CS189 Review - Midterm

Paul Legler

October 16, 2016

Contents

1 Introduction

- **Classification Problems**: the output is a label from a finite set (simplest case is binary)
- **Regression Problems**: the output is real-valued
- **Ranking Problems**: like an internet query for a document relevant to a search term
- **Structured Prediction Problems**: like finding most likely parse tree

1.1 ML Approach to Object Recognition

- **Training time**: compute feature vectors for positive and negative examples of image patches and train a classifier
- **Test time**: compute feature vector on image patch and evaluate the classifier

1.1.1 Classification Methods

- **Nearest neighbor**: Give the label to a new point the same label as its nearest neighbor
- **Linear classifier**: $wx + \beta = 0$ is the decision boundary. w and β are learned through training, x is the new point
- **Non linear classifier**: can be obtained by nearest neighbor, kernel trick, or perception
- If you have a circle where the positive points are withing $X_1^2 + x_2^2 = c$ you can still construct a linear classifier by making the boundary contain $[x_1 x_2 x_1^2 x_1 x_2 x_2^2]^T$
- **Neural Networks**: weight vectors connecting layers of the network. A node's value is $\sum_i x_i w_i$ of all incoming nodes multiplied by the connecting weight vector. The goal is the to make the last year the best classifier of the output. We define a loss function $l(w)$ and compute $\nabla_w l(w)$ and set

$$w_{old} - \alpha \nabla_w l(w)$$

- **Loss Function**

$$L = -\sum_{input} (y_i \ln(O_i) + (1 - y_i) \ln(1 - O_i))$$

- **Activation Function**

$$g(z) = \frac{1}{1 + e^{-z}}$$

- We use sgd to reduce w. With multiple layers we compute with respect to all the weights: from input to hidden to output.
- **The Bayes Classifier**: Is technically the correct classifier but we won't know the probabilities in real life
- Two kinds of error: Training set error and Test set error
- **Over-fitting**: the test set error is much greater than the training set error
- **Validation set**: to avoid over=fitting, we can measure error on a held-out set of the raining data
- **Cross-validation**: dividing the training set into k-fold and using k-1 to train and test on the last fold.
- **Square Error** = Bias^2 + Variance + Irreducible Noise
- **Unsupervised Learning**: Clustering, dimensional reduction, density estimation

2 Linear Classifier

Given data $x_1, x_2, ..., x_n$ that live in R^d where each coordinate of x is a feature// Given properties $y_1, y_2, ..., y_n$ Goal: predict y(x) out of sample x

- We can separate the data with lines and use the distance from the line to make future predictions

2.1 Decision Boundary

- The classifier separates the space in half, the line defines the boundary
- $f(x) = wx + \beta \sum_{i=1}^{n} w_i x_i + \beta + i$ // w ix the wights, B is the bias
  prediction(x) = True if $f(x) \geq 0$
  prediction(x) = False if $f(x) < 0$
  Boundary (B) = $xf(x) = 0 = x : wx + \beta = 0$
- In R^2 the boundary is a line, but in R^d is a hyperplane. The boundary is normal to w: if $x, x' \in B$ then:

$$0 = (wx + B) - (wx' + B) = w(x - x')$$

- Bias (β) is the threshold:
  $predict(x) = True if wx \geq -\beta$
  $predict(x) = False if wx < -\beta$
  The distance from origin to β is $-\frac{\beta}{||w||}$ since w is normal to β and the ray

hits when $w^T(\tau\frac{w}{||w||} + \beta = 0$ solve for τ
The distance from β to x is $\frac{wx+\beta}{||w||}$ as

$$w^T(x + \tau\frac{w}{||w||}) + \beta = 0$$

$$\tau = -\frac{w^T x + \beta}{||w||}$$

2.2 Margin

Goal to maximize the minimize distance to the hyperplane:

$$\min_i \underline{y_i(w^T x + \beta}$$

If we assume the data is linearly separable and the $\beta = 0$ then there is a w s.t. $y_i w x_i > 0$ (can make ¿1 by scaling)
Max Margin:

$$\text{minimize}||w||s.t.y_i(wx_i) \geq 1$$

which we can solve with a QP solver.
If $d > n$, the we can solve:

$$y_i(wx_i) = 1$$

We can also do the following:

$$\text{minimize}||w||^2$$

$$y_i(wx_i) \geq 1$$

If not linearly separable, add slack

$$\text{minimize}||w||^2$$

$$y_i(wx_i) \geq 1 - \epsilon_i$$

where $\epsilon_i \geq 0$ But it is best to set $\epsilon = 1$, $w = 0$ so we need to charge for using ϵ_i

$$\text{minimize}||w||^2 + c\sum \epsilon_i$$

$$y_i(wx_i) \geq 1 - \epsilon_i$$

where $\epsilon_i \geq 0$ This is equivalent to:

$$\text{minimize}c\sum_{i=1}^{n} max(1 - y_i(wx_i), 0) + ||w||^2$$

2.3 Perceptron

$$\text{Test} w^T w_x + \beta$$

$$\text{if} y_i(w^T x_i + \beta) < 0 w = w + y_i x_i, \beta = \beta + y_i$$

The perceptron will always terminate with a correct solution given linearly separable data for any order of updates. It will make no more than $\frac{R^2}{\sigma^2}$ updates where $R = \max_i ||x_i||$ and $\sigma =$ margin

