CS 189 Notes — Jeffrey Shen

①② Classifiers

n observations, each w/ d features

Perceptron Alg: slow but for linearly separable

Risk func: $R(w) = \frac{1}{n}\sum_{i \in V} -y_i X_i \cdot w$

③ Grad Descent

Grad Descent: $w \leftarrow w + \epsilon \sum_{i \in V} y_i X_i$

Stochastic Grad Desc: $w \leftarrow w + \epsilon y_i X_i$

④ Support Vector Mach (SVM); Features

Hard Margin SVM: find w, α to min $\|w\|^2$

constraints $y_i(w \cdot X_i + \alpha) \geq 1$

Soft Margin SVM: slack var to violate

find w, α and ξ to min $\|w\|^2 + C\sum_{i=1}^{n} \xi_i$

st $y_i(X_i \cdot w + \alpha) \geq 1 - \xi_i$, $\xi_i \geq 0$

big C: keep slack var small, overfitting, sensitive to outliers

Features: make nonlinear features into higher dim space, separable

⑤ ML Abstractions & Numerical Optimization

1) Unconstrained Optimization: min cont. obj func $f(w)$

2) Constrained: $f(w), g(w) = 0$, find w min $f(w)$

3) LP: max $c \cdot w$ st $Aw \leq b$, n linear constraints $A_i \cdot w \leq b_i$

4) QP: find w: min $f(w) = w^T Q w + c^T w$ st $A w \leq b$

Positive Definite (PD): $w^T Q w > 0$ for all $w \neq 0$

⑥ Design Theory

Risk: expected loss over x, y $E[L(r(x), Y)]$

Bayes Classifier: min $R(r)$ given $L(z,y) = 0$ for $z = y$

optimal Risk $R(r^*) = \int \min_{y \in \{1,-1\}} L(-y, y) f(X = x | Y = y) P(Y = y) dx$

⑦ Gaussian Discriminant Analysis

Assume Normal $X \sim N(\mu, \sigma^2) = \frac{1}{(\sqrt{2\pi}\,\sigma)^d} e^{\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}$

QDA: diff σ, μ

To recover posterior probability $S = \frac{1}{1+e^{-x}}$

$$P(Y=C|X) = \frac{f(x|y=C)\,\pi_C}{f(x|Y=C)\pi_C + f(x|Y=D)\pi_D} = S(Q_C(x) - Q_D(x))$$

LDA: same σ, diff μ, approx Bayes Decision rule

$$\max \frac{\mu_C \cdot x}{\sigma^2} - \frac{\|\mu_C\|^2}{2\sigma^2} + \ln \pi_C, \quad P(Y=C|X=x) = \delta(w \cdot x + \alpha)$$

Decision Boundary: $Q_C(x) - Q_D(x) = 0$

MLE: estimate model params

$$L(\mu, \sigma; X_1 \dots X_n) = f(x_1) f(x_2) \dots f(x_n)$$

log likelihood l: $\max \sum_{i=1}^{n} \left(-\frac{\|X_i - \mu\|^2}{2\sigma^2} - d \ln \sqrt{2\pi} - d \ln \sigma\right)$

$$\hat{\sigma}^2 = \frac{1}{dn}\sum_{i=1}^{n} \|X_i - \mu\|^2$$

⑧ Eigenvectors & Multivariate Normal Distr

if A is invertible, v is eigenvector of A^{-1} w/ eigenvalue $1/\lambda$

if v is an eigenvector of A w/ eigenvalue λ

v is eigenvector of A^k w/ eigenvalue λ^k

⑨⑩ Regression, LS, Linear & Logistic Regression

converges

Ridge: linear reg func + squared loss + L2 reg

Lasso: linear reg func + squared loss + L1 reg

Sparse sol

Logistic: logistic reg func + weighted sum + mean cost

LS: linear reg func + squared loss + mean cost

⑪ Regression, Newton's Method, ROC

LS: w that min $\|Xw - y\|^2$

$w = (X^T X)^{-1} X^T y$

Logistic Regression: find w min

$$J = \sum_{i=1}^{n} L(x_i \cdot w, y_i) = -\sum_{i=1}^{n} \left(y_i \ln s(x_i \cdot w) + (1-y_i)\ln(1 - s(x_i \cdot w))\right)$$

$$\nabla_w J = -\sum \left(\frac{y_i}{s_i}\nabla s_i - \frac{1-y_i}{1-s_i}\nabla s_i\right) = -X^T(y - s(Xw))$$

