Legend: h_i is the highlight i of a Lecture.

Lec 2. Linear Classifiers

$f(x) \geq 0$ class C

$f(x) < 0$ not class C

$f(x) = w \cdot x + \alpha$ is the decision function

$H = \{x : w \cdot x + \alpha = 0\}$, H is hyperplane, classifies 2 classes. or decision boundary

h1: w is orthorgonal to any line segment that lies on H, why?

Proof: pick 2 points x, y on the H.

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

h2: If w is a unit vector, $f(x) = w \cdot x + \alpha$ is the signed distance from x to H.

Proof. suppose there's a hyperplane $H' = \{x : w \cdot x + \alpha = 2\}$

H' is parallel to H (since w is orthogonal to both H/H',

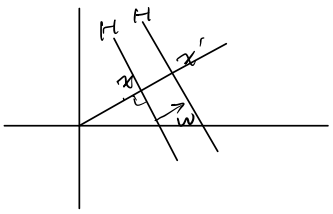
Then, pick a point x, x' that goes across origin,

and in the direction of w, then:

$x'w + \alpha - (xw + \alpha) = x'w - xw =$ the signed distance,

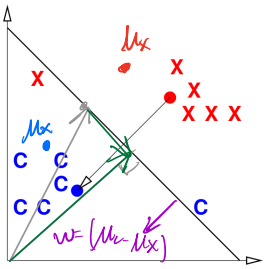
since $xw + \alpha = 0$, so must be the signed distance

from x' to H.



h3: Centroid Method: μ_c and μ_x are the mean of all training points in Class C and Class X, the decision function is:

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



distance $= \frac{\mu_c + \mu_x}{2} + \alpha(\mu_c - \mu_x)$

α = distance in this case since w is opposite.

h4 loss function / Risk function:

Goal: find weights w such that

$$\begin{cases} x_i \cdot w \geq 0 & \text{if } y_i = 1 \\ x_i \cdot w \leq 0 & \text{if } y_i = -1 \end{cases}$$

Equivalently: $y_i x_i \cdot w \geq 0$

Define loss function

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

So we should minimize $L(z, y_i)$

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

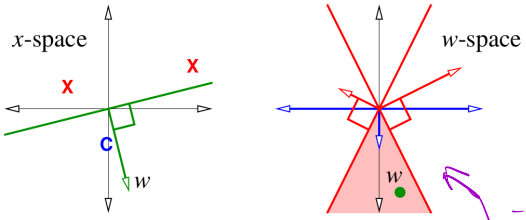
$$= \frac{1}{n} \sum_{i \in V} -y_i x_i \cdot w \quad \text{, where } V \text{ is the set of indices } i \text{ for which } y_i x_i \cdot w < 0$$

after finding best $R(w)$, we find a good classifier

Lec 3. Maximum Margin Classifiers, SVM

h1 If we want to enforce inequality $x \cdot w \geq 0$, that means

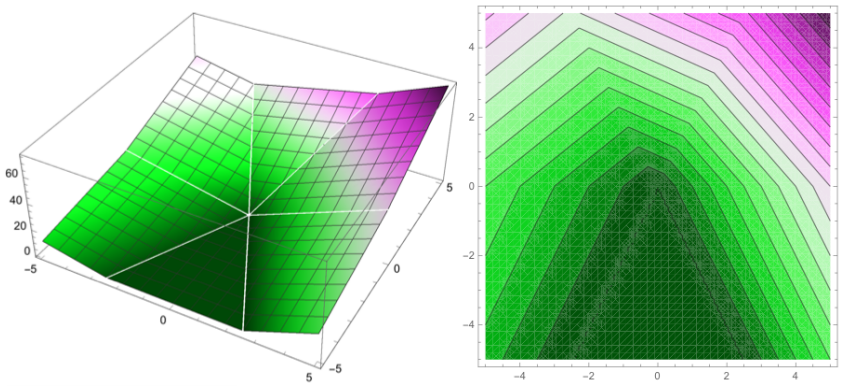
- in x-space, x should be on the same side of $\{z : w \cdot z = 0\}$ as w
- in w-space, w " " " " " " " $\{z : x \cdot z = 0\}$ as x



[Draw this by hand. xwspace.pdf]
[Observe that the x-space sample points are the normal vectors for the w-space lines. We can choose w to be anywhere in the shaded region.]