3 Support Vector Machines

$$\min_w C \sum_{i=1}^{n} (1 - y_i(wx_i), 0) + ||w||^2$$

We want to minimize f(w)
w_* is a minimizer if $f(w_*) \leq f(w)$ for all w
w_* is a local minimizer if for some $R \geq 0$ $f(w_*) \leq f(w)$ for all w in $||w - w_*|| \leq R$
f is convex for for all $w_1, w_2, t \in [0, 1]$ we have:

$$f(tw + (1-t)w) \leq tf(w_1) + (1-t)f(w_2)$$

$$loss(w) = c \sum_{i=1}^{n} max(1 - y_i(wx_i), 0) + ||w||^2$$

$$\nabla_w (max(1 - y_i(w^T x_i), 0))$$
$$= -e(y_i(wx_i)) * y_i x_i$$

$$\nabla loss(w) = -c \sum_{i=1}^{n} e(y_i(wx_i)) * y_i x_i + w$$

4 Gradient Descent

$w_{k+1} = w_k - \alpha_k \nabla f(w_k)$
We know that $-\nabla f(w_k)$ is direction of steepest descent:

$$f(w) = f(w_0) + \nabla f(w_0)^T (w - w_0) + 0(||w - w_0||^2)$$

Hence

$$f(w - \alpha \nabla f(w)) = f(w) - \alpha ||\nabla f(w_0)||^2 + 0(\alpha^2 ||\nabla f(w)||^2)$$

We see that for a small enough α:

$$f(w - \alpha \nabla f(w)) \leq f(w)$$

as long as $||\nabla f(w_0)||^2 \neq 0$ We also know that:

$$||\nabla f(w_0)|| = 0 -> \nabla f(w_0) = 0$$

w_* is a local minimum -¿ $\nabla f(w_0) = 0$ because $\nabla f(w_0 0$ is a descent direction
We note that the converse is not true:

$$w_* = 0, f(w) - ||w_*||^2$$

If f is convex, then w_* is a minimizer iff $\nabla f(w_*) = 0$ Proof:

$$t \in [0,1], w\text{arbitrary}$$

$$f(w_* + t(w - w_*)) = f((1-t)w_* + w)$$
$$=\leq (1-t)f(w_*) + tf(w)$$

This implies the following:

$$f(w) \geq f(w_*) + \frac{f(w_* + t(w - w_*) - f(w_*}{t}$$

We take the limit as t goes to 0:

$$f(w) \geq +\nabla f(w_*)^T (w - w_*)$$

$$w_n ew = (1-\alpha)w_o ld + \alpha C \sum_{i=1}^{n} e(y_i(wx_i)) * y_i x_i$$

Complexity = O(u) to compute step

5 Stochastic Gradient Descent

$$loss(w) = \frac{1}{n} \sum_{i=1}^{n} loss_i(w)$$

$$loss_i(w) = (nC)max(1 - y_i(wx_i), 0) + ||w||^2$$

Computing gradient takes O(n) time, but gradient of $loss_i(w)$ takes O(1) time
Main Idea: pick i unit of random from 1,. . . ,n descend along $\nabla loss_i(w)$ rather than $\nabla loss$
Note: $\mathbf{E}[\nabla loss_i(w)] = \nabla loss(w)$ so $\nabla loss_i(w)$ is a noisy version of $\nabla loss$
Note: $\mathbf{E}[\nabla loss_i(w)] = 0$ doesn't mean that $\nabla loss_i(w) = 0$

COME BACK HERE AND WATCH MIDDLE PART OF LECTURE

$$w_n ew = (1-\alpha)w_o ld + (\alpha C n)e(y_i(wx_i))y_i x_i$$

$$f(w) = \sum_{i=1}^{n} (w - y_i)^2$$

$$\nabla f(w) = 2\sum_{i=1}^{n}(w - y_i)$$

$$\nabla f(w) = 0 -> w_* = \frac{1}{n}\sum_{i=1}^{n} y_i$$

5.1 Tricks of the Trade

- Epochs: rather than choose i at random, shuffle the data and run in random order (can speed up)
- Reducing the learning rate: pick step size as big as possible so that you don't diverge and anneal using epoch doubling or exponential decay
- Momentum: if the previous direction was good, then continuing along that direction is good This helps us accelerate in narrow valleys. In the following $\beta = 0.9$ is common

$$w_{k+1} = w_k - \alpha_k g(w_k) + \beta(w_k - w_{k=1})$$

6 ML Abstractions

Data $x_1, \ldots, x_n \in R^d$
Labels $y_1, \ldots, y_n$
If you have a lot of features and $d > n$ you must use pen(w) like $||w||^2$

- in $[0, 1]$ classification
- in $0, 1, \ldots, k-1$ multi-class classification
- in R regression

Model: Function from x space to y space

6.1 Optimization Problem

Minimize: risk + λ model capacity where λ is the regularization parameter
Take the derivative of the cost function and set it equal to 0
Risk: given loss function, want a model such that the loss is small on average

$$\text{Risk} = \text{minimize}_i \mathbf{E}_{x,y}[loss(w, (x, y))]$$

$$= \int loss(w^t x, y) p(x, y)\, dx\, dy$$

The empirical risk is as follows:

$$\approx \frac{1}{n}\sum_{i=1}^{n} loss(w, (x_i, y_i)$$

The empircal risk will approximate for a fixed w with a large n.

