Lecture 1: Intro, Classification, training

1/19

40% 7 Homeworks 5 slip days

20% Midterm

CS 189: Core material

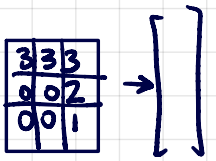
- Find patterns in data, use it to make predictions
- Models and statistics help us understand patterns
- Optimization algo "learn" the patterns

Classification

- Collect training data: reliable debtors & defaulted debtors
- Evaluate new applicants (predictions)

Classifying Digits

7 7 7
1 1 1



Images are points in 9 dimensional space. Hyperplane

Testing and Validation

- Train a classifier: it learns to distinguish 7 from not 7
- Test the classifier on NEW images

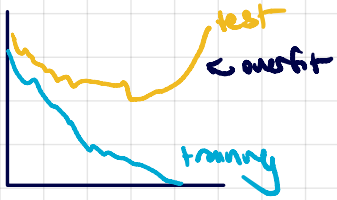
2 kinds of errors:

- Training set error: fraction of training images not classified correctly
- Test Set Error: fraction of misclassified NEW images, not seen in training

Outliers: points whose labels are atypical (eg. solvent borrowers who defaulted)

Overfitting: when test error deteriorates bc classifier becomes too sensitive to outliers

Most ML models have hyperparameters that control over/underfitting, k in k-nearest



Select them by validation

- Hold back a subset of the labeled data, validation set
- Train classifier multiple times w/ diff hyperparameters
- Choose setting that works best on validation

Training set — learn model weights

Validation set — used to tune hyperparameters, choose

test set — used as FINAL evaluation, run ONCE

- public available during competition
- private set labels known only to Kaggle

Techniques:

Supervised learning: is given labeled data

- Classification: is email spam
- Regression: how likely does this patient have cancer

Unsupervised learning: no labels

- Clustering: which DNA sequence similar to one another
- Dimensionality reduction: what are common features of faces? diff?

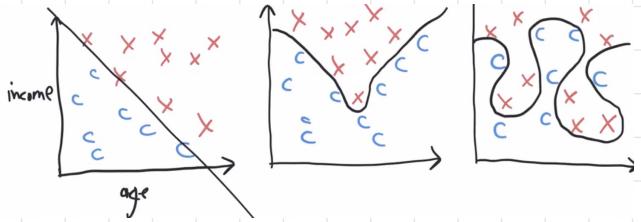
Lecture 2: Classifiers

1/24

Given sample of n observations, each w/ d features (predictors)

Some observations belong to class C; some do not

Ex) Observations are bank loans
Features: income & age (d=2)
Some in class "defaulted", some not
Goal: Predict future



Represent each observation as a point in d-dimensional space
called sample point / feature vector / independent variables

Decision Boundary: boundary chosen by our classifier to separate item in the class from those not

Overfitting: when sinuous decision boundary fits sample points so well that it doesn't classify future points well

Some (not all) classifiers work by computing

Decision Function: function $f(x)$ that maps sample point x to a scalar s.t.
(Predictor) (Discriminant)

$f(x) \geq 0$ if $x \in$ class C
$f(x) \leq 0$ if $x \notin$ class C

Decision Boundary is $\{x \in \mathbb{R}^d : f(x) = 0\}$

$\{x : f(x) = 0\}$ also called **Isosurface** of f for the isovalue 0

f has other isosurfaces for other isovalues $\{x : f(x) = 1\}$

Linear Classifiers: decision boundary is a line/plane. Linear decision func

Math Review

vectors: $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = [x_1\ x_2\ x_3\ x_4\ x_5]^T$

point in 5D space

Convention:

upper case	roman	= matrix, RV, set	X
lowercase	roman	= vector	x
	Greek	= scalar	α
other scalars			n : # sample pts
			d : = # of features = dimension of sample pts
			i, j, k = indices
			$f(), s()$ function scalar

Inner product (dot product): $x \cdot y, x^T y$

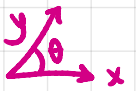
$= x_1 y_1 + x_2 y_2 \cdots x_d y_d$

linear function: $f(x) = w \cdot x + \alpha$

Euclidean norm: $\|x\| = \sqrt{x \cdot x} = \sqrt{x_1^2 + x_2^2 + \ldots + x_d^2}$
length (Euclidean length)

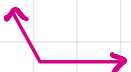
unit vector: $\frac{x}{\|x\|}$
"Normalize" vector x : replace x w/ $\frac{x}{\|x\|}$

Use dot products to compute angles



$$\cos\theta = \frac{x \cdot y}{\|x\| \, \|y\|} = \frac{x}{\|x\|} \cdot \frac{y}{\|y\|}$$

length 1 length 1



linear decision func

$f(x) = w \cdot x + \alpha$

$H = \{x : w \cdot x = -\alpha\}$

set H is hyperplane

Thm: Let x, y be 2 pts lie on H. Then $w \cdot (y - x) = 0$

proof: $w \cdot (y - x) = -\alpha - (-\alpha) = 0$

w is normal vector of H

If w is unit vector, $w \cdot x + \alpha$ is signed distance from x to H
ie. positive on w's side of H; negative on other side

distance from H to origin is α

$\alpha = 0$ iff H passes through origin

coefficients in w, plus α, are weights or parameters regression coefficients

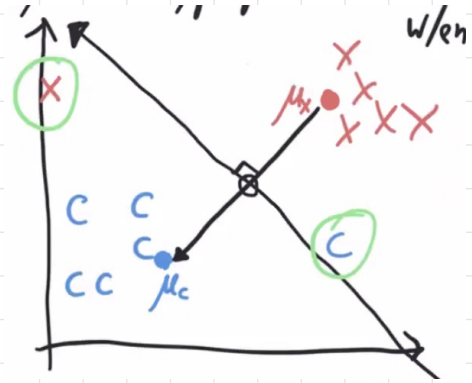
linearly separable: if exists a hyperplane that separates all sample pts in class C from all pts NOT in class C

Simple Classifier

Centroid method: compute mean μ_c in C
mean μ_x not in C

decision function

$$f(x) = \underbrace{(\mu_c - \mu_x) \cdot x}_{\text{normal vector}} - (\mu_c - \mu_x) \cdot \underbrace{\frac{\mu_c + \mu_x}{2}}_{\text{midpoint } \mu_c, \mu_x}$$



decision boundary hyperplane that bisects line segment w/ endpoints μ_c, μ_x

Perceptron Algorithm

slow but correct for linearly separable points

Uses numerical optimization algorithm, gradient descent

n training pts

label $y_i = \{-1$ if $x_i \notin C$

consider only decision passing through origin

Goal: fix weights st

$x_i \cdot w \geq 0$ if $y_i = 1$ and

$x_i \cdot w < 0$ if $y_i = -1$

Equivalently $y_i x_i \cdot w \geq 0$ ← inequality constraint

Idea: nsk function R positive if some constraints are violated. We optimization to choose w that minimizes R

loss function

$$L(z, y_i) = \begin{cases} 0 & \text{if } yz \geq 0 \\ -y_i z & \text{otherwise} \end{cases}$$

classifier prediction, label

If z same sign as y_i loss func is zero
if z has wrong sign, loss func is pos

Define nsk func, obj func, cost fun

$$R(w) = \frac{1}{n} \sum_{i=1}^{n} L(x_i \cdot w, y_i)$$

$$= \frac{1}{n} \sum_{i \in V} -y_i x_i \cdot w$$

V is set of indices i for which $y_i x_i \cdot w < 0$

Classify all $x_1, \ldots x_n$ correctly $R(w) = 0$

R(w) is positive want better w

solve

Find w that minimizes R(w)