Gradient descent: $w \leftarrow w + \epsilon X^T(y - s(Xw))$

Stochastic Gradient descent: $w \leftarrow w + \epsilon(y_i - S(X_i \cdot w))X_i$

Newton's Method: $w + e$: $(X^T S X)e = X^T(y - s)$

⑫ Bias-Variance Decomposition

Bias: error due to inability of hypothesis h to fit g perfectly, wrong model

Variance: error from fitting random noise in data,

- Underfitting = too much bias
- overfitting = too much variance
- training error tests bias not variance, test error for both

- variance $\to 0$ as distributions $n \to \infty$
- if h fit g well, distributions bias $\to 0$, $n \to \infty$
- good feature reduces bias
- adding feature increases variance

variance decreases as $\frac{1}{n}$ (sample pts), increases as d (features)

⑬ Ridge & Lasso Regression

Ridge: w to min $\|Xw - y\|^2 + \lambda\|w\|^2$

reg reduces overfitting by reducing variance

Maximum a posteriori (MAP): using likelihood, but max posterior

posterior $f(w|x,y) = \frac{f(y|x,w) \times \text{prior } f(w)}{f(y|x)} = \frac{L(w) f(w)}{f(y|x)}$

Maximize log posterior $= \ln L(w) + \ln f(w) - \text{const}$

$\Rightarrow$ Minimize $\|Xw - y\|^2 + \lambda\|w\|^2$

Lasso: $\min \|Xw - y\|^2 + \lambda\|w\|_1$

Feature Subset Selection: find poorly predictive features

Heuristic 1: Forward Step Wise Selection $O(d^2)$

1) Start w/ null model (0 features)
2) add best feature until valid error increasing

Heuristic 2: Backward stepwise selection $O(d^2)$

1) start w/ d features
2) remove gives best reduction in valid error

⑭ Decision Trees

classification & regression

Tree w/ 2 node types:

- internal nodes to test 1 feature value & branch
- leaf nodes specify class $h(x)$

Entropy: $H(S) = -\sum_c p_c \log_2 p_c$ $\quad p_c = \frac{|\{j \in S : y_j = c\}|}{|S|}$

Ex same class: 0, half C, half D: 1, n pts, diff classes: $\log_2 n$

Weighted Avg Entropy: $H_{\text{after}} = \frac{|S_l| H(S_l) + |S_r| H(S_r)}{|S_l| + |S_r|}$

Information gain: $H(S) - H_{\text{after}}$ max info gain

⑮ Decision Tree, Ensemble Learning, Random Forest

Multivariate Split: find non-axis aligned splits or generate randomly

Decision Tree Regression: creates piecewise constant regression func

Stopping Early: most points same, depth too great, prune

Pruning Greedily remove each split that improves validation, more reliable

Ensemble Learning: decrease variance

Bagging: same learning alg on random subsets of one training set

Random Forests: randomized decision trees on random subsamples

⑯ Kernel Trick

Optimize $w = X^T a = \sum_{i=1}^{n} a_i X_i$, $(X^T X + \lambda I) w = X^T y$

For solution a, $(XX^T + \lambda I)\, a = y$, $(X^T X + \lambda I) X^T a = X^T y$