[For a sample point x in class C, w and x must be on the *same* side of the hyperplane that x transforms into. For a point x not in class C (marked by an X), w and x must be on *opposite* sides of the hyperplane that x transforms into. These rules determine the shaded region above, in which w must lie.]

h2 Gradient descent



riskplot.pdf, riskiso.pdf [Plot & isocontours of risk $R(w)$. Note how R's creases match the lines in the w-space drawn above.]

$$\nabla R(w) = \begin{bmatrix} \frac{\partial R}{\partial w_1} \\ \frac{\partial R}{\partial w_2} \\ \vdots \\ \frac{\partial R}{\partial w_d} \end{bmatrix}$$

(where V is the set of indices i for which $y_i X_i \cdot w < 0$)

$$R(w) = \sum_{i \in V} -y_i X_i \cdot w \quad , \text{ so } \nabla R(w) = -\sum_{i \in V} y_i X_i$$

Given a starting point w, find gradient of R with respect to w, then take a step in the opposite direction.

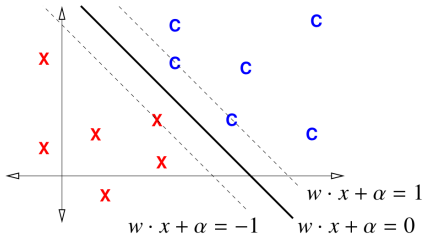
$$R(w)^{(k+1)} = R(w)^{(k)} - \nabla R(w) \cdot \underbrace{\text{step size}}_{\varepsilon}$$

ε: not fixed, $\varepsilon > 0$; aka learning rate

h3 Maximum Margin Classifiers

Margin: Margin of a linear classifier is the distance from the decision boundary to the nearest training point.

Goal: make the margin as wide as possible



We enforce the constraints

$$y_i (w \cdot X_i + \alpha) \geq 1 \quad \text{for } i \in [1, n]$$

The margin is $\frac{w}{\|w\|} \cdot x_i + \frac{\alpha}{\|w\|}$ get unit vector w,

$\frac{w}{\|w\|} \cdot x_i + \frac{\alpha}{\|w\|} = \frac{1}{\|w\|} |w \cdot x_i + \alpha| \geq \frac{1}{\|w\|}$ ($|w \cdot x_i + \alpha| \geq 1$)

$\downarrow$

margin.

To maximize margin, need to minimize $\|w\|$, but $\|w\|$ is not smooth, so we minimize $\|w\|^2$ instead.

> object: minimize $\|w\|^2$
> subject to: $y_i (x_i \cdot w + \alpha) \geq 1$ for all $i \in [1, n]$

This is Quadratic Program in $d+1$ dimensions and n constants, unique one soln! The solution gives us a maximum margin classifier, aka hard-margin support vector machine (SVM)

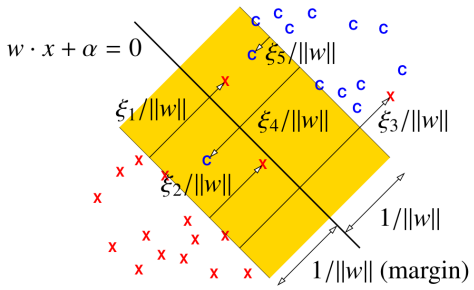
Lec 4 Soft-Margin Support Vector Machines

h1 Soft-Margin SVM.

Allow some points to violate the margin, with slack variables.

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

Then Redefine the margin to be $\frac{1}{\|w\|}$



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

Optimization problem becomes:

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

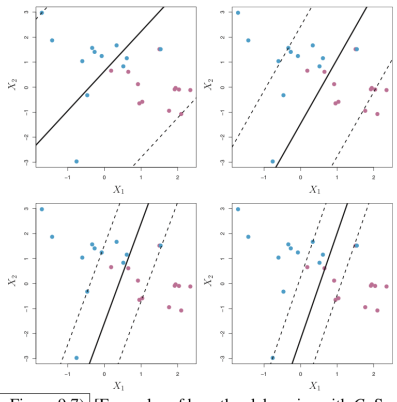
subject to $y_i (x_i \cdot w + \alpha) \geq 1 - \xi_i$ for all $i \in [1, n]$

$\xi_i \geq 0$

$C > 0$ is a scalar regularization hyperparameter that trades off:

	small C	big C
desire	maximize margin $1/\|\|w\|\|$	keep most slack variables zero or small
danger	underfitting (misclassifies much training data)	overfitting (awesome training, awful test)
outliers	less sensitive	very sensitive
boundary	more "flat"	more sinuous

Use validation to choose C.



svmC.pdf (ISL, Figure 9.7) [Examples of how the slab varies with C. Smallest C at upper left; largest C at lower right.]