Lecture 3

1/26

Perceptron Algorithm

- linear decision fn $f(x) = w \cdot x$
- decision boundary $\{x : f(x) = 0\}$ hyperplane
- find weights w st $y_i x_i \cdot w \geq 0$
- goal revised: find w that min $R(w) = \sum_{i \in V} -y_i x_i \cdot w$

$$\min R(w) = \sum_{i \in V} -y_i x_i \cdot w$$

V set of indices i for which $y_i x_i \cdot w < 0$

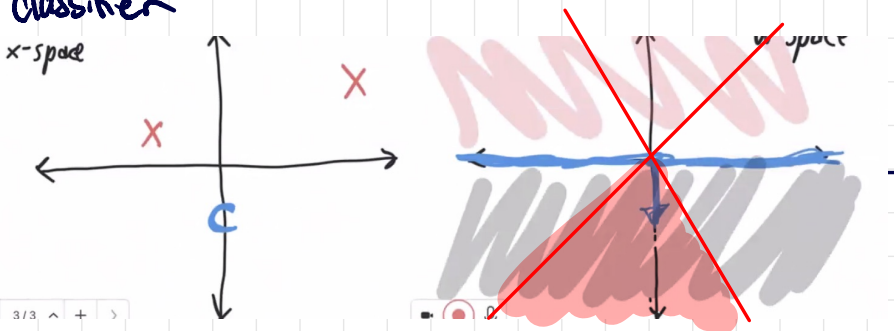
Object in x-space transform to object in w-space

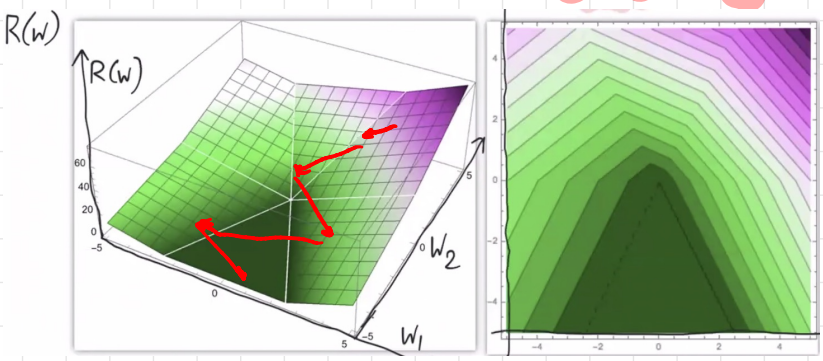
x space	w-space
hyperplane: $\{z : w \cdot z = 0\}$	point: w
point: x	hyperplane: $\{z : x \cdot z = 0\}$

Point x lies on hyperplane $\{z : w \cdot z = 0\} \Leftrightarrow w \cdot x = 0$

point w lies on hyperplane $\{z : x \cdot z = 0\}$ w-space

We want to enforce inequality $x \cdot w \geq 0$, in x space, X same side of classifier





Find w that minimizes

Optimization Algorithm: gradient descent on R

Given starting point w, find gradient of R w/ respect to w

take step in opposite direction

$$\nabla R(w) = \begin{bmatrix} \frac{\partial R}{\partial w_1} \\ \frac{\partial R}{\partial w_2} \\ \vdots \\ \frac{\partial R}{\partial w_d} \end{bmatrix} \quad \text{and} \quad \nabla_w (z \cdot w) = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_d \end{bmatrix} = z$$

$w \leftarrow$ arbitrary nonzero starting point (any $y_i x_i$)

while $R(w) > 0$:

$V \leftarrow$ set of indices i for which $y_i x_i \cdot w < 0$

$w \leftarrow w + \varepsilon \sum_{i \in V} y_i x_i$

return w

$\varepsilon > 0$ is the step size aka learning rate, chosen empirically

Problem: slow takes $O(nd)$ time

Optimization algo 2: stochastic gradient descent

Idea: pick one misclassified x_i

do gradient descent on loss fun $L(x_i \cdot w, y_i)$

perceptron algorithm step takes $O(d)$

while some $y_i x_i \cdot w < 0$

$w \leftarrow w + \varepsilon y_i x_i$

return w

Decision func: $f(x) = w \cdot x + \alpha = [w_1\ w_2\ \alpha] \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$

runtime: $O\left(\frac{r^2}{\gamma^2}\right)$ iterations

Maximum Margin Classifiers

margin of linear classifier is distance from decision boundary to nearest sample pt what if margin as wide as possible

enforce constraints $y_i (w \cdot x_i + \alpha) \geq 1$ for $i \in [1, n]$

if $\|w\| = 1$ margin $\min_i \frac{1}{\|w\|} |w \cdot x_i + \alpha| \geq \frac{1}{\|w\|}$

$\min \|w\|$

find w and α that $\min \|w\|^2$

s.t. $y_i (x_i \cdot w + \alpha) \geq 1 \quad \forall\ i \in [1, n]$

quadratic prog in $d+1$ dim n constraints

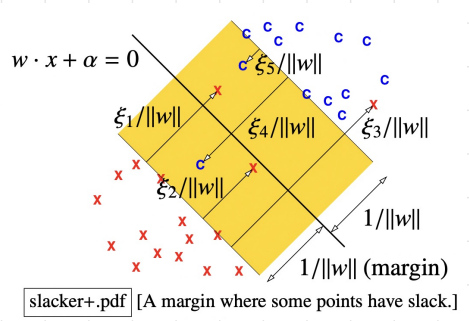
Lecture 4 Notes

Soft Margin Support Vector Machines

- Allow some points to violate the margin w/ slack variables
- Modified constraint for point i

$$y_i (x_i \cdot w + \alpha) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Define margin $1/\|w\|$



slacker+.pdf [A margin where some points have slack.]

Optimization Problem

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

st $y_i(x_i \cdot w + \alpha) \geq 1 - \xi_i$, $i \in [1,n]$

$\xi_i \geq 0$, $i \in [1,n]$

	small C	big C
desire	max margin $1/\|w\|$	keep slack 0 or small
danger	underfitting	overfitting
outliers	less sensitive	very sensitive

$C > 0$ is scalar regularization hyperparameter

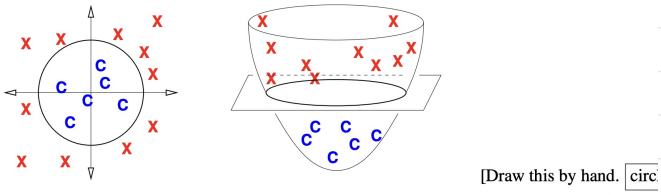
Features

Make nonlinear decision boundaries by lifting points into higher dimensional space

1) Parabolic lifting map

$\Phi: \mathbb{R}^d \to \mathbb{R}^{d+1}$

$$\Phi(x) = \begin{bmatrix} x \\ \|x\|^2 \end{bmatrix}$$



[Draw this by hand. circ

Theorem: $\Phi(X_1), \ldots, \Phi(X_n)$ are linearly separable iff $X_1, \ldots, X_n$ are separable by a hypersphere.

