

Classification

k-Nearest Neighbors

kNNs are bounded by $\leq 2x$ the Bayes optimal error, $N, k \rightarrow \infty, k/N \rightarrow 0$.	
Edge Case	2 pts w/ same features but diff classes.
Robustness	Generalizes better to test data.
Fit	Better training classification.
Validation	Hold back data subset as validation set. Train multiple times w/ diff hyper-params. Choose what is best on validation set.
Training Set	Used to learn model weights.
Validation Set	Tunes hyperparameters (ex. $k \in kNN$).
Test Set	used as FINAL evaluation of model.
Isocontour of f	$L_c = \{x \mid f(x) = c\}$, with isovalue c .
Isotropic Gaussian	Same var in ea dir: $\Sigma = cI$.
Anisotropic Gaussian	Allows diff amnts of var along diff dirs, $\Sigma \succ 0$.

Perceptron

Model/rule: 1 if $\vec{X}_i \cdot \vec{w} \geq 0$ elif $\vec{X}_i \cdot \vec{w} \leq 0 \implies -1$.
 Loss: $L(z, y_i) = 0$ if $y_i z \geq 0$ else $-y_i z$, (z =pred, y_i =true ans).

$$R(w) = \sum_{i=1}^n L(X_i \cdot w, y_i) = \sum_{i \in V} -y_i X_i \cdot w$$

Gives some linear boundary; if data is linearly separable, correctly classifies all data in at most $O\left(\frac{r^2}{\gamma^2}\right)$ iterations.

Support Vector Machines

Hard-Margin: $\min_{\vec{w}, b} \|\vec{w}\|_2^2$, s.t. $y_i(\vec{w}^\top \vec{x}_i - b) \geq 1 \forall i$

Fails w/ non-linearly sep. data. Margin size = $\frac{1}{\|\vec{w}\|}$, Slab size = $\frac{2}{\|\vec{w}\|}$

Hyperplane	$H = \{x : w \cdot x = -\alpha\}$ flat, infinite, $\dim(d-1)$ plane $\vec{w} \cdot (y-x) = 0, \vec{w}$ is normal vec of H .
$x, y \in H$	
Support Vectors	Examples needed to find $f(\mathbf{x}) \in SVM$. Examples with non-0 weight $\alpha_k \in SVM$.

Soft-Margin

Allows misclassifications: $\min_{\vec{w}, b, \xi_i} \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^n \xi_i$ s.t.

$$y_i(\vec{w}^\top \vec{x}_i - b) \geq 1 - \xi_i, \quad \forall i; \quad \xi_i \geq 0, \quad \forall i$$

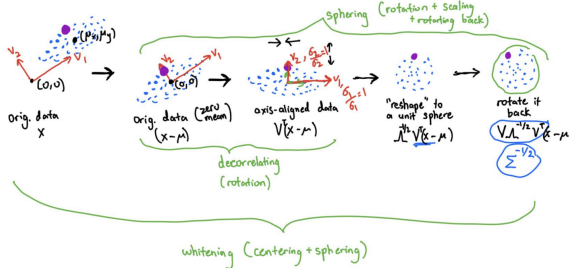
Small C: maximize margin, underfitting, less sensitive, more flat.
 Big C: minimize margin, overfitting, very sensitive, more sinuous.
 $C \rightarrow \infty \implies$ Soft-Margin \rightarrow Hard-Margin. Note $C \geq 0$.