Lec 12 Nonlinear decision boundaries

FEATURES

Q: How to do nonlinear decision boundaries?

A: Make nonlinear features that lift points into a higher-dimensional space.
High-d linear classifier → low-d nonlinear classifier.

[Features work with all classifiers—not only linear classifiers like perceptrons and SVMs, but also classifiers that are not linear.]

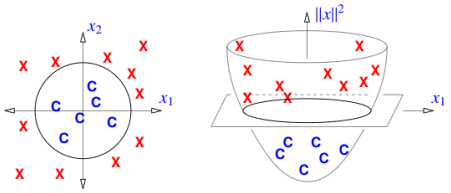
Example 1: The parabolic lifting map

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

$$\Phi(x) = \begin{bmatrix} x \\ \|x\|^2 \end{bmatrix} \quad \leftarrow \text{lifts } x \text{ onto paraboloid } x_{d+1} = \|x\|^2$$

[We've added one new feature, $\|x\|^2$. Even though the new feature is just a function of other input features, it gives our linear classifier more power. Now an SVM can have spheres as decision boundaries.]

Find a linear classifier in Φ-space.
It induces a sphere classifier in x-space.



[Draw this by hand. circledec.pdf]

Theorem: $\Phi(X_1), \ldots, \Phi(X_n)$ are linearly separable iff $X_1, \ldots, X_n$ are separable by a hypersphere.
(Possibly an ∞-radius hypersphere = hyperplane.)

Proof: Consider hypersphere in $\mathbb{R}^d$ w/center c & radius ρ. x is inside iff

$$\|x - c\|^2 < \rho^2$$
$$\|x\|^2 - 2c \cdot x + \|c\|^2 < \rho^2$$
$$\underbrace{[-2c^\top \; 1]}_{\text{normal vector in } \mathbb{R}^{d+1}} \underbrace{\begin{bmatrix} x \\ \|x\|^2 \end{bmatrix}}_{\Phi(x)} < \rho^2 - \|c\|^2$$

Hence points inside sphere ↔ lifted points underneath hyperplane in Φ-space.
[The implication works in both directions.]

h3 n-degree decision func

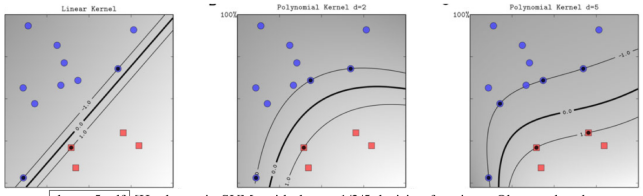
E.g., a cubic in $\mathbb{R}^2$:

$$\Phi(x) = [x_1^3 \quad x_1^2 x_2 \quad x_1 x_2^2 \quad x_2^3 \quad x_1^2 \quad x_1 x_2 \quad x_2^2 \quad x_1 \quad x_2]^\top$$

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

[Now we're really blowing up the number of features! If you have, say, 100 features per sample point and you want to use degree-4 decision functions, then each lifted feature vector has a length of roughly 4 million, and your learning algorithm will take approximately forever to run.]

eg.

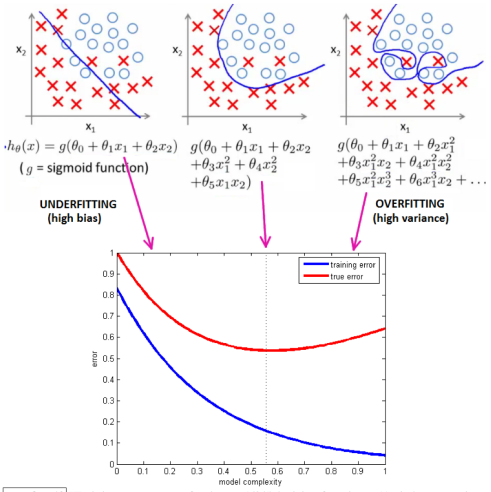


degree5.pdf [Hard-margin SVMs with degree 1/2/5 decision functions. Observe that the margin tends to get wider as the degree increases.]

[Increasing the degree like this accomplishes two things.

- First, the data might become linearly separable when you lift them to a high enough degree, even if the original data are not linearly separable.
- Second, raising the degree can widen the margin, so you might get a more robust decision boundary that generalizes better to test data.