Dual Form Ridge Regression: $\min \|XX^T a - y\|^2 + \lambda\|X^T a\|^2$

Training: Solve $(XX^T + \lambda I)a = y$

Testing: Regression $h(z) = w^T z = a^T X z = \sum_{i=1}^{n} a_i (X_i^T z)$

Kernel func: $k(x,z) = x^T z$

Kernel matrix: $K = XX^T$, $K_{ij} = k(X_i, X_j)$

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

Thm: $(x^T z + 1)^p = \Phi(x)^T \Phi(z)$

Gaussian kernel: Φ st $k(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$

⑰ Neural Networks

classification, regression

use a logistic function between linear combinations

Network w/ 1 Hidden Layer

Input: $x_1 \dots x_d$; $x_{d+1} = 1$ Hidden: $h_1 \dots h_m$; $h_{m+1} = 1$

Output: $z_1 \dots z_k$

Layer 1 weights: $m \times (d+1)$ matrix V V_i^T row i

Layer 2 weights: $k \times (m+1)$ matrix W W_i^T row i

Back-propagation

- dynamic programming algorithm for computing gradients we need for grad descent

⑱ Neurobio, Variations on NN

ReLU, ramp, hinge fn: $r'(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$

⑲ Better NN Training, CNN

Heuristics for Faster Training

1) Stochastic Gradient Descent: faster than batch on larger, redundant datasets
2) Epoch: presents every training pt once
3) Normalizing: center mean at zero, scale so variance = 1
4) Different learning rate for each layer of weights

Heuristic to Avoid Overfitting

1) Fewer hidden units
2) Weight decay: l_2 regularization

Convolutional Neural Networks

1) Local Connectivity: hidden layer to small patch of prev layer
2) Shared weight: hidden units share mask, filter, kernel

Convolution: same linear transformation applied to diff parts of image by shifting

⑳ Unsupervised Learning & Principal Components Anal

Unsupervised Learning: sample points but no labels, discover structure

PCA: find k directions to capture variance

Find w $\max \text{Var}(\{\tilde{X}_1, \tilde{X}_2 \dots \tilde{X}_n\}) = \frac{1}{n}\sum_{i=1}^{n}\left(X_i \cdot \frac{w}{\|w\|}\right)^2 = \frac{1}{n}\frac{\|Xw\|^2}{\|w\|^2} = \frac{1}{n}\frac{w^T X^T X w}{w^T w}$

㉑ Singular Value Decomposition, Clustering

$X = U D V^T = \sum_{i=1}^{d} \delta_i U_i V_i^T$ (rank 1 outer product matrix)

X: $n \times d$; U: $n \times d$; D: $d \times d$ diagonal $\delta_1 \dots \delta_d$; V^T: $d \times d$, $V_1^T \dots V_d^T$

left singular vectors $\to U^T U = I$ & right singular vectors $V^T V = I$

$\delta_1, \dots \delta_d$ are nonnegative singular values of X

V_i is an eigenvector of $X^T X$ w/ eigenvalue δ_i^2

Row i of UD gives principle coordinates of sample point X_i

K-means clustering (Lloyd's Algorithm) partition n points into k disjoint clusters, assign X_i a cluster label $y_i \in [1, k]$

$$\min_y \text{ in } \sum_{j=1}^{k} \sum_{y_i = j} \|X_i - \mu_j\|^2$$

Hierarchical Clustering

Creates a tree, every subtree is a cluster

Agglomerative Clustering: each point a cluster and combine

Hierarchical Clustering: all pts in one cluster, split

Dist func for clusters A, B

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

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

Average Linkage: $d(A,B) = \frac{1}{|A||B|}\sum_{w\in A}\sum_{x\in B} d(w,x)$

Centroid Linkage: $d(A,B) = d(u_A, u_B)$ u mean of S

22 High Dimensions, Random Projection, Pseudoinv

p: sampled from univariate normal dist w/ mean 0, variance 1

$p_i \sim N(0,1)$ $p_i^2 \sim \chi^2(1)$ $E[p_i^2]=1$ $Var(p_i^2)=2$

$E[\|p\|^2] = d\,E[p_i^2] = d$ $Var(\|p\|^2) = d\,Var(p_i^2) = 2d$ $SD(\|p\|^2) = \sqrt{2d}$

Random Projection: Project onto random subspace, could preserve dist better

Choose small ϵ, small β, random subspace $S \subset \mathbb{R}^d$ dimension k, $k = \left\lceil \frac{2\ln(\frac{1}{\beta})}{\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}} \right\rceil$

