbb{X})$

Eigenvector Decomposition of Empirical Correlation Matrix:

$\mathbb{X}\mathbb{X}^T = U \Lambda U^T$ $U = (u_1 \cdots u_d)$, $u^{-1} = u^T$

$\Lambda = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_d \end{pmatrix}$, $\lambda_1 \ge \lambda_2 \ldots \ge \lambda_d > 0$

Best k-dimensional Subspace: $V_k = \text{span}(u_1, \ldots u_k)$

$F = (u_1 \cdots u_k)$ $F^T F = I_k$

u_1 ... u_k

Encode/Embed Test Sample:

$x \in \mathbb{R}^d \to y = F^T x \in \mathbb{R}^k$

Reconstruct/Decode: $y \to \hat{x} = Fy = FF^T x \in \mathbb{R}^d$

$= \text{Proj}_{V_k}(x)$

Approximation Error:

$\|x - \hat{x}\|^2 = \|x\|^2 - \|\hat{x}\|^2$

$= \|x\|^2 - \|Fx\|^2$

$= \|x\|^2 - \|y\|^2$

Total Approx Error: $\sum_{j=1}^n \|x_j - \hat{x}_j\|^2 = \sum_{i=k+1}^d \lambda_i$

Dual/kernel PCA

Given $\mathbb{X}$, k (or κ)

Gram/kernel Matrix: $K = \mathbb{X}^T\mathbb{X} = V \Lambda V^T$, $V^{-1} = V^T$

Eigen Decomp: $\Lambda = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}$, $\lambda_1 \ge \ldots \lambda_n > 0$

Best k-dim subspace:

$F = \mathbb{X}\left[\frac{1}{\lambda_1} u_1 \cdots \frac{1}{\lambda_k} u_k\right] \Lambda_k^{-1/2}$

$\Lambda_k = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_k \end{pmatrix}$

$F^T F = I_k$, V_k best k span $\{f_1 \ldots f_k\}$

Embed / Encode:

$x \in \mathbb{R}^d \to y = F^T x = \Lambda_k^{-1/2} \begin{pmatrix} -v_1^T- \\ \vdots \\ -v_k^T- \end{pmatrix} (\mathbb{X}^T x)$ or $\left(k(x_i, x)\right)_{i=1}^n$ or $(k(x_i, x))$

Reconstruction: only if kernel feature mapping explicitly known

$\text{proj}_{V_k}(x) = \hat{x} = Fy = FF^T x$

Approx Error:

$\|x - \hat{x}\|^2 = \|x\|^2 - \|\hat{x}\|^2$

$= \|x\|^2 - \|F^T x\|^2$

$= \|x\|^2 - \|y\|^2$

Total Approx: $\sum_{j=1}^n \|x_j - \hat{x}_j\|^2 = \sum_{i=k+1}^n \lambda_i$

Encoding entire Data Set: $F^T\mathbb{X} = \Lambda_k^{1/2} V_k^T$

HW #8 Clustering

8.1 a) Let $X \sim \text{Unif}([-\pi/2, \pi/2])$ and $Y = \sin(X)$. Compute $(\omega_{\text{LINMSE}}, b_{\text{LINMSE}}) = \arg\min_{w,b} E[(Y - wX - b)^2]$.

For each value of w, value of b which minimizes $E[(Y-wX-b)^2]$ is given by $E[(Y-wX)] = \mu_Y - w\mu_X$. Now, $\mu_X = 0$ since X symmetrically distributed about 0, $E[\sin(X)] = \mu_Y = 0$ since X symmetrically distributed around 0 and $\sin(x)$ an antisymmetric function. So, $b_{\text{LINMSE}} = 0$

Best w obtained by orthogonality principle $E[(Y - \omega_{\text{LINMSE}} X - 0)X] = 0$.

This gives $\omega_{\text{LINMSE}} = \Sigma_X^{-1}\Sigma_{XY}$. $\Sigma_X = \text{Var}(X) = (\frac{\pi}{2} - (-\frac{\pi}{2}))^2 = \frac{\pi^2}{12}$

Cross variance $\text{Var}(X, Y) = E[(X - 0)(Y - 0)] = E[X \sin(X)] = \int_{-\pi/2}^{\pi/2} x \sin x \frac{1}{\pi} dx = \frac{2}{\pi}$

So, $\omega_{\text{LINMSE}} = \frac{12}{\pi^2} \times \frac{2}{\pi} = \frac{24}{\pi^3}$

b) Consider the training set $D = \{(x_1, y_1) = (-1, 1), (x_2, y_2) = (-\frac{1}{2}, -\frac{1}{2}), (x_3, y_3) = (0, 0), (x_4, y_4) = (\frac{1}{2}, \frac{1}{2}), (x_5, y_5) = (1, 1)\}$.