6.1.1 Loss Function Types

These 3 all give similar results with tuning:

$$\text{Hinge} = loss(w,(x,y)) = (1 - yw^t x)_+$$

$$\text{Least Squares} = loss(w,(x,y)) = (1 - yw^t x)^2$$

$$\text{Logistic Loss} = loss(w,(x,y)) = -yw^x + log(exp(-w^t x) + exp(w^t x))$$

For **regression**

$$\text{Least Squares} = loss(w,(x,y)) = (w^t x - y)^2$$

$$\text{Least Absolute Value} = loss(w,(x,y)) = |w^t x - y|$$

For **Maximum Likelihood** w is a parameter
Pick the value that maximizes $p(x,y;w)$

$$\text{maximize}_w \prod_{i=1}^{n} p(x,y;w)$$

$$\text{maximize}_w \frac{1}{n} \prod_{i=1}^{n} -\log(p(x,y;w))$$

6.1.2 Empirical Risk Minimization

We want to find a good classifier

$$\text{minimize}_i \sum_{i=1}^{n} loss(w^T x_i, y_i)$$

The true risk $R[w]$:
Generalization Error: $R[w] - R_T[w]$, Thing you want to make small
Training Error: $R_T[w]$, Thing you can make small

$$R[w] = (R[w] - R_T[w]) + R_T[w]$$

6.1.3 Lifting

When lifting from R^d to R^D, you will need to have $O(d^p)$ features where p is the degree of the polynomial

6.1.4 N-Grams

Bag of words: x_i is equal to the number of occurrences of term i where x is some document 2-gram is x_{ij} is the times i and j both appear in the same context (document)

6.1.5 Histograms

$x_{ij} = 1$ if $x_i \in \text{bin}(j)$ and it is 0 otherwise

6.2 Decision Theory

Posterior Probability = $P(y = 1|x) = \frac{P(x|y=1)P(y=1)}{P(x)}$
Class conditional distribution $P(x|y = \pm 1)$
We want to minimize $P(error)$
Expected probability of classification:

$$P(error) = \int P(error|x)P(x)$$

Declare class j, True class k -¿ loss
Want to minimize:

$$\sum_k loss_{k,j} P(c_k|x)$$

6.2.1 Good vs Bad Features

- We want the probability of error to be smaller. So we take the features where the area under the curve (the error area) is minimized
- Good classifiers have an expected loss that is closer to Bayes Risk of the problem, given a certain choice of features
- In the real world you don't know the probabilities and your job is to estimate

6.3 Classifiers

Problem: We have two gaussian distributions(C are classes) $P(x|C_1)$ with mean μ_1 and $P(x|C_2)$ with mean μ_2 where the means are different but they have the same std.dev. $\sigma_1^2 = \sigma_2^2$ Class 1 and 2 have prior probabilities π_1 and π_2

$$P(x|C_1) = \frac{1}{2\sigma\pi} exp(-\frac{(x-\mu_1)^2}{2\sigma^2})$$

Now using Bayes we see that:

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

$$= \frac{\pi_1 exp(-\frac{(x-\mu_1)^2}{2\sigma^2})}{\pi_1 exp(-\frac{(x-\mu_1)^2}{2\sigma^2}) + \pi_2 exp(-\frac{(x-\mu_2)^2}{2\sigma^2})}$$

$$= \frac{1}{1 + \frac{1-\pi_1}{\pi_1} exp(-\frac{1}{2\sigma^2}((x-\mu_2)^2 - (x-\mu_1)^2))}$$

$$= \frac{1}{1 + \frac{1-\pi_1}{\pi_1} exp(-\frac{1}{2\sigma^2}(2(\mu_1 - \mu_2)x + (\mu_2^2 - \mu_1^2))}$$

This leaves us with $P(C_1|x) = \frac{1}{1+exp(-\beta x+\gamma)} = \frac{1}{1+exp(-z)}$
where $z = \beta x + \gamma$
where $\beta = \frac{\mu_1-\mu_2}{\sigma^2}$ and $\gamma = \frac{\mu_2^2-\mu_1^2}{2\sigma^2} + ln(\frac{\pi}{1-\pi})$
We see that this z=0 when $\mu_1 = \mu_2$

7 Random Vectors

$x = [x_1 x_2 ... x_n]^T$
$p(x) > 0 \forall x$ is the probability density, maps from R^n to R
$Pr(x \subset A) = \int_A p(x)\, dx_1\, dx_2 ... dx_n$ where A is a set
f maps from R^n to R^m, then the mean μ_x (nx1) is:

$$\mathbf{E}[f(x)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x)p(x)\, dx_1 ... dx_n$$

where x is a vector
The Covariance Matrix Λ (nxn) is defined as follows:

$$\mathbf{E}[(x-\mu)(x-\mu)^T] = \mathbf{E}[xx^T] - \mu\mu^T$$

$$\text{var}(x_1) = \mathbf{E}[(x_1 - \mu_{x_1})^2]$$

$\Lambda_{ii} = \text{var}(x_i)$

$z = Ax + By + c$
$\mu_z = A\mu_x + B\mu_y + c$
$\Lambda_z = A\Lambda_x A^T + A\Lambda_{xy} B^T + B\Lambda_{yx} A^T + B\Lambda_y B^T$

$z = v^t$ means that $\Lambda_z = \sigma^2 = v^t \Lambda_x v \geq 0$ which means that Λ_x is positive semidefinite

7.1 Positive Definite Matrices

Let A be square and λ is an eigenvalue of A with corresponding eigenvector x
$Ax = \lambda x$ so we know $A^k x = \lambda^k x$

We also let A be invertible so we know that 0 isn't an eigenvalue (iff) as this would mean Ax = 0 for some nonzero x

We let A by symmetric, meaning A=A^T. We know there exists a set where the Spectral Theorem:

nonzero $_1, .., v_n \in R^n s.t. v_i^T v_j = 0$when i and j arent equal and $Av_i = \lambda_i v_i \forall i,$some $y_i \in R$

$AV = DV$ where $D_1 = [\lambda_1 00...0]$ $V^{-1} = V^T$ Diagonlization of A: $A = VDV^T$
Does $\lambda_i \neq \lambda_j \text{when} i \neq j$? NO

A is positive definite is A=A^T is and $\lambda_i > 0$ for all i which is equivalent to $x^T Ax = \lambda ||x||^2 > 0$ for all x
A is positive definite is A=A^T is and $\lambda_i \geq 0$ for all i
If A is p.d. then A^{-1} is positive definite
A, B are p.d. then A+B p.d.