For pt q, let $\vec{q}$ be orthogonal proj q onto S, multiplied by $\sqrt{\frac{d}{k}}$

Pseudo inverse D^+ by transposing D and replacing nonzeros w/ reciprocal

$DD^+D = D$ $D^+DD^+ = D^+$ $D^2D^+ = D$

X is $n \times d$ matrix $X = UDV^T$ $rank(D) = rank(X)$

Moore-Penrose pseudoinverse X^+ is $X^+ = VD^+U^T$

1) $XX^+ = UDV^TVD^+U^T = U(DD^+)U^T$ symmetric, PSD
2) $X^+X = VD^+U^TUDV^T = V(D^+D)V^T$ symmetric, PSD
3) same rank: $D, D^+, DD^+, D^+D, X, X^+, XX^+, X^+X$
4) $XX^+X = X$
5) $X^+XX^+ = X^+$

Solution to normal eq $X^TXw = X^Ty$ is $w = X^+y$

23 Learning Theory

Constrain hypotheses we allow learner to consider

Range Space (Set System): pair $(P, \mathcal{H})$, P set of all possible test/training

$\mathcal{H}$: hypotheses class, set of hypotheses

1) Power set classifiers: P set of h numbers, $\mathcal{H}$ power set of P, 2^h subsets of P
2) Linear classifier: $P = \mathbb{R}^d$, $\mathcal{H}$ set of all halfspaces form $\{x : w \cdot x \geq -\alpha\}$

Hoeffding's inequality: prob of bad estimate, how likely number drawn from binomial distr far from mean

$P_r(|\hat{R}(h) - R(h)| > \epsilon) \leq 2e^{-2\epsilon^2 n}$

Dichotomy: $X \cap h$, $h \in \mathcal{H}$, func that assigns training point to C or not-C

Shatter func: $\pi_{\mathcal{H}}(n) = \max_{|X|=n, X\subseteq P} \pi_{\mathcal{H}}(X)$ most dichotomies of point set size n

Vapnik-Chervonenkis dimension: $VC(\mathcal{H}) = \max\{n : \pi_{\mathcal{H}}(n) = 2^n\}$

VC dimension is upper bound on exponent of polynomial

24 Boosting; Nearest Neighbor Classification

AdaBoost: "adaptive Boosting" ensemble method for classification

- trains multiple learners on weighted sample points
- diff weights for each learner
- increases weights of misclassified training pts
- bigger vote to accurate learners

- Meta-learner: linear combination of learners, $M(z) = \sum_{r=1}^{T} \beta_r G_r(z)$

$\min_{G_r, \beta_r} \frac{1}{n}\sum_{i=1}^{n} L(M(x_i), y_i)$, $M(x_i) = \sum_{r=1}^{T} \beta_r G_r(x_i)$, $L(\rho, l) = e^{-\rho l} = \begin{cases} e^{-\rho} & l = +1 \\ e^{\rho} & l = -1 \end{cases}$

optimal β_T after $\frac{d}{d\beta_T} Risk = 0$, $\beta_T = \frac{1}{2}\ln\left(\frac{1-err_T}{err_T}\right)$

Boosting Benefits ↑: fast, no hyperparameter, subset selection,

Nearest Neighbor Classification

Idea: query point q find k sample pts nearest q distance metric

Regression: return avg label of k pts

Classification: return class w most votes from k pts

25 NN Algos, Voronoi Diagrams, k-D trees

Exhaustive k-NN Alg: query pt q

- Scan through all pts and find squared dist to q
- Keep max-heap w/ k shortest dist found so far

Voronoi Diagrams: X point set

Voronoi cell: $w \in X$ is $Vor\,w = \{p \in \mathbb{R}^d : \|p-w\| \leq \|p-v\|\ \forall v \in X\}$

Point location: for query point $q \in \mathbb{R}^d$, find point $w \in X$ where $q \in Vor\,w$

↳ good for 1-NN in 2 or 3 dimensions

k-d Trees: Decision Trees for NN search

- Choose splitting feature: w/ greatest width
  - ↳ splitting value: median point for feature i