i) Compute $(\omega_{OLS}, b_{OLS}) = \arg\min_{w,b} \sum_{j=1}^5 (y_j - wx_j - b)^2$

$\hat{\omega}_{OLS} = \frac{\hat{\Sigma}_{XY}}{\hat{\Sigma}_X}$, $\hat{b}_{OLS} = \hat{\mu}_Y - \hat{\omega}_{OLS} \cdot \hat{\mu}_X$

$\hat{\mu}_X = \hat{\mu}_Y = 0$, so $b_{OLS} = 0$

$\hat{\Sigma}_X = \frac{1}{5}\sum_{j=1}^5 (x_j - \hat{\mu}_X)^2 = 1/2$

$\hat{\Sigma}_{XY} = \frac{1}{5}\sum_{j=1}^5 (x_j - \hat{\mu}_X)(y_j - \hat{\mu}_Y) = 17/40$

So, $\hat{\omega}_{OLS} = 2 \times 17/40 = 17/20$

ii) Compute $(\omega_{PLS}, b_{PLS}) = \arg\min_{w,b} \sum_{j=1}^5 (y_j - w^T\phi_j - b)^2$,

$\phi_j = (x_j, x_j^2)^T$, $w = (\omega_1, \omega_2)^T$

For $j = 1, \ldots 5$, $\phi_j = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -\frac{1}{2} \\ \frac{1}{4} \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{1}{2} \\ \frac{1}{4} \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$\hat{\mu}_\phi = \begin{pmatrix} \frac{1}{5}(-1 + -\frac{1}{2} + 0 + \frac{1}{2} + 1) \\ \frac{1}{5}(1 + \frac{1}{4} + 0 + \frac{1}{4} + 1) \end{pmatrix} = \begin{pmatrix} 0 \\ 1/2 \end{pmatrix}$

For $j = 1, \ldots 5$, $\phi_j - \hat{\mu}_\phi = \begin{pmatrix} -1 \\ 1/2 \end{pmatrix}, \begin{pmatrix} -1/2 \\ -1/4 \end{pmatrix}, \begin{pmatrix} 0 \\ -1/4 \end{pmatrix}, \begin{pmatrix} 1/2 \\ -1/4 \end{pmatrix}, \begin{pmatrix} 1 \\ 1/2 \end{pmatrix}$

$\hat{\Sigma}_\phi = \frac{1}{5}\sum_{j=1}^5 \phi_j \phi_j^T = \frac{1}{5}\left\{\begin{pmatrix} -1 \\ 1/2 \end{pmatrix}(-1 \; 1/2) + \begin{pmatrix} -1/2 \\ -1/4 \end{pmatrix}(-1/2 \; -1/4) + \begin{pmatrix} 0 \\ -1/4 \end{pmatrix}(0 \; -1/4) + \begin{pmatrix} 1/2 \\ -1/4 \end{pmatrix}(1/2 \; -1/4) + \begin{pmatrix} 1 \\ 1/2 \end{pmatrix}(1 \; 1/2)\right\}$

$= \begin{pmatrix} 13/32 & 0 \\ 0 & 7/40 \end{pmatrix}$

$\hat{\Sigma}_{\phi y} = \frac{1}{5}\sum_{j=1}^5 \phi_j (y_j - \hat{\mu}_y) = \frac{1}{5}\left\{\begin{pmatrix} -1 \\ 1/2 \end{pmatrix}(-1) + \begin{pmatrix} -1/2 \\ -1/4 \end{pmatrix}(-1/2) + \begin{pmatrix} 0 \\ -1/4 \end{pmatrix}(0) + \begin{pmatrix} 1/2 \\ -1/4 \end{pmatrix}(1/2) + \begin{pmatrix} 1 \\ 1/2 \end{pmatrix}(1)\right\}$

$= \begin{pmatrix} 13/32 \\ 32/13 \end{pmatrix}$

So, $\hat{\omega}_{PLS} = \hat{\Sigma}_\phi^{-1}\hat{\Sigma}_{\phi y} = \begin{pmatrix} 32/13 & 0 \\ 0 & 40/7 \end{pmatrix}\begin{pmatrix} 13/32 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$b_{PLS} = \hat{\mu}_y - \hat{\omega}_{PLS}^T \hat{\mu}_\phi = 0$

So, $\hat{y}_{PLS} = \hat{\omega}_{PLS}^T \begin{pmatrix} x \\ x^2 \end{pmatrix} + b_{PLS} = x$

c) Let $X = Y + Z$ where $Y \perp Z$, $Z \sim \mathcal{N}(0, \sigma^2)$, $\sigma > 0$, and $Y \sim p(y) = \frac{1}{2}\lambda e^{-\lambda|y|}$, $\lambda > 0$.