2) Axis aligned ellipsoid/hyperboloid

$\Phi: \mathbb{R}^d \to \mathbb{R}^{2d}$

$\Phi(x) = [x_1^2 \ \ldots \ x_d^2 \ x_1 \ \ldots \ x_d]^T$

3) Ellipsoid / hyperboloid

$\Phi(x): \mathbb{R}^d \to \mathbb{R}^{(d^2+3d)/2}$

$Ax_1^2 + Bx_2^2 + \ldots + Dx_1x_2 + \ldots + Gx_1 \ldots + \alpha = 0$

Iso surface is a quadric

4) degree-p polynomial

Ex) cubic in $\mathbb{R}^2$

$\Phi(x): \mathbb{R}^d \to \mathbb{R}^{O(d^p)}$

$\Phi(x) = [x_1^3 \ \ x_1^2x_2 \ \ x_1x_2^2 \ \ x_2^3 \ \ x_1^2 \ \ x_1x_2 \ \ x_2^2 \ \ x_1 \ \ x_2]^T$

Lecture 5: ML Abstractions + Numerical operations

1) Unconstrained Optimization

Given: obj fn $f(x)$, smooth if

Goal: find x^* that min f

f has: global min/max
- local min/max

f is convex if for every $x, y \in \mathbb{R}^d$ a line segment connecting $(x, f(x))$ to $(y, f(y))$ does not go below $f(\cdot)$

Optimization Algorithms

Application / Data Classify? Regress
Model - features - decision fns - decision
Optimization Model - model → something can train
Optimization Algorithms - gradient descent, - simplex

continous convex function has:
1) no min (to $-\infty$)
2) one unique global min
3) connected set of local minima w/ equal $f(\cdot)$

Algo for general smooth f:

- Gradient descent
  - blind (w/ learning rate) $w \leftarrow w - \varepsilon \nabla f(w)$
  - stochastic blind
  - w/ line search
- Newton's method (requires Hessian matrix)
- Nonlinear conjugate gradient

Algos for non smooth f
- gradient descent
- BFGS

line search
Project down to 1D and use result to inform your step
↳ easy to use secant method, Newton's method, direct search

2) Constrained Optimization (smooth equality constraints)

Given: $f(x)$ and $g(x)=0$, find w that min $f(w)$

↳ use lagrange multipliers to rexpress as unconstrained, goto last sec

3) Linear Program

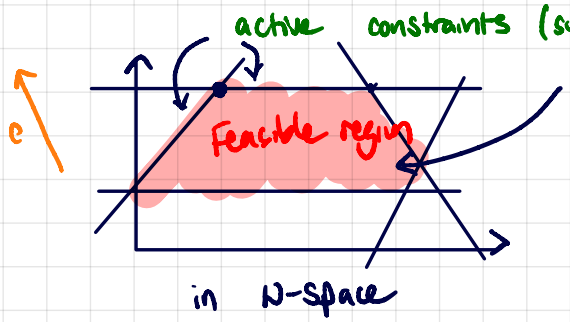
linear obj func + linear inequality constraints

Goal: Find w that max $c \cdot w$ st $Aw \leq b$

n linear constraints

$A_i \cdot w \leq b_i \quad i \in [1, n]$

$A \in \mathbb{R}^{n \times d}$
$b \in \mathbb{R}^n$
$x \in \mathbb{R}^d$



convex if every $p, q \in P$, lies in P

optimum point F furthest in direction of c

- Optima lies on vertices
- Vertex when some of constraints are active

Algs for LP

- Simplex
- interior point methods

4) Quadratic Programming

quadratic, convex obj fn + linear inequality constraints

Obj: Find w min $f(w) = w^T Q w + c^T w$

Constraint: $Aw \leq b$

Q: symmetric, PD matrix $Q \succeq 0$

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

ex) hard soft margin SVM

term $\|w\|^2 \leftrightarrow w^T w \leftrightarrow w^T \mathbb{I} w$

Algs for Quadratic

- Simplex (good in general)
- sequential minimal optimization
- coordinate descent

Lecture 6: Decision Theory

Recall: $P(X) = P(X|Y=1)P(Y=1) + P(X|Y=-1)P(Y=-1)$

Bayes' Theorem: $P(Y=1|X) = \frac{P(X|Y=1)P(Y=1)}{P(X)}$ ← prior prob

posterior prob

Loss function $L(z,y)$ specifies badness if predicts z, true class y

Ex) asymmetrical $L(z,y) = \begin{cases} 0 & \text{if } z=y \\ 1 & \text{if } z=1, y=-1 \\ 5 & \text{if } z=-1, y=1 \end{cases}$

false positive bad

false negative worse

0-1 loss function: symmetrical

Risk: expected loss over all values of x, y

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

$$= P(Y=1)\sum_x L(r(x),1)P(X=x|Y=1) + P(Y=-1)\sum L(r(x),-1)P(X=x|Y=-1)$$

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

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

- if L symmetric, choose class with bigger posterior probability

Bayes Risk: optimal risk $R(r^*)$

Continous Distributions

X: pdf

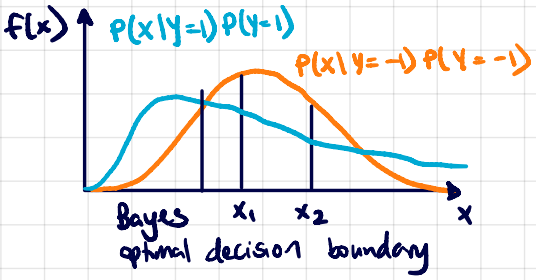
Prob $x \in [x_1, x_2] = \int_{x_1}^{x_2} f(x)\,dx$

Area under curve $= 1 - \int_{-\infty}^{\infty} f(x)\,dx$

Expected value $E[g(x)]$: $\int_{-\infty}^{\infty} g(x)f(x)\,dx$

mean $\mu = E[x] = \int_{-\infty}^{\infty} x f(x)\,dx$

variance $\sigma^2 = E[(x-\mu)^2] = E[x^2] - \mu^2$



For Bayes decision rule, Bayes risk is area under min of functions

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

Ways to build Classifiers

1) Generative models (LDA)
   - guess forms of distributions
   - for class C, fit params, $f(X|Y=C)$, estimate $P(Y=C)$

   ↑ $P(Y|X)$ tells you prob guess is wrong

   ↑ diagnose outliers easily, $P(x)$ small

   ↓ hard to estimate distributions accurately, rely on guessing distribution

2) Discriminative models (logistic regression)
   - Model $P(Y|X)$

   ↑ $P(Y|X)$ tells you prob guess is wrong

3) Find decision boundary (SVM)
   - Model $r(x)$ directly

Lecture 7: Gaussian Discriminant Analysis

Assumption: each class comes from normal distribution

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

$$r^*(x) \quad \max_c Q_c(x) = \ln\left((\sqrt{2\pi})^d f_c(x)\pi_c\right) = -\frac{\|x-\mu_c\|^2}{2\sigma_c^2} - d\ln\sigma_c + \ln\pi_c$$

Quadratic Discriminant Analysis (QDA)

In 2 class ex

$$r^*(x) = \begin{cases} C & \text{if } Q_c(x) - Q_D(x) > 0 \\ D & \text{otherwise} \end{cases}$$

Bayes Decision Boundary at $Q_c(x) - Q_D(x) = 0$

To recover posterior probability

$$P(Y=C|X) = \frac{f(X|Y=C)\pi_c}{f(X|Y=C)\pi_c + f(X|Y=D)\pi_D}$$

$$P(Y=C|X=x) = s(Q_c(x) - Q_D(x)) \quad \text{where} \quad s(r) = \frac{1}{1+e^{-r}}$$