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

A is p.d. then $B^T AB$ is p.s.d
A is p.s.d then $x^t AX$ is convex

7.2 Multivariate Gaussians

$$p(X) = \frac{1}{det(2\pi\Lambda_x)^{.5}} exp(-\frac{1}{2}(x - \mu_x)^T \Lambda_x^{-1}(x - \mu_x))$$

If you have $\mu_X \in R, \Lambda_x$ p.d., then you know p(x) is a density
z = Ax + b is still Gaussian (closed under linear transformations)
The marginal distribution (closed under marginal conditioning) = $p(x_1) = \int\int p(x)\, dx_2\, dx_3 ... dx_n = N(\mu_{x_1}, \Lambda_{11})$
If Λ_x is diagonal and x gaussian, we know that x_i are independent
If x is gaussian you can always rotate with some v such that z=Vx and z has independent columns

8 Maximum Likelihood

Estimation: p(data|model) is hypothesis testing p(data|hypothesis) with an infinite number of hypothesis

$$\text{Likelihood function} = p(\text{data}|\text{model})$$

Prior: p(model)
If we assume prior is uniform, then ML is maximizing maximum a posteriori (after you see the data): p(model|data) (using bayes rule)
Maxmium Likelihood is also Empirical Risk Minimization. To define this you need a loss and want to compute the distance to the true model:

$$\text{loss}(p(\text{data}|\text{model}))$$

$$loss(model) = distance(p(x|model), p(x))$$

where p(x) is the truth
Distance Kullback-Leibler:

$$D_{kl}(p||q) = \int p(x) \log(\frac{p(x)}{q(x)}\, dx)$$

$$= \mathbf{E}_p[\log(p) - \log(q)]$$

This value is always negative, $D_{kl}(p||q)$ is 0 if p=q
ERM on KL:

$$minimize \frac{1}{n} \sum_{i=1}^{n} -\log(p(data|model))$$

If we keep around the prior:

$$maximize \sum_{i=1}^{n} \log(p(x_i|model)) + \log(p(model))$$

8.1 Biased Coin

With n flips and x heads:

$$P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$$

n=10 and x=8, hat means estimate

$$P(X = 8) = \binom{10}{8} p^8 (1-p)^2$$

Maximum Likelihood:

$$0 = \nabla_p P(X = 8) = 360p^7(1-p)^2 - 90p^8(1-p)$$

$$4(1-p) - p = 0$$

so p = 0.8, but what if we do the log likelihood instead

$$\text{Log Likelihood} = \log P(X = 8) = \log(45) + 8\log(p) + 2\log(1-p)$$

Additionally, we know that this log likelihood, ll, is concave, so when we take the derivative we get:

$$\nabla_p ll(p) = \frac{8}{p} - \frac{2}{1-p} = 0$$

$$8 - 10p = 0$$

So we see that p=0.8

8.2 Likelihood of a Gaussian

$x_1, ..., x_n$ are sampled independently from a normal distribtion N(μ, σ^2) The probability of these samples:

$$p(x_i|\mu, \sigma) = \prod_{i=1}^{n} p(x_i|\mu, \sigma)$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(x_i-\mu)^2}{2\sigma^2})$$

The negative log likelihood is as follows:

$$\sum_{i=1}^{n} \log(2\pi\sigma) + \frac{(x_i-\mu_i)^2}{2\sigma^2}$$

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$0 = \sum_{i=1}^{n} \frac{(x_i-\mu_i)^2}{\sigma^3} + \frac{1}{\sigma}$$

$$\hat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_i)^2$$

8.2.1 Multivariate

$$p(x|\mu,\sigma^2) = \frac{1}{det(2\pi\Lambda_x)^{.5}} exp(-\frac{1}{2}(x-\mu_x)^T\Lambda_x^{-1}(x-\mu_x))$$

$$-ll = \sum_{i=1}^{n} \frac{1}{2}(x_i-\mu)^T\Lambda^{-1}(x_i-\mu) + \frac{1}{2}\log(det(2\pi\Lambda))$$

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$0 = \Lambda^{-1}(\sum_{i=1}^{n}(x_i-\mu)(x_i-\mu)^T)\Lambda^{-1} + n\Lambda^{-1}$$

$$\hat{\Lambda} = \frac{1}{n}(x_i-\hat{\mu})(x_i-\hat{\mu})^T$$

9 Classification

- Empirical Risk Minimization:

$$minimize \frac{1}{n}\sum_{i=1}^{n} loss(w;(x_i,y_i)) + \lambda pen(w)$$

- **Generative**: fit a model to the data and use that model to classify, For each class C fit $p(x|y=C)$. We will also estimate p(y=C)
  predictions: pick y such that p(y—x) is maximized: Equivalent to

$$\max_c p(x|y=c)p(y=c)$$

  Model $p(x|C_K), p(C_K)$, obtain$p(C_K|x)$ from Bayes

- Discriminative: fit p(y—x) this is just function fitting

$$maximize \prod_{i=1}^{n} p(y_i|x_i) = maximize \frac{1}{n} \sum_{i=1}^{n} log(y_i|x_i)$$