Derive $h_{MAP}(x)$, MAP estimate of Y given $X = x$

$p(x, y) = \frac{1}{2}e^{-\lambda|y|}\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(x-y)^2}$

$h_{MAP}(x) = \arg\min_y [-\ln p(x, y)]$

$= \arg\min_y [(x-y)^2 + 2\lambda\sigma^2|y|]$

$\psi(y)$ is a continuous function that is differentiable everywhere except $y = 0$.

$\psi'(y) = \frac{d}{dy}\psi(y) = \begin{cases} 2[(y-x) + \lambda\sigma^2 \text{sign}(y)] & y \ne 0 \\ \text{not diff} & y = 0 \end{cases}$

$= \begin{cases} 2(y - x - \lambda\sigma^2) \ldots y < 0 \\ 2(y - x + \lambda\sigma^2) \ldots y > 0 \\ \text{not diffable} \quad y = 0 \end{cases}$

Case 1: $|x| \le \lambda\sigma^2$

$\psi'(y) = \begin{cases} < 0, & y < 0 \\ > 0, & 0 < y \\ \text{not diff } y = 0 \end{cases}$ So, ψ strictly decreasing when $y < 0$, increasing when $y > 0$

So, ψ minimized @ $y = 0$

$h_{MAP}(x) = 0$ if $|x| \le \lambda\sigma^2$

Case 2: $x > \lambda\sigma^2$

$\psi'(y) = \begin{cases} < 0 & y < 0 \\ \text{not diff } y = 0 \\ < 0 & 0 < y < x - \lambda\sigma^2 \\ \ge 0 & x - \lambda\sigma^2 \le y \end{cases}$

So, ψ decreasing for $y < x - \lambda\sigma^2$, increasing for $x - \lambda\sigma^2 \le y$

ψ minimized at $y = x - \lambda\sigma^2$

$h_{MAP}(x) = x - \lambda\sigma^2$ if $x > \lambda\sigma^2$

Case 3: $x < -\lambda\sigma^2$

$\psi'(y) = \begin{cases} < 0 & y \le x + \lambda\sigma^2 \\ > 0 & x + \lambda\sigma^2 < y < 0 \\ \text{not diff} & y = 0 \\ > 0 & x - \lambda\sigma^2 < y \end{cases}$

So, ψ decreasing for $y < x + \lambda\sigma^2$ and increasing for $x + \lambda\sigma^2 \le y$. ψ therefore minimized at $y = x + \lambda\sigma^2$

$h_{MAP}(x) = x + \lambda\sigma^2$ if $x < -\lambda\sigma^2$

So, $h_{MAP}(x) = \begin{cases} 0, & |x| \le \lambda\sigma^2 \\ x - \lambda\sigma^2, & \lambda\sigma^2 < x \\ x + \lambda\sigma^2, & x < -\lambda\sigma^2 \end{cases}$

$= \begin{cases} 0 & |x| \le \lambda\sigma^2 \\ x - \text{sign}(x)\lambda\sigma^2, & |x| \ge \lambda\sigma^2 \end{cases}$

$h_{MAP}(x) = \text{sign}(x) \cdot \max(0, |x| - \lambda\sigma^2)$

HW #9 Clustering

9.1 Consider a 1-dimensional Dataset of $n = 2m+1$ points ($m \ge 1$) spread on the real line with one point at 0, m points at $a > 0$, and m points at $b > a$, where $0 < a < \frac{1}{2}a < b$. Now suppose we run 2-means on the dataset w/ $k = 2$ and initial centroids chosen over the range $[0, a+b]$.

a) Determine an initialization that will cause the 2-means algorithm could converge to

Clustering 1: $\{0, a, a \ldots a\}$, $\{(a+b), \ldots (a+b)\}$

Clustering 2: $\{0\}$, $\{a \ldots a, (a+b) \ldots (a+b)\}$

b) For each solution, provide expressions for centroids and WCSS in terms of m, a, b.

Clustering 1: $\mu_1 = \frac{am}{m+1} = 0 \cdot \frac{1}{m+1} + a\frac{m}{m+1}$

$\mu_2 = a+b$

$WCSS_1 = 1 \cdot (0 - \mu_1)^2 + m(a - \mu_1)^2 + m(a+b - \mu_2)^2$

$= \frac{a^2 m^2}{(m+1)^2} + \frac{ma^2}{(m+1)^2} + 0 = \frac{ma^2}{m+1}$

Clustering 2: $\mu_1 = 0$, $\mu_2 = \frac{a+b}{2}$

$WCSS_2 = 1 \cdot (0 - \mu_1)^2 + m(a - \mu_2)^2 + m(a+b - \mu_2)^2$

$= 0 + \frac{m}{4}b^2 + \frac{m}{4}b^2 = \frac{mb^2}{2}$

c) Compute the ratio of the largest and smallest WCSS as a function of m, a, b and comment on the implications. Clustering 1 is WCSS$_1 \le$ WCSS$_2$:

$WCSS_1 = \frac{ma^2}{m+1} \le m\frac{a^2}{2} \le \frac{mb^2}{2} = WCSS_2$

Ratio $\frac{WCSS_2}{WCSS_1} = \frac{(m+1)}{2}\frac{b^2}{a^2}$ this ratio can be made arbitrarily large for suitably large m.