Linear Discriminant Analysis (LDA)

variant of QDA w/ linear boundaries
less likely overfit

All Gaussian have same variance σ

$$\max_c \quad \frac{\mu_c \cdot x}{\sigma^2} - \frac{\|\mu_c\|^2}{2\sigma^2} + \ln\pi_c$$

Decision Boundary: $w \cdot x + \alpha = 0$

Posterior: $P(Y=C|X=x) = s(w \cdot x + \alpha)$

Maximum Likelihood Estimation of Parameters

Maximum Likelihood Estimation (MLE): method of estimating statistical model parameters to maximize (likelihood) func

1) Density Estimation: estimating a PDF from data

Likelihood of Gaussian:
likelihood of generating those points: $\mathcal{L}(\mu, \sigma; x_1, \ldots x_n) = f(x_1)f(x_2)\ldots f(x_n)$
log likelihood ℓ $\max \sum_{i=1}^{n}\left(-\frac{\|x_i-\mu\|^2}{2\sigma^2} - d\ln\sqrt{2\pi} - d\ln\sigma\right)$

$$\nabla_\mu \ell = 0 \qquad \frac{\partial \ell}{\partial \sigma} = 0$$

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

QDA: estimate conditional mean $\hat{\mu}_c$ & conditional variance $\hat{\sigma}_c^2$ of class C separately

LDA: same means & priors; one variance for all classes

$$\hat{\sigma}^2 = \frac{1}{dn}\sum_{c}\sum_{i:y=c} \|x_i - \hat{\mu}_c\|^2$$

pooled within class variance

Lecture 8: Eigenvectors & Multivariate Normal Distr 2/14

Eigenvectors

For square matrix A, if $Av = \lambda v$ for $v \neq 0$, λ

v is eigenvector — v points in same direction or opp dir
λ is eigenvalue

Thm: if v is an eigenvector of A w/ eigenvalue λ
v is eigenvector of A^k w/ eigenvalue λ^k

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

Spectral Theorem: every real, symmetric $n \times n$ matrix has real eigenvalues and n eigenvectors that are mutually orthogonal $v_i^T v_j = 0$ $i \neq j$

- stretches along direction w/ eigenvalue 2, shrinks w/ $-1/2$

Symmetric matrix M
- Positive Definite: if $w^T M w > 0$ all $w \neq 0 \Leftrightarrow$ pos eigenvalues
- Positive Semidefinite: if $w^T M w \geq 0$ all $w \Leftrightarrow$ nonnegative eigenvalues
- Indefinite: if pos & neg eigenvalue
- Invertible: no zero eigenvalue

Eigendecomposition: matrix factorization

$$A = V \Lambda V^T = \sum_{i=1}^{n} \lambda_i v_i v_i^T$$

$$\Lambda = \begin{bmatrix} \lambda_1 & \cdots & 0 & 0 \\ \vdots & \lambda_2 & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

V^T rotates to be axis aligned

Anisotropic Gaussians: $X \sim N(\mu, \Sigma)$

$$f(x) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)}$$

Σ $d \times d$ SPD covar
Σ^{-1} $d \times d$ SPD precision matrix

For Gaussian $R \sim N(\mu, \Sigma)$, $\text{Var}(R) = \Sigma$

R_i, R_j independent $\Rightarrow \text{Cov}(R_i, R_j) = 0$

$\text{Cov}(R_i, R_j) = 0$ & multivariate normal dist $\Rightarrow R_i, R_j$ independent

all features pairwise independent $\Rightarrow \text{Var}(R)$ diagonal
$\text{Var}(R)$ is diagonal & joint normal $\Leftrightarrow$ axis-aligned Gaussian

Lecture 9: Anisotropic Gaussians, MLE, QDA, LDA

2/16

Anisotropic Gaussians

Normal PDF: $f(x) = n(q(x))$ $\quad n(q) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-q/2}$, $q(x) = (x-\mu)^T \Sigma^{-1} (x-\mu)$

Maximum Likelihood Estimation for Anisotropic Gaussians

Sample points $x_1 \dots x_n$, classes $y_1 \dots y_n$ best-fit Gaussians

QDA: $\hat{\Sigma}_c = \frac{1}{n_c} \sum_{i:y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T$: conditional covariance for pts in class c

$\hat{\Sigma}_c$ is PSD

n_c: # pts in class c $\quad \hat{\mu}_c$: mean of sample pts c

$\hat{\pi}_c$: $\frac{\text{\# pts in class } c}{\text{total sample pts}}$,

Gaussian PDF

$$Q_c(x) = \ln\left((\sqrt{2\pi})^d f_c(x)\right) \pi_c = -\frac{1}{2}(x-\mu_c)^T \Sigma_c^{-1} (x-\mu_c) - \frac{1}{2}\ln|\Sigma_c| + \ln \pi_c$$

max linear discriminant fn: $\mu_c^T \Sigma^{-1} x - \frac{1}{2}\mu_c^T \Sigma^{-1} \mu_c + \ln \pi_c$

LDA: $\hat{\Sigma} = \frac{1}{n} \sum_c \sum_{i:y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T$ pooled within class covariance matrix

Design Matrix: X $n \times d$ matrix sample pts, row i is sample pt x_i^T

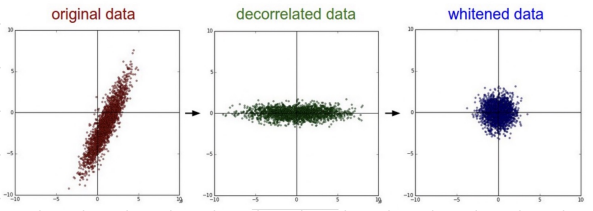
Centering: $\dot{X}$ subtract μ^T from each row of X

Sample covar matrix: $\text{Var}(R)$ $\quad \frac{1}{n}\dot{X}^T\dot{X}$

Decorrelating $\dot{X}$: apply rotation $Z = \dot{X}V$, $\text{Var}(R) = V\Lambda V^T$, $\text{Var}(Z) = \Lambda$

Sphering $\dot{X}$: apply transformation $W = \dot{X}\text{Var}(R)^{-1/2}$

Whitening X: centering + sphering, $X \to W$



Lecture 10: Regression, LS, Linear & Logistic Regression 2/23

Regression: fitting curves to data

Classification: given point x, predict class
Regression: given point x, predict numerical value

- Choose regression func $h(x; p)$ hypothesis h, parameters p
  Decision funcs 1) linear: $h(x; w, \alpha) = w \cdot x + \alpha$
  2) polynomial: added polynomial features
  3) logistic: $h(x; w, \alpha) = s(w \cdot x + \alpha)$ $s(\gamma) = \frac{1}{1+e^{-\gamma}}$

- Choose cost func to optimize
  loss func A) squared loss $L(z,y) = (z-y)^2$
  B) absolute error $L(z,y) = |z-y|$
  C) logistic loss, cross entropy $L(z,y) = -y \ln z - (1-y)\ln(1-z)$ $z \in [0,1]$ $y \in [0,1]$
  cost func a) mean loss $J(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x_i), y)$
  b) max loss $J(h) = \max_{i=1}^{n} L(h(x_i), y)$
  c) weighted sum $J(h) = \sum_{i=1}^{n} w L(h(x_i), y_i)$
  d) L2 penalized/reg $J(h) =$ a, b, or c $+ \lambda \|w\|^2$
  e) L1 penalized/reg $J(h) =$ a, b, or c