Model $p(C_k|x)$

9.1 Gaussian Discriminate Analysis

We want to find a good model of p(x—y) in higher dimensions

$$p(x|y = C) = N(\mu_C, \Lambda_C)$$

where I_C = elements in C and n_C are the number of elements in c

$$\mu_C = \frac{1}{n_C} \sum_{i \in I_C} x_i$$

$$\Lambda_C = \frac{1}{n_C} \sum_{i \in I_C} (x_i - \hat{\mu_C})(x_i - \hat{\mu_C})^T$$

$$\pi_C = \frac{n_C}{n}$$

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

9.1.1 Decision Rule

classificatoin = arg $\max_C p(x|y = C)p(y = C)$
in log space:

$$arg \max_C \sum_{i=1}^{n} \frac{1}{2}(x_i - \mu_C)^T \hat{\Lambda^{-1}}(x_i - \mu_C) + \frac{1}{2} \log(det(2\pi\hat{\Lambda})) + \log(\frac{n_C}{n})$$

Quadratic Discriminant Analysis: To do this method you need $n >> d^2$ (much bigger). We are fitting a mixture of gaussians to our data and classifying on which mean you are closer to where $p(x) = \sum_{i=1}^{n} \pi_i N(\mu_i, \Lambda_i)$

$$Q_C(x) = \frac{1}{2}(x_i - \mu_C)^T \hat{\Lambda^{-1}}(x_i - \mu_C) + \frac{1}{2} \log(det(2\pi\hat{\Lambda}))$$

For two classes, the decision boundary is $Q_1(x) = Q_2(x)$
How do we determine our class is correct?

$$p(y = C|x) = \frac{p(x|y = C)\pi_C}{p(x|y = C)\pi_C + p(x|y = \text{not } C)\pi_{\text{not } C}}$$

$$\frac{1}{1 + exp(Q_C - Q_{\text{not } C})}$$

9.1.2 Linear Discriminant Analysis

Method of Centroids: Big assumption: $\Lambda_C = \Lambda$ for all classes To know covariance: you need axes and how much variance on all of them.

$$\hat{\Lambda} = \frac{1}{n}\sum_{k=1}^{k}\sum_{i \in C_k}(x_i - \mu_k)(x_i - \mu_k)^T$$

$$= \frac{1}{n}\sum_{i=1}^{k}(x_i - \mu_{y_i})(x_i - \mu_{y_i})^T$$

Now we see that

$$Q_a - Q_b = -x^t\Lambda^{-1}x - x^T\Lambda^{-1}\mu_a + \mu_a^T\Lambda^{-1}x - \mu_a^T\Lambda^{-1}\mu_a - 2log(\pi_a)$$
$$-(-x^t\Lambda^{-1}x - x^T\Lambda^{-1}\mu_b + \mu_b^T\Lambda^{-1}x - \mu_b^T\Lambda^{-1}\mu_b - 2log(\pi_b))$$
$$= 2(\mu_a - \mu_b)\Lambda^{-1}x - \mu_a^T\Lambda^{-1}\mu_a + \mu_b^T\Lambda^{-1}\mu_b - 2\log(\frac{\pi_a}{\pi_b})$$
$$= 2(\mu_a - \mu_b)\Lambda^{-1}x + (\mu_a + \mu_b)^T\Lambda^{-1}(\mu_a - \mu_b) - 2\log(\frac{\pi_a}{\pi_b})$$
$$= 2(\mu_a - \mu_b)\Lambda^{-1}(x - \frac{1}{2}(\mu_a + \mu_b)) - 2\log(\frac{\pi_a}{\pi_b})$$

We can see that the Decision Boundary has the form:

$$x : w^T x + \beta = 0$$

where $w = \Lambda^{-1}(\mu_a - \mu_b)$ and $\beta = \frac{1}{2}(\mu_a^T\Lambda^{-1}\mu_a - \mu_b^T\Lambda^{-1}\mu_b) - \log(\frac{\pi_a}{\pi_b})$
Assume $\pi_a = \pi_b$ and $\Lambda = \sigma^2 I$ then Λ is spherical the decision boundary is:

$$-\frac{1}{\sigma^2}(\mu_a - \mu_b)^T(x - \frac{1}{2}(\mu_a + \mu_b))$$

If you are on the positive side (b) you will get a positive number, and on the negative side you will get a negative number. You need 2d parameters to estimate this. As soon as the covariance matrix isn't the identity, you need d^2 If Λ isn't spherical: If we relabel all the datapoints $z_i = \Lambda^{1/2}x_i$ (to get $\Lambda^{1/2}$ we diagonalize and take the square root of the diagonal) then we see that:

$$\mathbf{E}[z] = \Lambda^{1/2}\mu_x$$

10 Linear Least Squares

The output y_i are $\in R$
$y = w^t x + \epsilon$ where ϵ is the Gaussian noise with 0 mean and σ squared
$p(y|x)$ is a Guassian with mean $w^t x$ and σ^2

Goal: we want to use MLE to estimate w (also called *theta* the model)
x_i, y_i are

$$p(data|\theta) = \prod_{i=1}^{n} p(y_i|x_i, \theta)$$

Then the we want to maximize the log likelihood function of w:

$$\sum_{i=1}^{n} \log(\frac{1}{\sqrt{2\pi\sigma}}) - \frac{(y - w^T x_i)^2}{2\sigma^2}$$

We want to minimize:

$$\sum_{i=1}^{n} (y_i - x_i^T w_i)^2$$

We make the matrix A (The Design Matrix, nxd) to have its rows be x_i^T
Now we see that we are trying to minimize:

$$||y - Aw||^2 = (y - Aw)^T (y - Aw)$$

The loss function l is:

$$\frac{1}{2}(y - Aw)^T (y - Aw)$$

$$= \frac{1}{2}(y^T y - 2w^T A^T y + w^T A^T A w$$

Now we will compute the gradient with respect to w:

$$\nabla_w l = -A^T y + A^T A w$$

$$H(l) = A^T A$$

We will set the gradient with respect to w equal to 0 and solve:

$$\nabla_w l = -A^T y + A^T A w = 0$$

$$A^T y = A^T A w$$

which leaves us with $w = (A^T A)^{-1} A^T y$

We keep in mind that we are trying to minimize

$$||y - Aw||^2$$

so we want y $\approx$ Aw. We can also think about A as column vectors a_i. We want to construct a linear combination of these column vectors that is the same as y. We know that the error (y-Aw) cannot lie in the column space of A because we could cancel it out if it did - so the error must be perpendicular the a_i