Generative

Want to learn **everything** about data before you classify:
 the **priors** $\hat{\pi}_i = \Pr(Y = C_i)$ and **cond. dist** $\mathbb{P}(X|Y = C_i)$.
Posterior: $\mathbb{P}(Y = C_i|X) = \frac{\mathbb{P}(X|Y=C_i) \cdot \mathbb{P}(Y=C_i)}{\mathbb{P}(X)}$

Logistic: $\frac{1}{1+e^{-h(x)}}$, where $h(x)$ is **linear** in terms of features.
 Function: True in LDA but not QDA (where $h(x)$ is quadratic).
 GDA: Assumes each class models a Gaussian distribution.
 $Q_C(x) = -\frac{\|x - \mu_C\|^2}{2\sigma_C^2} - d \ln \sigma_C + \ln \pi_C =$
 $\ln(\sqrt{2\pi}^d f_C(x) \pi_C) = -\frac{1}{2}(x - \mu_C)^\top \Sigma_C^{-1}(x - \mu_C)$
 QDA: Works with any number of classes; $\frac{d(d+3)}{2} + 1$ params.
 LDA: when variances are equal; $d + 1$ params.

Isotropic:
 QDA: $\hat{\sigma}^2 = \frac{1}{dn} \sum_{i: y_i=C} \|x_i - \hat{\mu}_c\|^2$
 LDA: $\hat{\sigma}^2 = \frac{1}{dn} \sum_C \sum_{i: y_i=C} \|x_i - \hat{\mu}_c\|^2$
 Anisotropic:
 QDA: $\hat{\Sigma}_c = \frac{1}{nc} \sum_{i: y_i=C} (X_i - \hat{\mu}_c)(X_i - \hat{\mu}_c)^\top$
 LDA: $\hat{\Sigma} = \frac{1}{n} \sum_C \sum_{i: y_i=C} (X_i - \hat{\mu}_c)(X_i - \hat{\mu}_c)^\top$



Discriminative

Want to learn a *few* things before trying to classify.
 Only tries to model $\mathbb{P}(Y|X)$ from training data.
Logistic Reg (2 classes): For a training point, $P(Y = y_i | x) = p^{y_i} (1-p)^{1-y_i}$. Note that $p = s(w^T x)$ as given by our model of the posterior $P(Y = 1 | x)$. MLE on this leads to the cross entropy loss function (which is convex!), namely

$$L(w) = - \sum y_i (\ln p_i + (1 - y_i) \ln (1 - p_i))$$

Note: $P(Y = 1 | x) = \frac{1}{1 + \exp(-w^T x)}$; $s'(\gamma) = s(\gamma)(1 - s(\gamma))$

Decision Boundary: of the form $w^T x > c_1$ thus must be linear.
 Though probability predictions are non-linear, actual boundary is linear. Log Reg always separates linearly separable points.
Softmax Reg: logistic regression for multiple classes

Probability

Multivariate Gaussian PDF:
 $f_{\mathbf{X}}(x_1, \dots, x_k) = \frac{\exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}))}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}}$

MLE (Maximum Likelihood Estimate)

We have A, B, C, D . $P(A | B) > P(A | C) > P(A | D)$
 $\implies B$ is the MLE of A . MLE Estimate of Anisotropic can be PSD.
 $\hat{\theta}_{MLE}(x) = \arg \max_{\theta} \Pi f(x | \theta) = \arg \max_{\theta} \ln L(\theta; x)$
 Mean is unbiased; Variance is biased (usually underestimate)
 Predicts parameter which max the probability of the data.
 Implicitly assumes uniform prior

MAP (Maximum a Posteriori)

We have A, B, C, D . $\mathbb{P}(A | B) > \mathbb{P}(C | B) > \mathbb{P}(D | B) \implies A$ is the MAP of B .

$$\hat{\theta}_{MAP} = \arg \max_{\theta} f(\theta | x) = \arg \max_{\theta} f(x | \theta) \cdot g(\theta)$$

Predicts the parameter which maximizes the conditional probability of the parameter given the data.
 Should be used when you have the prior probabilities.

MLE = MAP when all parameters have equal prior probability.
 The axis lengths of Gaussian Isocontours are σ_i s.t. $\sigma^2(X) = \text{Var}(X)$. Independent \iff uncorrelated (only for Multivariate Gaussian).