Least Squares linear regr: 1+A+a
Weighted least squares linear: 1+A+c
Ridge Regression: 1+A+d
Lasso Regression: 1+A+e
Logistic Regr: 3+C+a
Least Absolute deviations 1+B+a
Chebyshev criterion 1+B+b

} Quadratic cost, min w/ calculus
quadratic program
convex cost, min w/ grad desc
} Linear program

Least Squares Linear Regression

Find w that min $\|Xw - y\|^2$ Residual Sum of Squares (RSS)

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

↑ Easy to compute, linear system, unique stable solution
↓ sensitive to outliers, errors squared
↓ fails if $X^T X$ is singular

Logistic Regression

Fits probabilities in range $(0,1)$

Optimization Prob: find w to min

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

$$s(Xw) = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix}$$

compute derivatives if $s_i = s(x_i \cdot w)$

$$s'(x) = \frac{d}{dx} \frac{1}{1+e^{-x}} = s(x)(1-s(x))$$

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

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

Stochastic Gradient descent: $w \leftarrow w + \epsilon (y_i - s(x_i \cdot w)) x_i$

↑ always separates linearly separable pts

Lecture 11: Regression, Newton's Method, ROC Curves

Least-Squares Polynomial Regression

Replace x_i with $\Phi(x_i)$ degree $0 \dots p$

$$\Phi(x_i) = [x_{i1}^2 \quad x_{i1}x_{i2} \quad x_{i2}^2 \quad x_{i1} \quad x_{i2} \quad 1]^T$$

↓ Very Easy to overfit

Weighted Least Squares Regression

Some sample points more trusted than others

Assign weight w, have w_i in $n \times n$ diagonal matrix Ω

Optimization prob: find w min

$$(Xw - y)^T \Omega (Xw - y)$$

Solve in nomal equations $X^T \Omega X w = X^T \Omega y$

Newton's Method

Iterative optimization method for smooth func $J(w)$, often faster than grad desc

At point v, approx $J(w)$ by quadratic func, jump to critical pt

Taylor series about v: $\nabla J(w) = \nabla J(v) + (\nabla^2 J(v))(w - v) + O(\|w - v\|^2)$

critical pt w: $w = v - (\nabla^2 J(v))^{-1} \nabla J(v)$

$\nabla^2 J(v)$: Hessian matrix

pick starting pt w
repeat until convergence
  $e \leftarrow$ solution to linear system $(\nabla^2 J(w))e = -\nabla J(w)$
  $w \leftarrow w + e$

↓ Hessian computation expensive
↑ finds right step length for most, better direction

Logistic Regression

Can use Newton's method to solve

$$\nabla_w^2 J(w) = \sum_{i=1}^{n} s_i(1 - s_i) x_i x_i^T = X^T \Omega X \qquad \Omega = \begin{bmatrix} s_1(1-s_1) & 0 & & 0 \\ 0 & s_2(1-s_2) & \ddots & \\ \vdots & & & \\ 0 & & & s_n(1-s_n) \end{bmatrix}$$

$w \leftarrow 0$
repeat until convergence
  $e \leftarrow$ solution to nomal eq $(X^T \Omega X)e = X^T(y - s)$
  $w \leftarrow w + e$

LDA vs. Logistic Regression

LDA adv	Logistic Adv
- well separated classes, stable ->2 classes, easy - more acc when classes normal	- emphasis on decision boundary - robust to non-Gaussian distributions - naturally fits 0,1 labels (probabilities)

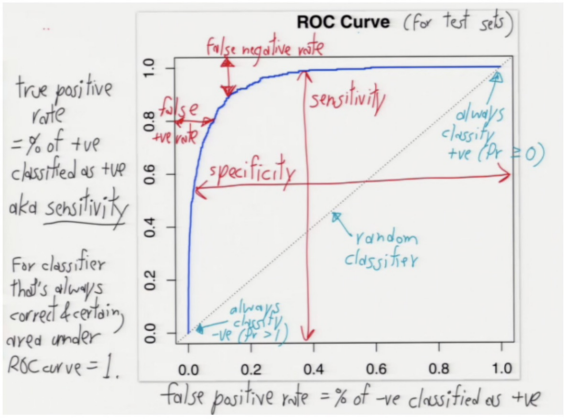
ROC Curves [Reciever Operating characteristics]

rate of false pos vs. true pos

False Negative Rate: vertical distance from curve to top

Specificity: horizontal distance from curve to right

Classifiers effectiveness can roughly be measured by area under the curve



Lecture 12: Statistical Justifications; Bias-Variance Decomposition

3/2

Model of reality:
- sample from unknown dist: $X_i \sim D$
- y-values sum of unknown, func + random noise

$\forall X_i \quad y_i = g(x_i) + \epsilon_i \qquad \epsilon_i \sim D'$ D' mean zero

Regression goal: find h to estimate g

$h(x) = E_y[Y|X=x] = g(x) + E[\epsilon] = g(x)$

Least Squares Regression from Maximum Likelihood

estimate $\epsilon_i \sim N(0, \sigma^2) \qquad y_i \sim N(g(x_i), \sigma^2)$

$$\ell(g; X, y) = -\frac{1}{2\sigma^2} \sum (y_i - g(x_i))^2 - \text{constant}$$

Max likelihood on "parameter" $g \Rightarrow$ estimate g by least-squares regression

Empirical Risk

Risk: expected loss $R(h) = E[L] \qquad x \in \mathbb{R}^d, y \in \mathbb{R}$

Empirical Distribution: discrete uniform distribution over sample pts

Empirical Risk: expected loss under empirical distribution

$$\hat{R}(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(x_i), y_i) \qquad \hat{R}: \text{approx of statistical risk}$$

Empirical risk minimization: h that minimizes $\hat{R}$

Logistic loss from Maximum Likelihood

For probabilities y_i probability pt x_i in class

$h(x_i)$ predicted probability

β duplicate copies x_i, $y_i \beta$ in class, $(1-y_i)\beta$ not

Likelihood: $L(h; X, y) = \prod_{i=1}^{n} h(x_i)^{y_i \beta} (1 - h(x_i))^{(1-y_i)\beta}$

Log likelihood: $\ell(h) = \beta \sum_i (y_i \ln h(x_i) + (1-y) \ln (1 - h(x_i))) = -\beta \sum \text{logistic loss } L(h(x_i), y_i)$

max likelihood $\Rightarrow$ minimize $\sum L(h(x_i), y_i)$

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,

Model: $X_i \sim D, \epsilon_i \sim D', y_i = g(x_i) + \epsilon$, hypothesis h

for some $z \in \mathbb{R}^d$, $Y = g(z) + \epsilon$, $E[Y] = g(z)$, $\text{Var}(Y) = \text{Var}(\epsilon)$

$$R(h) = E[L(h(z), Y)] = E[(h(z) - Y)^2]$$
$$= E[h(z)^2] + E[Y^2] - 2E[Y h(z)]$$
$$= (E[h(z)] - g(z))^2 + \text{Var}(h(z)) + \text{Var}(\epsilon)$$