10.1 Variants to Reduce Overfitting

- Ride Regression

$$\text{Minimize}\frac{1}{2}(y - Aw)^T + \lambda||w||^2$$

$$\hat{w} = (X^T X + \lambda I)^{-1} X^T y$$

- Lasso or L1 Regularization

$$\text{Minimize}\frac{1}{2}(y - Aw)^T + \lambda||w||_1$$

We can't take this derivative as the L1 norm is not differentiable, but we can use a variety of techniques which achieve the global minimum. The function is convex and as λ goes to zero this goes to least squares

11 Logistic Regression

$P(x|C_1)$ is $N(\mu_1, \sigma^2)$ and $P(C_1) = \pi_1$ We know that $P(C_1|x) = \frac{1}{1+exp(-z)}$ where $z = \beta x + \gamma$

11.1 Multivariate Case

$$ln\frac{p(C_2|x)}{p(C_1|x)}$$

$$= -\frac{1}{2}(x_i - \mu_2)^T \hat{\Lambda^{-1}}(x_i - \mu_2) + \frac{1}{2}(x_i - \mu_1)^T \hat{\Lambda^{-1}}(x_i - \mu_1) + ln(\frac{\pi_2}{\pi_1})$$

which can be written as $\beta^T x + \alpha$
The log odds (logit) are the following affine (linear + bias) function:

$$ln\frac{p}{1-p} = \beta^T x + \alpha$$

$$\frac{p}{1-p} = e^{\beta^T x + \alpha}$$

$$p + pe^{\beta^T x + \alpha} = e^{\beta^T x + \alpha}$$

$$p = \frac{1}{1 + e^{-(\beta^T x + \alpha)}}$$

We know that $0 \leq p \leq 1$ We use logit to make the range better:

$$logit(p) = ln\frac{p}{1-p} = \beta^T x + \alpha$$

11.2 Maximum Likelihood For Logistic

x here is a dx1 where each element is a feature, y $\in 0, 1$

$$p(Y = 1|x) = \frac{1}{1 + exp(-\beta^T x)}$$

$$\mu(x) = \frac{1}{1 + exp(-\eta(x))}$$

We know that $p(y = 1) = \mu(x)$ and $p(y = 0) = 1 - \mu(x)$ so we write:

$$p(y|x) = \mu(x)^y (1 - \mu(x))^{1-y}$$

Now we will compute the likelihood of independent measurements:

$$p(y_1, .., y_n|x_1, ..., x_n, \theta) = \prod_{i=1}^{n} \mu_i(x_i)^{y_i} (1 - \mu_i(x_i))^{1-y_i}$$

Now we get the log likelihood, which we want to maximize:

$$\sum_{i=1}^{n} y_i \ln(u_i(x_i)) + (1 - y_i) \ln(1 - \mu_i(x_i))$$

We want to write this interms of the parameters θ which are inside β
We know that $\nabla_\beta \mu = \mu(1 - \mu)x$

$$\nabla_\beta ll = \frac{y_i}{\mu_i} \nabla_\beta \mu_i - \frac{1 - y_i}{1 - \mu_i} \nabla_\beta \mu_i$$

$$= (\frac{y_i}{\mu_i} - \frac{1 - y_i}{1 - \mu_i}) \mu_i (1 - \mu_i) x_i$$

$$\sum_{i=1}^{n} (y_i - \mu_i) x_i$$

If we want to increase the likelihood, we take a step in the direction that will increase the likelihood where α is the learning rate for Stochastic Gradient Descent:

$$\beta^{t+1} = \beta^t + \alpha(y_i - \mu_i^t x_i)$$

Examples for which the current probability estimates well $y_i \approx \mu_i \approx 1$ or $y_i \approx \mu_i \approx 0$

12 Bias Variance Tradeoffs

Bias: fitting the data, how far away from the mean is our estimator. Bias is also fitting linear functions to polynomials.
Variance: robustness to changes in the data, over lots of instances how much

does it vary over its own expectation. Variance is also fitting high degree polynomials to the data
Irreducible Error: noisy/incorrect labels
Suppose x is $N(\mu, \sigma^2 I)$ is a d dimensional vector
$\mu_{\hat{ML}} = x$
$\mathbf{E}[\mu_{\hat{ML}} - \mu] = 0$
$\mathbf{E}[||\mu_{\hat{ML}} - \mu||^2] = \mathbf{E}[(\mu_{ML} - \mu)(\mu_{ML} - \mu)^T] = \mathbf{E}[Tr((x-\mu)(x-\mu)^T)] = Tr(\Lambda) = d\sigma^2 \neq 0$
We make a new estimate $\hat{\mu_b} = \alpha x$ for some $0 \leq \alpha \leq 1$
We see that the variance will go down:

$$\mathbf{E}[\hat{\mu_b}] = \alpha\mu$$

$$\mathbf{E}[\hat{\mu_b} - \mu] = (\alpha - 1)\mu$$

$$\mathbf{E}[||\hat{\mu_b} - \mu||^2] = \mathbf{E}[||\hat{\mu_b} - \alpha\mu + \alpha\mu - \mu||^2]$$