Bayesian Risk

L (loss function) is symmetric: pick class with max posterior prob.
 L is asymmetric: minimize $\mathbb{E}[L(\text{true class, prediction}) | \text{data}]$ or pick max loss-weighted posterior prob.
 The risk for r is the expected loss over all values of x, y . Equals to 0 when class distros don't overlap or prior prob for one class is 1.

$$R(r) = \mathbb{E}[L(r(X), Y)]$$

$$= \sum_x \left(\sum_{c \in \{-1, 1\}} L(r(x), c) P(Y = c | X = x) \right) P(X = x)$$

$$= \sum_{c \in \{-1, 1\}} \left(P(Y = c) \sum_x L(r(x), c) P(X = x | Y = c) \right)$$

$$R(\hat{y} = i | x) = \sum_{j=1}^C \lambda_{ij} P(Y = j | x)$$

The Bayes decision rule aka Bayes classifier is the fn r^* that minimizes functional $R(r)$. Assuming $L(z, y) = 0$ for $z = y$:

$$r^*(x) = \begin{cases} 1 & \text{if } L(-1, 1)P(Y = 1 | X = x) > L(1, -1)P(Y = -1 | X = x) \\ -1 & \text{otherwise} \end{cases}$$

Regression Methods

Model: $y = Xw$, Loss Function: least squares, $n \in N(X)$

Name	Objective	Solution
OLS	$\frac{1}{n} \ Y - Xw\ _2^2$	$w^* = (X^\top X + X^\dagger y + n) \in$
Ridge: Assumes Gaussian Priors	$\frac{1}{n} \ Y - Xw\ _2^2 + \lambda \ w\ _2^2$	$w^* = (X^\top X + n\lambda I)^{-1} X^\top y$
LASSO	$\frac{1}{n} \ Y - Xw\ _2^2 + \lambda \ w\ _1$	No closed form

Linear Algebra

Matrix Calculus

$$\nabla_{\vec{x}} \vec{w}^T \vec{x} = \left(\frac{\partial \vec{w}^T \vec{x}}{\partial \vec{x}} \right)^T = \vec{w}$$

$$\nabla_{\vec{x}} (\vec{w}^T A \vec{x}) = A^T \vec{w}$$

$$\nabla_A \vec{w}^T A \vec{x} = \vec{w} \vec{x}^T$$

$$\nabla_{\vec{x}} (\vec{x}^T A \vec{x}) = (A + A^T) \vec{x}$$

$$\nabla_{\vec{x}}^2 (\vec{x}^T A \vec{x}) = A + A^T$$

$$\nabla_{\vec{x}} (\alpha \vec{y}) = (\nabla_{\vec{x}} \alpha) \vec{y}^T + \alpha \nabla_{\vec{x}} \vec{y}$$

$$\nabla_{\vec{x}} \tilde{f}(\vec{y}) = (\nabla_{\vec{x}} \vec{y}) (\nabla_{\vec{y}} \tilde{f}(\vec{y}))$$

$$\nabla_{\vec{x}} (\vec{y} \cdot \vec{z}) = (\nabla_{\vec{x}} \vec{y}) \vec{z} + (\nabla_{\vec{x}} \vec{z}) \vec{y}$$

$$\nabla_{\vec{x}} C \vec{y}(\vec{x}) = (\nabla_{\vec{x}} \vec{y}(\vec{x})) C^T$$

$$\frac{\partial \|\vec{x}\|_2^2}{\partial \vec{x}} = \frac{\partial (\vec{x}^T \vec{x})}{\partial \vec{x}} = 2\vec{x}$$

$$\nabla_{\vec{y}} (\vec{y} - A\vec{x})^T W (\vec{y} - A\vec{x}) = 2W(\vec{y} - A\vec{x})$$

$$\nabla_{\vec{x}} (\vec{y} - A\vec{x})^T W (\vec{y} - A\vec{x}) = -2A^T W (\vec{y} - A\vec{x})$$

$$\nabla_{\vec{w}} (\|X\vec{w} - \vec{y}\|_2^2 + \lambda \|\vec{w}\|_2^2) = 2X^T X\vec{w} - 2X^T \vec{y} + 2\lambda \vec{w}$$