↑ bias² of method ↑ var of method ↑ irreducible error

Consequences
- 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

Ex] Least Squares Linear Reg

Model: $g(z) = v^T z$

training labels: $y = Xv + e$ unknown v, e

Linear Regression computes w

$$w = X^+ y = X^+(Xv + e) = v + X^+ e$$

Bias: $E[h(z)] - g(z) = E[w^T z] - v^T z = E[z^T X^+ e] = z^T E[X^+] E[e] = 0$

- 0 bias means perfit fit possible

Variance: $\text{Var}(h(z)) = \text{Var}(w^T z) = \text{Var}(z^T v + z^T X^+ e) = \text{Var}(z^T X^+ e) = \sigma^2 z^T (X^T X)^{-1} z$

$$\text{Var}(h(z)) \approx \sigma^2 \frac{d}{n}$$

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

Lecture 13: Shrinkage: Ridge Regression, Subset Selection, and Lasso

3/7

Ridge Regression

Find w to min $\|Xw-y\|^2 + \lambda\|w'\|^2 = J(w)$

- Adding regularization term for shrinkage to encourage small $\|w'\|$
  - guarantees PD normal equations
  - cost function w/ many minima is ill-posed
- Regularization reduces overfitting by reducing variance by penalizing large weights

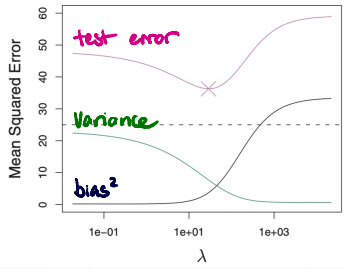
To find solution: $\nabla J=0$, $(X^TX+\lambda I')w = X^Ty$

I': identity w/ 0 in bottom right

- Solve for w, increasing $\lambda \rightarrow$ more regularization

Ridge Regression Var: $Var\left(z^T(X^TX+\lambda I')^{-1}X^Te\right)$

- normalize features to have same variance



Bayesian Justification for Ridge Reg

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

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

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

$$\Rightarrow \text{Minimize } \|Xw-y\|^2 + \lambda\|w'\|^2$$

Feature Subset Selection

Idea: Identify poorly predictive features, less overfitting, smaller test error

Alg: Best subset selection: Try all 2^d-1 nonempty subsets of features

Heuristic 1: Forward Step wise Selection $O(d^2)$ instead of $O(2^d)$

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

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

1) Start w/ d features
2) remove who gives best reduction in validation error

Lasso

least absolute shrinkage and selection operator

Find w that min $\|Xw-y\|^2+\lambda\|w\|_1$ where $\|w\|_1=\sum_{i=1}^{d}|w_i|$

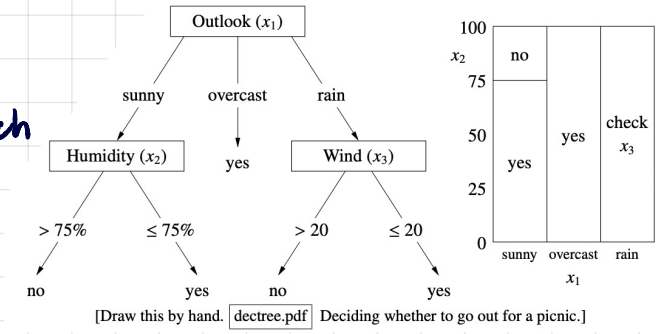
Cross-polytopes: iso surfaces of $\|w\|_1$, convex hull of all pos/neg unit coords

Sometimes sets some weights to zero, especially larger λ

Algs: subgradient descent, least-angle regression (LARS), forward stagewise

Lecture 14: Decision Trees

Nonlinear method for classification and regression



[Draw this by hand. dectree.pdf Deciding whether to go out for a picnic.]

Tree w/ 2 node types:
- internal nodes to test 1 feature value & branch
- leaf nodes specify class h(x)

Greedy algorithm

if leaves are pure (same class C)
  return new leaf
else
  choose best splitting feature j and value β
  $S_l = \{i \in S : X_{ij} < \beta\}$
  $S_r = \{i \in S : X_{ij} \geq \beta\}$
  return new node $(j, \beta, \text{GrowTree}(S_l), \text{GrowTree}(S_r))$

Use entropy to decide split

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

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

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

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

Algs & running time:

Classify test point: walk down tree until leaf, return label $O(\text{tree depth})$

Training: binary $O(d)$ splits, quantitive features $O(n'd)$, running time $O(nd \text{ depth})$

Lecture 15: More Decision Trees, Ensemble Learning, and Random Forests

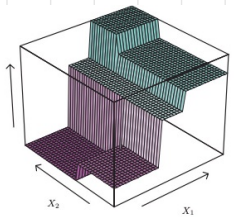
3/14

Decision Tree Variations

Multivariate Split: find non-axis aligned splits or generate randomly (SVM, logistic, GDA)

Decision Tree Regression: creates piecewise constant regression func

$$\text{cost } J(S) = \frac{1}{|S|}\sum_{i \in S}(y_i - \mu_S)^2$$



Stopping Early

- Limit tree depth (for speed), size (big data sets), overfitting
- Stop by: most points same, depth too great, prune

Pruning

Greedily remove each split that improves validation performance, more reliable

- Look at validation in two leaves to remove

Ensemble Learning

Decision Trees not best at predicting, high variance

Weak learner: does better than guessing randomly

Can combine weak learners to get a strong one

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

Random Forests: randomized decision trees on random subsamples

Use learners with low bias

High variance & some overfitting is okay

Averaging reduces variance, sometimes variance

Bagging (Bootstrap AGGregatING): alg for diff learner from same dataset

- n-point training sample, subsample n' sampling w/ replacement
- Points chosen j times have greater weight
- Repeat until T learners, Meta learners takes test point, feed into T, return avg

Random Forests: classification: $m \approx \sqrt{d}$ regression: $d/3$

- At tree node, take random sample m of d features, choose best split from m
- disadvantages: slow loses inference, generate s random multivariate splits.

Lecture 16: The Kernel Trick

Kernels

- Weights can be linear combo of sample points
- Can use inner products of $\Phi(x)$

Optimize $w = X^T a = \sum_{i=1}^{n} a_i x_i$

optimize n dual weights a instead of d^p primal weights w

Kernel Ridge Regression

Center X and y so means are zero: $x_i \leftarrow x_i - \mu_x$, $y_i \leftarrow y_i - \mu_y$, $x_{i,d+1} = 1$

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

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

Since $w = X^T a$ is a solution to normal equations and w is linear combination of sample points, a is dual solution

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: $K = XX^T$, $K_{ij} = k(x_i, x_j)$

Dual Ridge Regression Alg:

$\forall i,j$ $K_{ij} \leftarrow k(x_i, x_j)$

Solve $(K + \lambda I) a = y$ for a

For each test pt z

$h(z) \leftarrow \sum_{i=1}^{n} a_i k(K_i, z)$

Kernel Trick (Kernelization)

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

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

Ex) $d = 2, p = 2$ $(x^T z + 1)^2 = x_1^2 z_1^2 + x_2^2 z_2^2 + \dots$

$$= [x_1^2 \;\; x_2^2 \;\; \sqrt{2} x_1 x_2 \dots][z_1^2 \;\; z_2^2 \;\; \dots]^T$$

$$= \Phi(x)^T \Phi(z)$$

Kernel Ridge Regression: $k(x,z) = \Phi(x)^T \Phi(z)$

Kernel Perceptron

$\Phi(X)$ is $n \times D$ matrix rows $\Phi(x_i)^T$

Featurized perceptron algorithm:

$w \leftarrow y_i \Phi(x_i)$

while some $y_i \Phi(x_i) \cdot w < 0$

$w \leftarrow w + \epsilon y_i \Phi(x_i)$

for each test pt z

$h(z) \leftarrow w \cdot \Phi(z)$

Dual perceptron alg:

$a \leftarrow [y_1 \; 0 \; .. \; 0]^T$

$\forall i,j \quad K_{ij} \leftarrow k(x_i, x_j)$

while some $y_i (Ka)_i < 0$

$a_i \leftarrow a_i + \epsilon y_i$

for each test pt z

$h(z) \leftarrow \sum_{j=1}^{n} a_j k(x_j, z)$

Kernel Logistic Regression

Stochastic Gradient Descent Step: $a_i \leftarrow a_i + \epsilon (y_i - s((Ka)_i))$

Batch Gradient Descent: $a \leftarrow a + \epsilon (y - s(Ka))$, $h(z) \leftarrow s\left(\sum_{j=1}^{n} a_j k(x_j, z)\right)$

Gaussian Kernel

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

hypothesis $h(z) = \sum_{j=1}^{n} a_j k(x_j, z)$ is linear combo of Gaussian centered at sample pts

- gives smooth h
- behaves like k-nearest neighbors
- oscillates less than polynomials
- similarity measure: $k(x,z)$, max $z=x$, 0 distance increases

Choose σ by cross-validation,

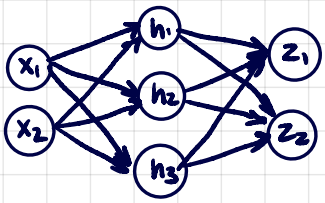
larger $\sigma \rightarrow$ wider Gaussians & smoother h $\rightarrow$ more bias & less variance

Lecture 17: Neural Networks

3/30

can do both classification and 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 $\quad V_i^T$ row i
Layer 2 weights: $k \times (m+1)$ matrix W $\quad W_i^T$ row i

Training

Stochastic or batch grad descent

Loss func $L(z, y)$

prediction ↗ ↖ true labels

Cost func $J(V, W) = \frac{1}{n} \sum_{i=1}^{n} L(z(x_i), y_i)$

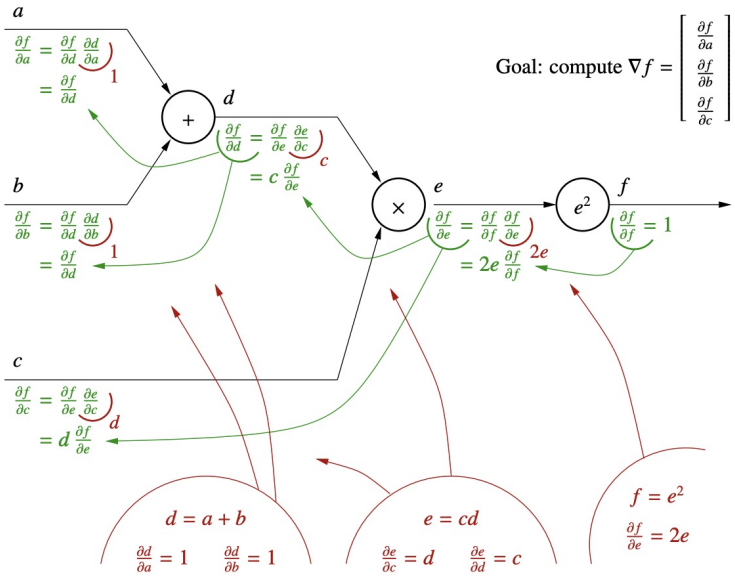
Start w/ random weights

$w \leftarrow w - \epsilon \nabla J(w)$

Naive gradient computation: $O(\text{edges}^2)$ time

Back-propagation: $O(\text{edges})$

Computing Gradients for Arithmetic Expression



Each value z gives partial derivative of the form

$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial n} \frac{\partial n}{\partial z}$ — computed during forward pass; computed during backward pass after forward pass "backpropagation"

where z is an input to n.

Back-propagation

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

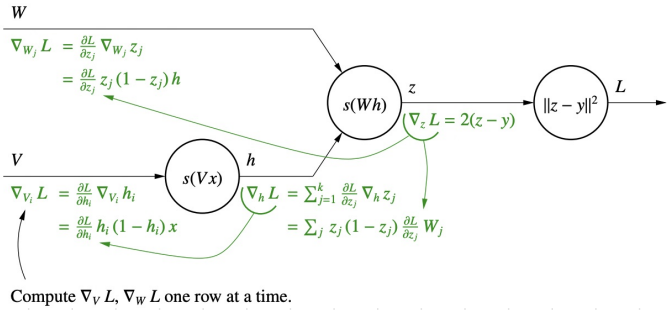
V_i^T is row i of weight matrix V

since $s'(y) = s(y)(1 - s(y))$

$h_i = s(V_i \cdot x) \quad \nabla_{V_i} h_i = s'(V_i \cdot x) x = h_i (1 - h_i) x$

$z_j = s(W_j \cdot h) \quad \nabla_{W_j} z_j = s'(W_j \cdot h) h = z_j (1 - z_j) h$

$\nabla_h z_j = z_j (1 - z_j) W_j$



Compute $\nabla_V L$, $\nabla_W L$ one row at a time.

Lecture 18: Neurobiology; Variations on Neural Networks

Neural Networks inspired by real brains

CPUs: sequentially, nanosecond gates, fragile if gates fail
↑ superior for arithemetic, logical rules, key based memory

Brains: very parallel, millisecond neurons, fault tolerant
↑ superior for vision, speech, associative memory

Neural Networks	Brains
- output of unit - weight of connection	- firing rate of neuron - synapse strength

Brain stem: heart beat, breathing, sleep

Cerebellum: coordination of motor skills

Limbic System: emotion and motivation

Visual Cortex: input from eyes to more useful form

Regression: linear output units - omit sigmoid

Classification: softmax func

Can use ReLU: rectified linear units

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

popular for many-layered networks w/ large training sets

Lecture 19: Better Neural Network Training; Convolutional Neural Nets

4/6

Heuristics for Faster Training

1) Stochastic Gradient Descent: faster than batch on large, redundant data sets
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

Often overparametrized: too many weights, not enough

Convnet Ideas:

1) Local Connectivity: hidden layer to small patch of prev layer
2) Shared weight: hidden units share mask, filter, kernel
   - learn several masks, if one learns edges, all learn edges

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

Lecture 20: Unsupervised Learning and Principal Components Analysis

4/11

Unsupervised Learning

Sample points but no labels, want to discover structure in data

Principal Component Analysis

Goal: Given sample points $\mathbb{R}^d$, find k directions to capture variance

Ex) X be $n \times d$ design matrix, centered $\mu_x = \text{mean}_i X_i = 0$

To get k Gaussian axes of greatest variance, Fit a Gaussian to data w/ MLE

↳ Compute unit eigenvectors/values of X^TX

Choose k best dimensional subspace

Compute k principal coordinates $x \cdot v_i$ of each training point

Ex2) Find direction w that max sample variance of projected data

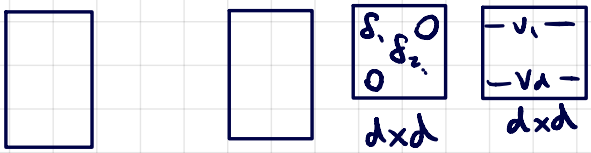
$$\text{Find } w \max \text{Var}([\tilde{x}_1, \tilde{x}_2 \dots \tilde{x}_n]) = \frac{1}{n}\sum_{i=1}^{n}\left(x_i \cdot \frac{w}{\|w\|}\right)^2 = \frac{1}{n}\frac{\|Xw\|^2}{\|w\|^2} = \frac{1}{n}\frac{w^TX^TXw}{w^Tw}$$