However, if you raise the degree too high, you will overfit the data and then generalization will get worse.]



overfit.pdf [Training vs. test error for degree 1/2/5 decision functions. (Artist's conception; these aren't actual calculations, just hand-drawn guesses. Please send me email if you know where to find figures like this with actual data.) In this example, a degree-2 decision gives the smallest test error.]

[If you're using both polynomial features and a soft-margin SVM, now you have two hyperparameters: the degree and the regularization hyperparameter C. Generally, the optimal C will be different for every polynomial degree, so when you change the degree, you should run validation again to find the best C for that degree.]

Lec 5 ML Abstractions and Numerical Optimization

h1. Abstraction

APPLICATION/DATA
data labeled or not? yes: labels categorical (classification) or quantitative (regression)? no: similarity (clustering) or positioning (dimensionality reduction)?
MODEL [what kinds of hypotheses are permitted?]
e.g.: – decision fns: linear, polynomial, logistic, neural net, . . . – nearest neighbors, decision trees – features – low vs. high capacity (affects overfitting, underfitting, inference)
OPTIMIZATION PROBLEM
– variables, objective fn, constraints e.g., unconstrained, convex program, least squares, PCA
OPTIMIZATION ALGORITHM
e.g., gradient descent, simplex, SVD

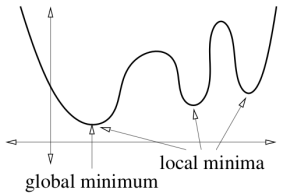
[In this course, we focus primarily on the middle two levels. As a data scientist, you might be given an application, and your challenge is to turn it into an optimization problem that we know how to solve. We will talk about optimization algorithms, but usually data analysts use optimization codes that are faster and more robust than what they would write themselves.]

h2 Optimization problem — Unconstrained

Goal: Find w that minimizes (or maximizes) a continuous objective fn $f(w)$.

f is smooth if its gradient is continuous too.

A global minimum of f is a value w such that $f(w) \le f(v)$ for every v.
A local minimum " " " " " " " " " " "
for every v in a tiny ball centered at w.
[In other words, you cannot walk downhill from w.]

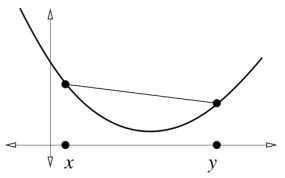


[Drav

Usually, finding a local minimum is easy;
finding the global minimum is hard. [or impossible]

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

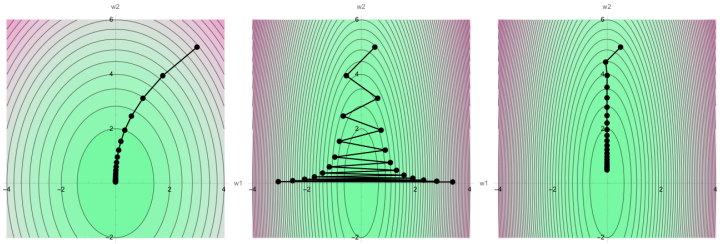
Convex function



[Draw this t

Formally: for every $x, y \in \mathbb{R}^d$ and $\beta \in [0, 1]$, $f(x + \beta(y - x)) \le f(x) + \beta(f(y) - f(x))$.
E.g., perceptron risk fn is convex and nonsmooth.

h3 Gradient Descent



goodcondition.pdf, illcondition105.pdf, illcondition055.pdf [Left: 20 iterations of gradient descent on a well-conditioned quadratic function, $f(w) = 2w_1^2 + w_2^2$, with a modest step size $\epsilon = 0.105$. Center: 20 iterations on an ill-conditioned function, $f(w) = 10w_1^2 + w_2^2$; the same step size is now too large. Right: after reducing the step size to $\epsilon = 0.055$, we have convergence again but we aren't approaching the minimum nearly as quickly.]

h4 Constrained – Linear Program

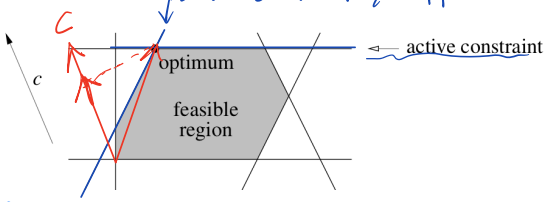
Linear objective fn + linear **inequality** constraints.