Matrix A is Positive Semi-Definite iff

- (a) $\forall \vec{x} \neq \vec{0} \in \mathbb{R}^n, \vec{x}^T A \vec{x} \geq 0$. Symmetric.
- (b) All eigenvalues of A are non-negative ($\lambda_i \geq 0$).
- (c) \exists unique matrix $L \in \mathbb{R}^{n \times n}$ such that $A = LL^T$ (Cholesky decomposition).

PSD Example: $A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$, with $\lambda = 3, 1$. $L = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$.

All diagonal entries of A are non-negative and $\text{Tr}(A) \geq 0$.

Sum of all the entries ≥ 0 . $\text{Var}(Mx) = M \text{Var}(x) M^T$, M is constant. $M \succ 0, N \succ 0, M - N \succeq 0 \implies N^{-1} - M^{-1} \succeq 0$
 $M \succeq 0, N \succeq 0 \implies M - N \succeq 0 \iff \lambda_{\min}(M) > \lambda_{\max}(N)$.

$$A = A^{\frac{1}{2}} A^{\frac{1}{2}} = U \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} U^T, A^{\frac{1}{2}} = U \Lambda^{\frac{1}{2}} U^T$$

A function is convex iff Hessian is PSD. Strict Convexity:

$$(\forall 0 < t < 1), f(tx_1 + (1-t)x_2) < tf(x_1) + (1-t)f(x_2)$$

Covariance Matrix

$$\Sigma = \frac{1}{n} \hat{X}^T X = \begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \dots & \text{Cov}(X_1, X_d) \\ \text{Cov}(X_2, X_1) & \text{Var}(X_2) & \dots & \text{Cov}(X_2, X_d) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(X_d, X_1) & \text{Cov}(X_d, X_2) & \dots & \text{Var}(X_d) \end{bmatrix}$$

$$= \mathbb{E}[(X - \mu)^T (X - \mu)] \text{ where } X \in \mathbb{R}^{n \times d}, \text{ all diag entries } > 0$$

Symmetric, PSD $\implies \exists \Sigma = V \Lambda V^T$ by Spectral Theorem. PD

\implies symmetric in this class. Eigenvectors are orthogonal

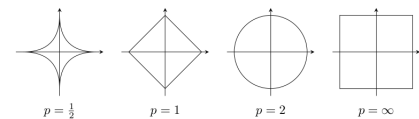
directions along which points are uncorrelated.

$$\Sigma^{-1} = V \Lambda^{-1} V^T = \sum_i \frac{1}{\Lambda_{ii}} v_i v_i^T$$

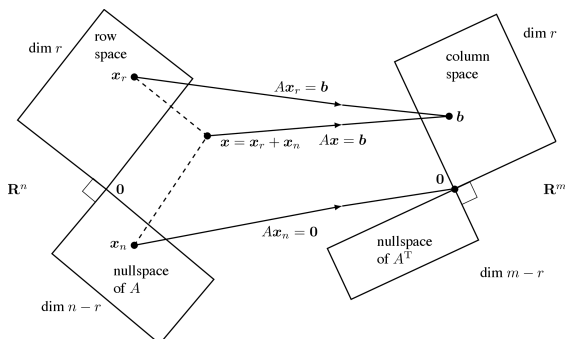
Spectral Theorem: $A = V \Lambda V^T$. All real+symmetric $n \times n$ matrices have real eigenvalues and n eigenvectors that are mutually orthogonal: $v_i^T v_j = 0 \quad \forall i \neq j$.

Norm Ball

ℓ_0 and ℓ_1 encourage sparsity (more than ℓ_2).