Lecture 21: Singular Value Decomposition; Clustering

Every X has SVD $X = UDV^T$ if $n \geq d$

$X = U D V^T = \sum_{i=1}^{d} \delta_i u_i v_i^T$

$\delta_i u_i v_i^T$: rank 1 outer product matrix



$U^T U = I$ ↑ left singular vectors

$V^T V = I$ ↑ right singular vectors

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

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

Find k greatest singular values & vectors in $O(ndk)$

- Row i of UD gives principle coordinates of sample point x_i

Clustering

k-means clustering (Lloyd's Algorithm) Runtime: $O(nk^n)$

Goal: partition n points into k disjoint clusters

Assign each input point x_i a cluster label $y_i \in [1, k]$

$$\min_y \text{ in } \sum_{i=1}^{k} \sum_{y_j = i} \| x_j - \mu_i \|^2$$

k-means heuristic, alternate between

(1) fix labels y_i, update cluster mean μ_i

(2) cluster mean μ_i are fixed, update labels y_i

finds a local min

k-Medoid Clustering

Specify distance func $d(x,y)$, dissimilarity

Use medoid instead of mean, min distance to other pts in cluster

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: $d(A,B) = d(\mu_A, \mu_B)$ μ mean of S

Lecture 22: High Dimensions; Random Projection Psendoinverse; Personality

4/18

Geometry of High Dimensional Spaces

In high dimension, most points are same dist from mean

Pythagoras Theorem: $\|p\|^2 = p_1^2 + p_2^2 + \dots + p_d^2$

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

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

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

Angles between Vectors

$\cos\theta = \frac{p \cdot q}{\|p\|\,\|q\|} = \frac{p_i}{\|p\|}$ $E[\cos\theta] = 0$ $SD(\cos\theta) \approx \frac{1}{\sqrt{d}}$

Random Projection

Project onto random subspace, could preserve dist better

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

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

Pseudoinverse and the SVD

For D diagonal n×d matrix

Psendo inverse D^+ by transposing D and replacing nonzero w/ reciprocal

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

X is n×d matrix $X = UDV^T$ $\text{rank}(D) = \text{rank}(X)$

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 $w = X^+y$

Lecture 23: Learning Theory

Generalization if we want to generalize, constrain hypotheses we allow learner to consider

Range Space (Set System): pair (P, H), P set of all possible test / training

H: hypotheses class, set of hypotheses

Examples:

1) Power set classifiers: P set of k numbers, H power set of P, 2^k subsets of P
2) Linear classifier: $P = \mathbb{R}^d$, 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

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

Dichotomy: X is $X \cap h$, $h \in H$, function that assigns training point to C or not-C

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

Vapnik-Chervonenkis dimension: $VC(H) = \max\{n : \pi_H(n) = 2^n\}$ VC dimension is upper bound on exponent of polynomial

- For generalizations, limit expressiveness of your hypothesis class to limit the num of possible dichotomies

Lecture 24: Boosting; Nearest Neighbor Classification 4/25

AdaBoost

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

Input: design matrix X $n \times d$, labels $y \in \mathbb{R}^n$, $y_i = \pm 1$

- Train T classifiers $G_1 \dots G_T$,
- Weight for sample point X_i in G_T increases depending on how many misclassify
- Train G_t to correctly classify sample pts w/ larger weights
- Metalearner: linear combination of learners, $M(z) = \sum_{t=1}^{T} \beta_t G_t(z)$

$$\min_{G_T, \beta_T} \frac{1}{n} \sum_{i=1}^{n} L(M(x_i), y_i), \quad M(x_i) = \sum_{t=1}^{T} \beta_t G_t(x_i), \quad L(p, l) = e^{-pl} = \begin{cases} e^{-p} & l = +1 \\ e^{p} & l = -1 \end{cases}$$

Separate classified correctly and misclassified

$$n \cdot \text{Risk} = e^{-\beta_T} \sum_{y_i = G_T(x_i)} w_i^{(T)} + e^{\beta_T} \sum_{y \neq G_T(x_i)} w_i^{(T)}$$

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

AdaBoost Alg:
1. Initialize weights $w_i \leftarrow \frac{1}{n}$, $\forall i \in [1, n]$
2. for $t \leftarrow 1$ to T
   a. Train G_t w/ weights w_i
   b. weighted error $err \leftarrow \frac{\sum_{\text{misclassified}} w_i}{\sum_{\text{all}} w_i}$, coeff $\beta_t \leftarrow \frac{1}{2} \ln\left(\frac{1 - err}{err}\right)$
   c. reweight pts: $w_i \leftarrow w_i \cdot \begin{cases} e^{\beta_t} & G_t \text{ misclassifies } x_i \\ e^{-\beta_t} & \text{otherwise} \end{cases} = w_i \cdot \begin{cases} \sqrt{\frac{1-err}{err}} \\ \sqrt{\frac{err}{1-err}} \end{cases}$
3. return metalearner $(h(z) = \text{sign}\left(\sum_{t=1}^{T} \beta_t G_t(z)\right))$

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

as $n \to \infty, k \to \infty, \frac{k}{n} \to 0$ k-NN converges to B

Lecture 25: Nearest Neighbor Algorithms Voronoi Diagrams and k-D Trees

4/27

Nearest Neighbor Algorithms

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

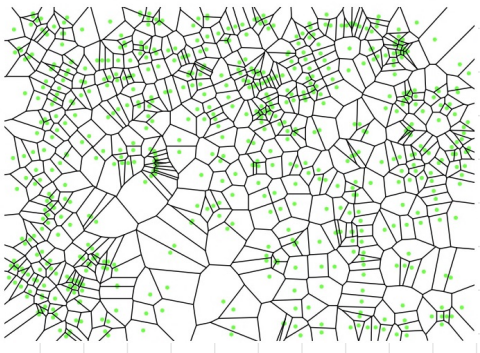
Preprocess training pts,

Voronoi Diagrams: diagrams, pdf of states, exhaustive k-NN, PCA, random projection

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

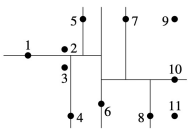
Point location: for query point $q \in \mathbb{R}^d$, find point $w \in X$ where $q \in \text{Vor}\, w$

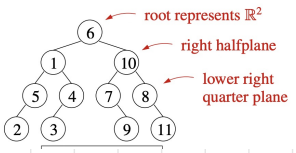
↳ good for 1-NN in 2 or 3 dimensions



k-d Trees: Decision Trees for NN search

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





Goal: given query q, find pt w st

$\|q-w\| \le (1+\varepsilon)\|q-s\|$ s: closest

maintain: nearest neighbor found so far,
binary min-heap of unexplored subtrees, dist from q

1-NN query

$Q \leftarrow$ heap containing root node w/ key zero
$r \leftarrow \infty$
while Q not empty and $(1+\varepsilon)\cdot \text{minkey}(Q) < r$
  $B \leftarrow \text{removemin}(Q)$
  $w \leftarrow B$'s sample pt

$r \leftarrow \min\{r, \text{dist}(q, w)\}$

$B', B'' \leftarrow$ child boxes of B

if $(1+\varepsilon) \cdot \text{dist}(q, B') < r$ then insert $(Q, B', \text{dist}(q, B'))$

if $(1+\varepsilon) \cdot \text{dist}(q, B'') < r$ then insert $(Q, B'', \text{dist}(q, B''))$