Goal: Find w that maximizes (or minimizes) $c \cdot w$ object
subject to $Aw \leq b$

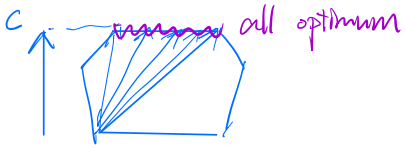
where A is $n \times d$ matrix, $b \in \mathbb{R}^n$, expressing n linear constraints:
$A_i \cdot w \leq b_i, \quad i \in [1, n]$

active constraint / support vector in SVM

in w-space:



If c is vertical



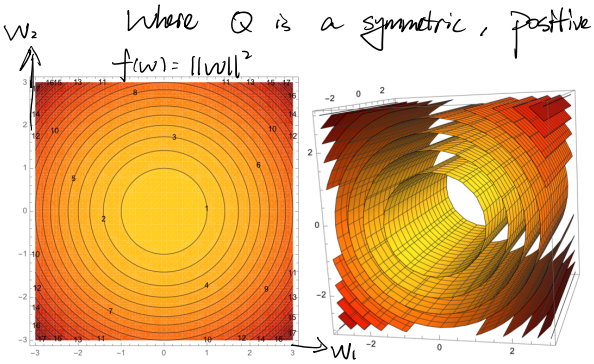
Simplex algorithm: travels from vertex to vertex in the feasible region, until find the optimum.

h5 Quadratic Program

Object: $\min_w \; w^T Q w + c^T w$

Subject to: $Aw \leq b$

where Q is a symmetric, positive definite matrix (PSD)



Lec 6 Decision Theory

Sometimes, two training points at the same location may cover two classes.
Then it's impossible to draw a decision boundary that classifies all points with 100% accuracy.
So,

Suppose 10% of population has cancer, 90% doesn't.
Probability distributions for occupation conditioned on cancer, $P(X|Y)$:

job	(X)	miner	farmer	other
cancer	$(Y = 1)$	20%	50%	30%
no cancer	$(Y = -1)$	1%	10%	89%

[caps here mean random variables, not matrices.]

You meet a farmer. Guess whether he has cancer?

Bayes' Theorem:

↓ posterior probability ↓ prior prob. ↓ if $X =$ farmer

$$P(Y = 1|X) = \frac{P(X|Y = 1)P(Y = 1)}{P(X)} = \frac{0.05}{0.14}$$

$$P(Y = -1|X) = \frac{P(X|Y = -1)P(Y = -1)}{P(X)} = \frac{0.09}{0.14}$$

[These two probs always sum to 1.]

$P(\text{cancer} \mid \text{farmer}) = 5/14 \approx 36\%$.

[BUT . . . we're assuming that we want to maximize the chance of a correct prediction. But that's not always the right assumption. If you're developing a cheap screening test for cancer, you'd rather have more false positives and fewer false negatives. A false negative might mean somebody misses an early diagnosis and dies of a cancer that could have been treated if caught early. A false positive just means that you spend more money on more accurate tests. When there's an asymmetry between the awfulness of false positives and false negatives, we can quantify that with a loss function.]

4.1 asymmetric & symmetric loss function

A loss function $L(z, y)$ specifies badness if classifier predicts z, true class is y.

E.g., $L(z, y) = \begin{cases} 1 & \text{if } z = 1, y = -1, \quad \text{false positive is bad} \\ 5 & \text{if } z = -1, y = 1, \quad \text{false negative is BAAAAAD} \\ 0 & \text{if } z = y. \quad \text{[loss should } \textit{always} \text{ be zero for a perfectly correct prediction!]} \end{cases}$

A 36% probability of loss 5 is worse than a 64% prob. of loss 1,
so we recommend further cancer screening.

It means: False Positive: If he has cancer, but we diagnose he doesn't,
Then $r = L(-1, 1) \times P(y=1) = 5 \times 0.36 = 1.8$
False Negative: If he doesn't have cancer, but we diagnose he has,
Then $r = L(1, -1) \times P(y=-1) = 1 \times 0.64 = 0.64$.
$0.64 < 1.8$, so we'd better tell him you're diagnosed HAVING Cancer.

[A symmetrical loss is the same for false positives and false negatives. For example . . .]

The 0-1 loss function is $L(z, y) = \begin{cases} 1 & \text{if } z \neq y, \quad \text{[always 1 for a wrong prediction]} \\ 0 & \text{if } z = y. \quad \text{[always 0 for a correct prediction]} \end{cases}$

h2 Risk

Let $r : \mathbb{R}^d \to \pm 1$ be a decision rule, aka classifier:
a fn that maps a feature vector x to 1 ("in class") or -1 ("not in class").