Fundamental Theorem of Linear Algebra



$$(N(A)^\perp = R(A^T)) \oplus (N(A^T A) = N(A) = R(A^T)^\perp) = \mathbb{R}^n$$

$$(N(A^T)^\perp = R(A)) \oplus (N(A^T) = R(A)^\perp) = \mathbb{R}^m$$

Rank-nullity Theorem: $\dim(R(A)) + \dim(N(A)) = n$

Jensen's Inequality: If $f(x)$ is strictly convex, $\mathbb{E}[f(x)] > f(\mathbb{E}[x])$.

$$\dim(\text{Row}(X)) = \dim(R(X^T)) = \text{rank}(X^T) = \text{rank}(X)$$

$$\text{Row}(X^T X) = R(X^T X) = \text{Row}(X) = R(X^T)$$

Update Rule

Gradient Descent: $w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla_w J(w^{(t)})$

Logistic Reg: $w^{(t+1)} \leftarrow w^{(t)} + \epsilon X^T (y - s(Xw^{(t)}))$

Newton's Method: $w^{(t+1)} \leftarrow w^{(t)} - (\nabla_w^2 J(w^{(t)}))^{-1} \nabla_w J(w^{(t)})$

*** Note:

If J quadratic, Newton's method only needs one step to find exact solution. Newton's Method doesn't work for most non-smooth functions, and is generally faster than BGD/SGD.

Stochastic GD: $w \leftarrow w - \epsilon \nabla_w J(w)_i$ for some $i \in U([1, \dots, n])$

Logistic Reg: $w \leftarrow w + \epsilon (y_i - s(X_i \cdot w)) X_i$

Cost Functions

$y_i = f(X_i) + \epsilon_i$: ϵ_i from Gaussian, all ϵ_i same mean, all y_i same var

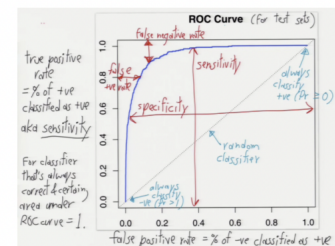
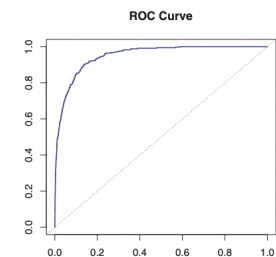
General: $J = \sum_{i=1}^n L(X_i \cdot w, y_i)$

Linear: $J = \sum_{i=1}^n (X_i \cdot w + \alpha - y_i)^2 = \|Xw - y\|_2^2$

Logistic: $J = -\sum_{i=1}^n (y_i \ln s(X_i \cdot w) + (1 - y_i) \ln(1 - s(X_i \cdot w)))$

Weight LS: $J = \sum_{i=1}^n w_i (X_i \cdot w - y_i)^2 = (Xw - y)^T \Omega (Xw - y)$

ROC Curve



Design Matrix

Centering:

subtracting μ^T from each row of X : $X \rightarrow \tilde{X}$

Decorrelating:

Applying rotation $Z = \tilde{X}V$ where $\text{Var}(X) = V \Lambda V^T$. Covariance matrix of Z is Λ (diagonal)

Sphering:

$W = \tilde{X} \text{Var}(X)^{-1/2}$ ($\Sigma^{-1/2}$: ellipsoid to sphere)

Whitening:

Perform centering, and then sphering

Bias-Variance Tradeoff

Statistical Bias: $\mathbb{E}[\hat{\theta} - \theta] = \mathbb{E}[\hat{\theta}] - \theta$.

Bias: error due to inability of hypothesis h to fit g perfectly e.g., fitting quadratic g with a linear h

Variance: error due to fitting random noise in data e.g., we fit linear g with a linear h , yet $h \neq g$.

Overfitting: Low Bias, High Variance

Underfitting: High Bias, Low Variance.

Adding a feature usually increases variance [don't add a feature unless it reduces bias more]. Adding a feature results in a non-increasing bias.