Misc

Linear Algebra

L2 Norm (Euclidean): $\|x\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{x^Tx}$

L1 Norm: $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

L∞ Norm: $\|x\|_\infty := \max_{1\leq i\leq n} |x_i|$

Frobenius Norm: $\|A\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2} = \sqrt{Tr(A^TA)}$

Orthonormal: $u_i^Tu_j = 0$, $i \neq j$ and $\|u_i\|_2 = 1$

Trace: $Tr\,A = \sum_{i=1}^{n} A_{ii}$ sum of diagonal elements

Cauchy-Schwartz Inequality: $|x^Ty| \leq \|y\|_2 \cdot \|x\|_2$

Fundamental Theorem of Linear Algebra

Range of a matrix is the orthogonal complement of the nullspace of its transpose

$R(A)^\perp = N(A^T)$

Spectral Theorem (Symmetric eigenvalue decomposition (SED))

$A = \sum_{i=1}^{n} \lambda_i u_i u_i^T = U\Lambda U^T$, $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$

$Tr\,\Sigma = \frac{1}{m}XX^T = \frac{1}{m}\|X\|_F^2$

Total Variance: $Tr\,\Sigma = Tr(U\Lambda U^T) = Tr(U^TU\Lambda) = Tr\Lambda = \lambda_1 + \ldots + \lambda_n$

rank: linearly independent columns $X: \mathbb{R}^{m\times n}$

Rank-Nullity Theorem: $n - dim(nullspace(X)) = rank(A)$

$dim(rowspace(X)) = dim(columnspace(X^T)) = rank(X^T) = rank(X)$

- Symmetric matrix has real eigenvalues
- eigenvalues neg if concave
- Axis scaled by square roots of eigenvalues of Σ
- Covar = $\frac{XX^T}{n}$ $X \in \mathbb{R}^{n\times d}$

Symmetric matrix M

Positive Definite: if $w^TMw > 0$ all $w \neq 0$ ⇔ pos eigenvalues

Positive Semidefinite: if $w^TMw \geq 0$ all w ⇔ nonnegative eigenvalues

Indefinite: if pos & neg eigenvalue

Invertible: no zero eigenvalue

Convex: $x^2 \cup$

Concave: $\sqrt{x} \cap$

$\cos\theta = \frac{x\cdot y}{\|x\|\|y\|} = \frac{x}{\|x\|}\cdot\frac{y}{\|y\|}$ (each factor has length 1)

$E[\cos\theta] = 0$ $SD(\cos\theta) = \frac{1}{\sqrt{d}}$

Probability

Bayes Rule: $P(y=1|x) = \frac{P(x|y=1)P(y=1)}{P(x)}$ (posterior, prior)

$P(A,B) = \sum_C P(A,B,C)$

Chain Rule: $P(A,B,C) = P(A,B|C)P(C) = P(A|B,C)P(B|C)P(C)$

A conditionally independent given C: $P(A,B|C) = P(A|C)P(B|C)$

A independent of B given C: $P(A|B,C) = P(A|C)$

A,B independent: $P(A,B) = P(A)P(B)$

$P(A|B,C) = \frac{P(A,B,C)}{P(B,C)} = \frac{P(A,B|C)}{P(B|C)}$

$P(x) = P(x|y=1)P(y=1) + P(x|y=-1)P(y=-1)$

Matrix Derivatives

$$\frac{\partial \mathbf{x}^T\mathbf{a}}{\partial \mathbf{x}} = \frac{\partial \mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}$$

$$\frac{\partial \mathbf{a}^T\mathbf{X}\mathbf{b}}{\partial \mathbf{X}} = \mathbf{a}\mathbf{b}^T$$

$$\frac{\partial \mathbf{a}^T\mathbf{X}^T\mathbf{b}}{\partial \mathbf{X}} = \mathbf{b}\mathbf{a}^T$$

$$\frac{\partial \mathbf{a}^T\mathbf{X}\mathbf{a}}{\partial \mathbf{X}} = \frac{\partial \mathbf{a}^T\mathbf{X}^T\mathbf{a}}{\partial \mathbf{X}} = \mathbf{a}\mathbf{a}^T$$