So, $\exists$ a dataset for which 2-means can converge to a solution whose WCSS is arbitrarily higher than the best solution.

Takeaway: necessary to run k-means w/ multiple initializations of centroids & choose solution resulting in the smallest WCSS.

d) Suppose two initial centroids are chosen independently & uniformly at random over $[0, a+b]$. Compute the probability of converging to each possible solution.

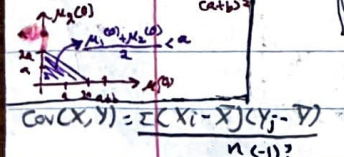
Solution is determined by whether the mid-point of the initial centroids lies to the left or right of a.

$P(\text{Cluster 2}) = P\left(\frac{\mu_1^{(0)} + \mu_2^{(0)}}{2} < a\right)$

where $\mu_1^{(0)}, \mu_2^{(0)} \sim \text{Unif}[0, a+b]$

$\frac{1}{2} \times 2a \times 2a \times \frac{1}{(a+b)^2} = \frac{2a^2}{(a+b)^2} < \frac{1}{2}$

So, $P(\text{Cluster 1}) = 1 - \frac{2a^2}{(a+b)^2}$



$\text{Cov}(X, Y) = \frac{\sum (x_i - \bar{X})(y_j - \bar{Y})}{n(-1)}$?

Distance b/w a point & a line:

$\text{distance}(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$

e.g. Find eigenvalues & eigenvectors of a 2x2 matrix

$A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$

$|A - \lambda \cdot I| = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = 0$

$\left|\begin{bmatrix} -\lambda & 1 \\ -2 & -3-\lambda \end{bmatrix}\right| = \lambda^2 + 3\lambda + 2 = 0 \to \lambda_1 = -1, \lambda_2 = -2$

$A \cdot v_1 = \lambda_1 \cdot v_1$

$(A - I\lambda_1) \cdot v_1 = 0$

$\begin{bmatrix} -\lambda_1 & 1 \\ -2 & -3-\lambda_1 \end{bmatrix} v_1 = 0$

$\begin{bmatrix} 1 & 1 \\ -2 & -2 \end{bmatrix}\begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = 0 \to$ (top row) $v_{11} + v_{12} = 0$, $v_{11} = -v_{12} \to \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

(same procedure for second eigenval)

proportion of variance for λ_r: $\frac{\lambda_r}{\sum \lambda_s}$ ← eigenval

PCA

1) Subtract mean from each data dimension (get $\bar{X}$) (Center data to get $\hat{X}$)

2) Calculate covariance matrix

3) Calculate eigenvals/vecs

4) Eigenvec w/ highest eigenval is the PC of the data set

5) Final data = (eig1 eig2 ...)^T $(x_1^1 \; x_{2n})$

6) Back to orig: $\text{orig}_1 = (\text{eig}_1^1 \; \text{eig}_2^1 \ldots)$(Final data)

d = # of features

n = # samples

$\mathbb{X}, \mathbb{X}_i \to \hat{Y} \to$ new transformed matrix

P = transformation mat

$V = (\text{eig}_1^1 \; \text{eig}_2^1)$

$P\mathbb{X} = Y \to$ goal

$d \times n \quad d \times n$

$S_X = \frac{1}{n-1}\hat{\mathbb{X}}\hat{\mathbb{X}}^T = \text{covmat}$ Choose $P = V^T$ (eig 1 ... / eig 2 ...)

Final Data: data items in columns, dims on rows

USE UNIT EIGENVECS

Orig data = (eig eig ...) * (Final data) + (orig means)

	1	2	3	4	5	6
x	10	11	8	3	2	1
y	6	4	5	3	2.8	1

avg measure for group 1 : $\frac{35}{6}$

group 2 :

"fit" = project data onto it, minimize distances

Purity

n_{ij} = # of points in cluster i belonging to class j

$n_i = \sum_{j=1}^k n_{ij}$ = total # of points in cluster i

purity of a cluster = $p_i = \frac{\max_j n_{ij}}{n_i}$

purity of clustering: $p = \sum_{i=1}^k \frac{n_i}{n} p_i$

$= \frac{1}{n}\sum_{i=1}^k \max_j n_{ij}$