Forward/Backward stepwise selection aren't guaranteed to find optimal features. Backward stepwise selection looks at $d' - 1$ features at a time, where d' is current num of features (one at a time). Use Forward selection if we think few features important, Backward selection if many features important.

higher residuals \implies higher bias

higher complexity \implies higher variance

$$\text{Var}(h(z)) = E[(h(z) - E[h(z)])^2] \approx \sigma^2 \frac{d}{n}$$

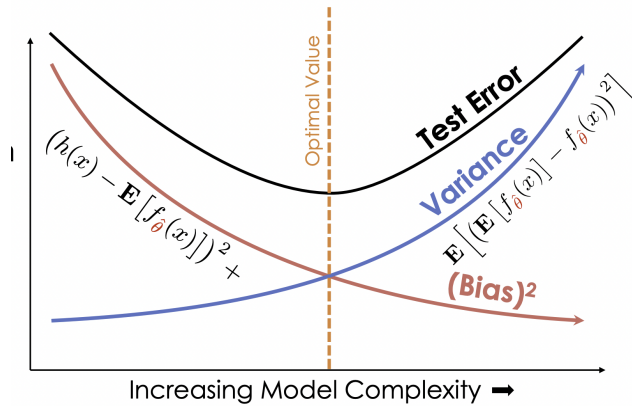
Bias-Variance Decomposition:

$$\text{Model Risk} = \mathbb{E}[L(h(z), \gamma)] = \mathbb{E}[(h(z) - \gamma)^2]$$

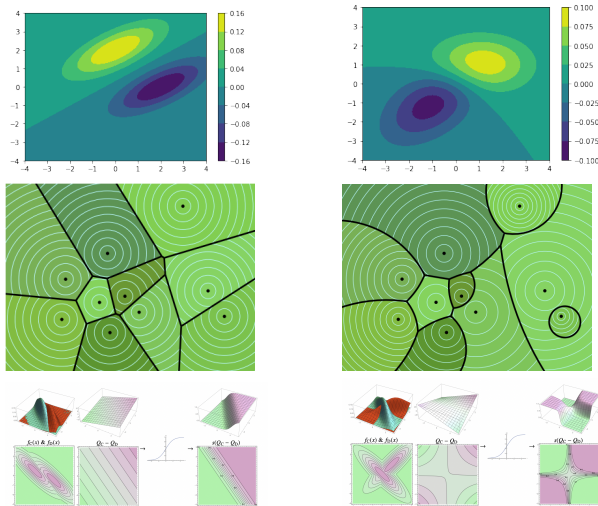
$$= \underbrace{(E[h(z)] - g(z))^2}_{\text{bias}^2 \text{ of method}} + \underbrace{\text{Var}(h(z))}_{\text{variance of method}} + \underbrace{\text{Var}(\epsilon)}_{\text{irreducible error}}$$

where $E[\gamma] = g(z)$; $\text{Var}(\gamma) = \text{Var}(\epsilon)$.

Note: the model determines Bias-Variance Tradeoff, not the algorithm used to solve the model/optimization problem.

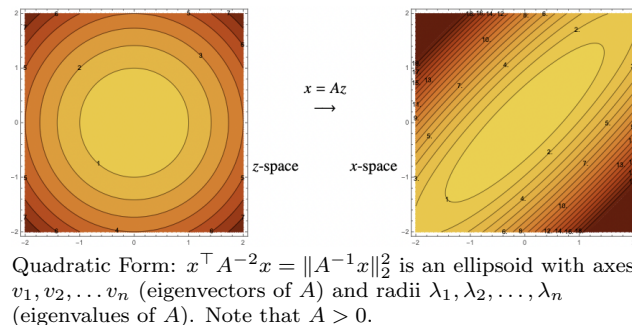


Isocontour/Voronoi Diagrams



LDA: same variance; decision boundary is linear

QDA: different variance; decision boundary is curved towards class(es) w/ lower variance



Gaussian with covariance matrix $\Sigma = \frac{1}{n} \hat{X}^T \hat{X}$ isocontours with radii of length $\sqrt{\lambda_i(\Sigma)} = \sigma_i(X)$

Decision Trees

Tree with each node denoting a split over some feature. Leaf node specifies class. deep decision tree = overfit = high variance

Random Forests

At each node, take rnd sample of m features (out of d).