$$\frac{\partial \mathbf{X}}{\partial X_{ij}} = \mathbf{J}^{ij}$$

$$\frac{\partial (\mathbf{X}\mathbf{A})_{ij}}{\partial X_{mn}} = \delta_{im}(\mathbf{A})_{nj} = (\mathbf{J}^{mn}\mathbf{A})_{ij}$$

$$\frac{\partial (\mathbf{X}^T\mathbf{A})_{ij}}{\partial X_{mn}} = \delta_{in}(\mathbf{A})_{mj} = (\mathbf{J}^{nm}\mathbf{A})_{ij}$$

$$\frac{\partial}{\partial X_{ij}}\sum_{klmn} X_{kl}X_{mn} = 2\sum_{kl} X_{kl}$$

$$\frac{\partial \mathbf{b}^T\mathbf{X}^T\mathbf{X}\mathbf{c}}{\partial \mathbf{X}} = \mathbf{X}(\mathbf{b}\mathbf{c}^T + \mathbf{c}\mathbf{b}^T)$$

$$\frac{\partial (\mathbf{B}\mathbf{x}+\mathbf{b})^T\mathbf{C}(\mathbf{D}\mathbf{x}+\mathbf{d})}{\partial \mathbf{x}} = \mathbf{B}^T\mathbf{C}(\mathbf{D}\mathbf{x}+\mathbf{d}) + \mathbf{D}^T\mathbf{C}^T(\mathbf{B}\mathbf{x}+\mathbf{b})$$

$$\frac{\partial (\mathbf{X}^T\mathbf{B}\mathbf{X})_{kl}}{\partial X_{ij}} = \delta_{lj}(\mathbf{X}^T\mathbf{B})_{ki} + \delta_{kj}(\mathbf{B}\mathbf{X})_{il}$$

$$\frac{\partial (\mathbf{X}^T\mathbf{B}\mathbf{X})}{\partial X_{ij}} = \mathbf{X}^T\mathbf{B}\mathbf{J}^{ij} + \mathbf{J}^{ji}\mathbf{B}\mathbf{X} \quad (\mathbf{J}^{ij})_{kl} = \delta_{ik}\delta_{jl}$$

$$\frac{\partial \mathbf{x}^T\mathbf{B}\mathbf{x}}{\partial \mathbf{x}} = (\mathbf{B} + \mathbf{B}^T)\mathbf{x}$$

$$\frac{\partial \mathbf{b}^T\mathbf{X}^T\mathbf{D}\mathbf{X}\mathbf{c}}{\partial \mathbf{X}} = \mathbf{D}^T\mathbf{X}\mathbf{b}\mathbf{c}^T + \mathbf{D}\mathbf{X}\mathbf{c}\mathbf{b}^T$$

$$\frac{\partial}{\partial \mathbf{X}}(\mathbf{X}\mathbf{b}+\mathbf{c})^T\mathbf{D}(\mathbf{X}\mathbf{b}+\mathbf{c}) = (\mathbf{D}+\mathbf{D}^T)(\mathbf{X}\mathbf{b}+\mathbf{c})\mathbf{b}^T$$

$Tr(AB) = Tr(BA)$

$(A^T)^T = A$

$(A+B)^T = A^T + B^T$

$(AB)^T = B^TA^T$

PSD $Q = PDP^T$

$PD^{1/2}D^{1/2}P^T$

$= UU^T$

$A = A^{1/2}A^{1/2}$

$= (PD^{1/2})(PD^{1/2})^T = PDP^T$

Eigenvalues 2×2

$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ $m \pm \sqrt{m^2 - p}$

$\frac{trace}{2}$ $det(M)$ ad-bc

MC

- rotation does not change principal comp
- dot prod $x \cdot v_i$ to get principle comp
- variance: how much change w/ diff dataset
  - ↳ high var → overfit
- bias: how well model predicts training
- ↑ entropy ↑ uncertainty
- VC of halfplane is 3
- XX^T columns are eigenvectors
- cross entropy instead of MSE for sigmoid
- pruning has better test acc than stop early