$$= \mathbf{E}[||\hat{\mu_b} - \alpha\mu||^2] + 2\mathbf{E}[< \hat{\mu_b}\alpha\mu, (1 - \alpha\mu >] + (1-\alpha)^2||\mu||^2$$

$$\alpha^2\sigma^2 d + (1-\alpha)^2||\mu||^2$$

There is another estimator The James-Stein Estimator:

$$\hat{\mu_{JS}} = (1 - \frac{(d-2^2}{\sigma}\ ||x||^2)x$$

Now we deal with a general estimator $\hat{\mu}$

$$\text{Bias}(\hat{\mu}) = ||\mathbf{E}[\hat{\mu}] - \mu||^2$$

$$\text{Variance}(\hat{\mu}) = ||\mathbf{E}[\hat{\mu}] - \hat{\mu}||^2 = Tr(\Lambda_{\hat{\mu}})$$

$$\mathbf{E}[||\hat{\mu} - \mu||^2] = bias(\hat{\mu}) + var(\hat{\mu})$$

$$= E[||\hat{\mu} - E[\hat{\mu}] - (\mu - E[\hat{\mu}])||^2]$$

$$= E[||\hat{\mu} - E[\hat{\mu}]||^2] - 2E[< \hat{\mu} - E[\hat{\mu}], \mu - E[\hat{\mu}] >] + E[||\mu - E[\hat{\mu}]||^2]$$

The middle term goes to 0, leaving us with the error term being made up of simply the variance and the bias

12.1 Bias Variance trade offs in regression

We generate $x_1, ..., x_n$ through a random process $\in R^d$
$y_i = f(x_i) + \epsilon_i$ where f is unknown, ϵ_i are random and not dependent on x_i
$\mathbf{E}[f_i] = 0$
Estimator: $\hat{y} = h(x)$ Now when we want to fit h(x) to $\hat{y}$
$\mathbf{E}[(\hat{y} - y)^2]$
$= \mathbf{E}[(h(x) - y)^2]$ is expectation with respect to some distribution
$= \mathbf{E}[(h(x) - f(x_i) + \epsilon_i)^2]$
$= \mathbf{E}[(h(x) - f(x_i))^2] + var(\epsilon_i)$
$= E[(E[h(x)] - f(x))^2] + E[((h(x) - E[h(x)])^2] + var(\epsilon_i)$
Which is the bias + variance + irreducible error

12.2 More Practical Version

Minimizing over a special kind of functions in some class (linear, etc.) $S = X_1, ..., X_n$ This gives us the best function in the class, with respect to the training data

$$\text{Empirical Risk Minimizer} = \min_{f \in \text{class}} \sum_{i=1}^{n} \text{loss}(f(x_i), y_i) \to f_s$$

We can't actually compute the following. It gives us the best function in the class, with respect to the expected value (over everything)

$$\text{Empirical Risk Minimizer} = \min_{f \in \text{class}} \mathbf{E}[loss(f(x_i), y_i)] \to f_c$$

This gives us the best possible function.

$$\text{Empirical Risk Minimizer} = \min_{f} \mathbf{E}[loss(f(x_i), y_i)] \to f_*$$

12.3 Risk

Both regularization and cross validation can help with variance and bias. The risk of a function is defined as:

$$R[f] = \mathbf{E}[loss(f(x), y)]$$

$$R[f_s] = R[f_s] - R[f_c] + R[f_c] - R[f_*] + R[f_*]$$

$R[f_s] - R[f_c]$ is the variance: error of the function on the training data - the best you could have done knowing the probabilities (no perfect information). We can decrease this if we make the class simpler (generalize)

$R[f_c] - R[f_*]$ is the bias: even if you had the best possible function in the class, you wouldn't get the best function. We can make this smaller if we have a more complicated model (closer to f_*)

$R[f_*]$ is the irreducible error, as there will still be errors in the y.

12.4 Newton's Method

Using the second order derivative to model a function:

$$x_{k+1} = x_k - H(f(x_k))^{-1} \nabla(f(x_k))$$

$$H(\frac{1}{n} \sum loss(w^t x_i, y_i)) = X^T \Omega X$$

for a diagonal Ω

$$\nabla(\frac{1}{n} \sum loss(w^t x_i, y_i)) = X^T e$$

for some e

13 Regularization

We want to minimize our empirical loss plus a penalty term
We do this because we can reduce the effect of the noise, reduce overfitting, but it always increases bias

13.1 Ridge Regression

$$minimize_w \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \lambda||w||^2$$

$$minimize_w \frac{1}{n}||Xw - y||^2 + \lambda||w||^2$$

If we assume that $x_i = N(0, \beta I^2)$ and $y_i = w_*^T x_i + \epsilon_i$ and $\epsilon_i = N(0, \sigma^2)$
where (x_0, y_0) is a new sample and LS=Least Squares.

$$w_{LS} = (X^T X)^{-1} X^T (X w_* + \epsilon)$$

$$= w_* + (X^T X)^{-1}\epsilon$$

Which is the real solution plus the pseudoinverse times some noise vector. This means that $\mathbf{E}[w_{LS}\hat{} - w_*] = 0$

$$\text{Bias} = \mathbf{E}[\hat{w_{LS}} x_0 - y_0]$$

$$\text{Bias} = \mathbf{E}[\hat{w_{LS}} x_0 - y_0]$$

$$var((w_{ls} - w_*)^T x) \approx \frac{\sigma^2 d}{n}$$

The ridge regression solution:

$$\hat{w_R} = (X^T X + n\lambda I_d)^{-1} X^T y$$

If n is much less than d you can solve:

$$\hat{w_R} = X^T (X X^T + n\lambda I_n)^{-1} y$$

The Bias of the ridge regression estimate is not 0. If we assume $\frac{1}{n}X^T X \approx \beta^2 I$, we get αw_* where $\alpha < 1$ (shrunk w_*)
The variance is as follows with the same assumptions:

$$var((\hat{w_R} - w_*)^T x) \approx \frac{\sigma^2 d}{n}\frac{n\beta^2}{n\beta^2 + \lambda}$$

If we make λ arbitrarily big, we can make the variance as small as we want.