Classification: $m_{\text{init}} \approx \sqrt{d}$

Regression: $m_{\text{init}} \approx \frac{d}{3}$

Smaller m = less features = less complex model = more bias

Kernels

Can speedup algorithms such as SVMs, k NN, Regression (linear & logistic), k -means, etc.

$w = X^T a = \sum_{i=1}^n a_i X_i$ Substitute this into the algorithm so we have to optimize n weights a instead of d weights w

Kernel Ridge Regression: Center X and y so means are 0, $X_{i,d+1} = 1$. Solve normal equations: $(X^T X + \lambda I)w = X^T y$

If a is a solution to $(X^T X + \lambda I)a = y$,

$X^T y = X^T X X^T a + \lambda X^T a = (X^T X + \lambda I)X^T a \Rightarrow w = X^T a$

The dual: $\min ||X X^T a - y||^2 + \lambda ||X^T a||^2$

Test phase: $h(z) = w^T z = a^T X z = \sum_{i=1}^n a_i (X_i^T z)$, if $X_i^T z$ are precomputed, it takes $O(n)$ time to evaluate h vs $O(d)$ time for primal method.

kernel fn: $k(x, z) = x^T z$, kernel mtx: $K = X X^T$

$(K_{ij} = k(X_i, X_j))$

kernel matrix must be PSD + symmetric

Solve $(K + \lambda I)a = y$ $O(n^3)$, test $h(z) = \sum a_i k(X_i, z)$ $O(nd)$ time

Dual: $O(n^3 + n^2 d)$, Primal: $O(d^3 + d^2 n)$, dual is better when $d > n$.

Kernel Trick: Polynomial kernel $k(x, z) = (x^T z + 1)^p$

Can compute $k(x, z) = \Phi(x)^T \Phi(z)$ in $O(d)$ time instead of $O(d^p)$

Enables us to implicitly handle polynomial features efficiently

Kernel Perceptrons:

$w \leftarrow y_i \Phi(X_i)$: while some $y_i \Phi(X_i) \cdot w < 0$, $w \leftarrow w + \epsilon y_i \Phi(X_i)$

for each test pt z , $h(z) \leftarrow w \cdot \Phi(z)$

Let $\Phi(X)$ be $x \times D$ matrix with rows $\Phi(X_i)^T$

Dualize with $w = \Phi(X)^T a$, $a_i \leftarrow a_i + \epsilon y_i$,

$h(z) = \sum_{i=1}^n a_j + k(X_{ij}, z)$

$\Phi(X_i) \cdot w = (\Phi(X)w)_i = (\Phi(X)\Phi(X)^T a)_i = (Ka)_i$

$a \leftarrow [y_1 \ 0 \ \dots \ 0]^T$: while some $y_i (Ka)_i < 0$, $a_i \leftarrow a_i + \epsilon y_i$

$O(n^2 d)$ to kernel mtx $O(1)$ to update a , $O(n)$ to update Ka

Learning Theory

Range Space (P, H) : P = set of all possible train/test pts (ex. \mathbb{R}^d)

H = set of all possible hypotheses Dichotomy: split of input data into two separate classes, not necessarily linear decision boundary. Shatter Function: $\Pi_H(X) = |\{X \cap h : h \in H\}|$ = maximum number of dichotomies hypothesis class H can produce in particular set of points X .

$\Pi_H(n) = \max_{|X|=n, X \subseteq P} \Pi_H(X)$ = maximum number of dichotomies

hypothesis class H can produce amongst a set of n data points and is only ever 2^n (known as shattering) or polynomial in n .

VC Dimension = the largest number of points D s.t.

$\Pi_H(D) = 2^n$. Basically largest number of points a hypothesis class can produce all dichotomies of. Can be infinite, can be 0. A linear perceptron classifier with d parameters (d -dimensional weight vectors) has a VC dimension of d . For example, 2d linear perceptron (three weights - one for each dimension and one fictitious dimension = bias) has VC dimension of 3.

AdaBoost

Ensemble method that trains multiple learners on weighted sample points and weights learner. (misclassified points = increased weights, accurate learners = increased weights). Find classifier G_T and coefficient β_T to minimize Risk = $\frac{1}{n} \sum_{i=1}^n L(M(X_i), y_i)$ with $M(X_i) = \sum_{t=1}^T \beta_t G_t(X_i)$

AdaBoost metalearner uses $L(\rho, \ell) = e^{-\rho \ell}$

$w_i^{(T+1)} = w_i^{(T)} \exp(-\beta_T y_i G_T(X_i))$, $\beta_T = \frac{1}{2} \ln \left(\frac{1 - \text{err}_T}{\text{err}_T} \right)$

$\text{err}_T = \frac{\sum_{y_i \neq G_T(X_i)} w_i^{(T)}}{\sum_{i=1}^n w_i^{(T)}}$. metalearner:

$h(z) = \text{sign}(\sum_{t=1}^T \beta_t G_t(z))$

Neural Nets

Multi-layered perceptron, each layer puts outputs of previous layer linear function and then activation function. Minimize loss via gradient descent

Principal Component Analysis

PCA only makes linear relations. Low-rank approximation is lossy feature selection: you lose lower variance components yet select out low-singular values (often “noise” components) XX^T

Clustering

NP-hard: find $\text{argmin}_y \sum_{i=1}^k \sum_{y_j=i} ||X_j - \mu_i||^2$

K-means: alternate between fixed y_j 's update μ_i 's and vice versa. Halt when step 2 changes no assignments. Both steps decrease objective fn unless they change nothing; alg must terminate.

Initialization: Forgy method (choose k random points to be centroids). Can also use the (worse) random partitions - randomly assign each point to a cluster. K-medoids: K-means, but instead of using mean, use medoid, sample point minimizing total distance to other points in cluster.

Hierarchical clustering: creates a tree, every subtree is a cluster. Bottom-up (agglomerative) start with each point a cluster; repeatedly fuse pairs minimizing $d(A, B)$. Linkage functions:

complete: $d(A, B) = \max\{d(w, x) : w \in A, x \in B\}$

single: $d(A, B) = \min\{d(w, x) : w \in A, x \in B\}$

average: $d(A, B) = \frac{1}{|A||B|} \sum_{w \in A} \sum_{x \in B} \text{dist}(w, x)$

centroid: $d(A, B) = \text{dist}(\mu_A, \mu_B)$

Top-down (divisive) starts with single cluster, repeatedly splits

Miscellaneous

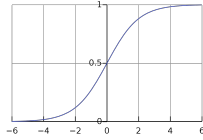
NP-Hard to find optimal linear classifier

Bayes vs. GDA Bayes uses true mean/variance, while
GDA uses sample mean/variance.
True mean/variance equal \nRightarrow Sample
mean/variance equal

Cauchy-Schwarz $|\langle x, y \rangle| \leq \|x\| \cdot \|y\|$

Sigmoid Function: Graph:

$$s(\gamma) = \frac{1}{1+e^{-\gamma}}$$



Unique Optimum Only ridge regression has one unique optimum (not Least Squares, Lasso, or Logistic).

Training Data: Training on less data can improve training accuracy, training on more data can improve validation/test accuracy.

$$\ln(ab) = \ln(a) + \ln(b); \quad a^{x+y} = a^x \cdot a^y$$