13.2 Other Penalties

The L1 norm:

$$l_1 = \sum_{i=1}^{d} |x_i| = ||x||_1$$

The biggest reason to use the L1 norm gives sparse w, meaning the solutions to $\frac{1}{n}||Xw - y||^2 + \lambda||w||_1$ are mostly zeroes
Model Selection: sparse w means we can ignore a lot of the elements when coming up with a classifier - it helps give a more understandable solution

Bayes Information Criteria

$$\min_w ||Xw - y||^2 + \lambda nnz(w)$$

Where nnz(w) is the number of non zero elements in w. But this is hard, so instead we use the best approximater (want convex):

$$\min_w (w^T x_i - y_i)^2 + \lambda||w||_1$$

13.3 L1 Shrinkage

$Shrink(X_i, \lambda)$: it will be 0 inside the range, and pushed to be inside the range (towards the origin) when outside it

$$minimize_{\hat{X}} \frac{1}{2}||\hat{X} - X||^2 + \lambda||\hat{X}||_1$$

$$minimize_{\hat{X}_i} \sum_{i=1}^{d} \frac{1}{2}(\hat{X}_i - X_i)^2 + \lambda|\hat{X}_i|$$

$$\hat{X}_i = X_i - \lambda \text{if } X_i \geq \lambda$$

$$\hat{X}_i = X_i + \lambda \text{if } X_i \leq -\lambda$$

Otherwise ($-\lambda \leq X_i \leq \lambda$) we see $\hat{X}_i = 0$

13.4 Iterative Shrinkage Thresholding

We simply take a gradient step and apply the shrinking function

$$minimize f(w) + \lambda||w||_1$$

$$w_{k=1} = shrink(w_k - \alpha\nabla f(w_k), \alpha\lambda)$$

14 Generalization

This means you do **the same** on new data as on the data you have seen before
We want to minimize the Risk:

$$R[w] = \mathbf{E}[loss(w; (x, y))]$$

We see the training set $(x_1, y_1)...(x_{n_T}, y_{n_T})$ Can compute the empirical risk:

$$R_T[w] = \frac{1}{n_T} \sum_{i=1}^{n_t} loss(w, (x_i, y_i))$$

We know that the true risk is the generalization error plus the empirical risk:

$$R[w] = R[w] - R_T[w] + R_T[w]$$

generalization error $= R[w] - R_T[w]$
For some independent (of the data) fixed w:

$$\mathbf{E}[R[w] - R_T[w]] = 0$$

So the empirical risk is an unbiased estimate of risk.

$$var(R_T[w] - R[w]) = \mathbf{E}[(\frac{1}{n_T} \sum_{i=1}^{n_t} loss(w, (x_i, y_i)) - R[w])^2]$$

$$= \frac{1}{n_T} \sum_{i=1}^{n} \mathbf{E}[(loss(w, (x_i, y_i)) - R[w])^2]$$

And we assume $loss(w, (x_i, y_i)) \leq \beta$

$$= \frac{\beta^2}{n_T}$$

There fore we see that: $R[w] \leq R_T[w] + \frac{B}{\sqrt{n}}$ with probability at least 86 percent

14.1 Hoeffding's Inequality

If $x_1, ..., x_n$ are i.i.d random variable with mean μ. Assume that x_i are bounded so that $P(0 \leq x_i \leq \beta) = 1$

$$P(\frac{1}{n} \sum_{i=}^{n} x_i - \mu \geq t) \leq exp(-\frac{2nt^2}{\beta^2})$$

$$P(\mu - \frac{1}{n} \sum_{i=}^{n} x_i \geq t) \leq exp(-\frac{2nt^2}{\beta^2})$$

$R[w] \leq R_T[w] + t$ with prob $exp(\frac{2nt^2}{\beta^2})$

We have $w_1, ..., w_k$ and we used super dumb estimator:

$$\hat{w} = arg \min_{1 \leq i \leq k} R_T[w_i]$$

$R[\hat{w}] \leq R_T[\hat{w}] + \beta\sqrt{\frac{logk}{n_T}}$ with probability at least $1 - \frac{1}{k}$ Using the Union Bound:

$$P(\max_{1 \leq i \leq k} R[w_i] - R_T[w_i] \geq t) \leq kexp(-\frac{2nt^2}{\beta^2})$$

If you have d parameters set

$$w_s = arg \min_{||w|| \leq \beta} R_s[w]$$

$R[w_s] \leq R_s[w_s] + c\sqrt{\frac{d}{n}}$ with $\frac{3}{4}$ probability
Think about checking $w = \frac{\beta}{\sqrt{d}}(\pm 1, \pm 1, ..., \pm 1)$ Either n is bigger than d and the norm is bounded and everything is good. If d is bigger than n you have to use regularization/cross validation!

14.2 Cross Validation

We keep a validation set $(x_1, y_1)...(x_{n_v}, y_{n_v})$ and we have k parameter setting we try on the training set - this gives us k different w's. We don't tune parameters on the validation set, we only evaluate

$$\hat{w} = arg \min_i R_V[w_i]$$

$$R[\hat{w}] \leq R_V[\hat{w}] + \beta\sqrt{\frac{logk}{n_v}}$$

In reality, you must split your training set into a partition with training and validation sets where you put random points in the validation set.