$r(x) = \Sigma$ above

The risk for r is the expected loss over all values of x, y: [Memorize this definition!]

$$\begin{aligned} R(r) &= \mathrm{E}[L(r(X), Y)] \\ &= \sum_x \left[L(r(x), 1)\, P(Y = 1 | X = x) + L(r(x), -1)\, P(Y = -1 | X = x) \right] P(X = x) \\ &= P(Y = 1) \sum_x L(r(x), 1)\, P(X = x | Y = 1) + P(Y = -1) \sum_x L(r(x), -1)\, P(X = x | Y = -1) \end{aligned}$$

To be more convinient to find the optimum of R(r), we use $r^*(x)$.

The Bayes decision rule aka Bayes classifier is the fn r^* that minimizes functional $R(r)$.
Assuming $L(1, 1) = L(-1, -1) = 0$,

func1(func2)

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

When L is symmetrical, [the big, key principle you should memorize is]
☆ **pick the class with the biggest posterior probability.**

[But if the loss function is asymmetrical, then you must weight the posteriors with the losses.]
In cancer example, r^*(miner) = 1, r^*(farmer) = 1, and r^*(other) = -1.

☆☆☆

Ref here → $r^*(\text{miner})$: if $L(-1,1)\, P(Y=1|X=x) > L(1,-1)\, P(Y=-1|X=x)$ = $(5 \times 0.2 > 1 \times 0.8)$ ✓
otherwise. ×

So $r^*(\text{miner}) = 1$

Same for $r^*(\text{farmer}) = 1$, $r^*(\text{other}) = -1$

$R(r^*) = 0.1(5 \times 0.3) + 0.9(1 \times 0.01 + 1 \times 0.1) = 0.249$ No decision rule gives a lower risk.

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

$$= 0.1 \cdot 5 \cdot 0.3 + 0.9 \cdot (1 \cdot 0.01 + 1 \cdot 0.1)$$

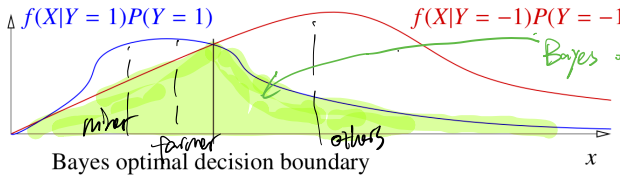
$$= 0.249$$ ← minimized risk for this model

which is give miners false negative, farmers false negative, and false positive for others.

h3 Continuous Probability

This graph is for symmetric loss func. for asymmetric, scale by $L(z,y)$

Suppose $P(Y = 1) = 1/3$, $P(Y = -1) = 2/3$, 0-1 loss.



Bayes optimal decision boundary

Draw this figure by hand (cancerposterior.png)

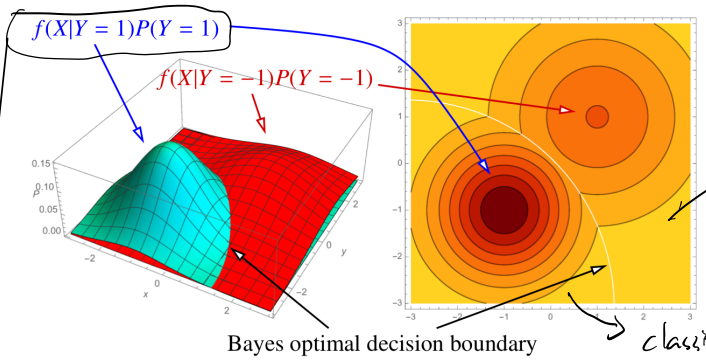
X is job type

Define risk:

$$R(r) = E[L(r(x), y)]$$
$$= P(Y=1) \int L(r(x), 1) f(X=x|Y=1)\,dx + P(Y=-1) \int L(r(x),-1) f(X=x|Y=-1)\,dx$$

Expectation of Bayes decision

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



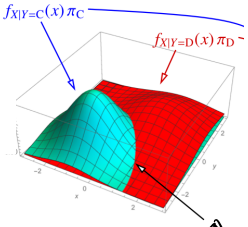
Bayes optimal decision boundary

qda3d.pdf, qdacontour.pdf [Two different views of the same 2D Gaussians.]

Lec 7 Gaussian Discriminant Analysis, QDA, LDA

Fundamental assumption that each class has a normal distribution

$$X \sim N(\mu, \sigma^2) : f(x) = \frac{1}{(\sqrt{2\pi}\sigma)^d} \exp\left(-\frac{\|x-\mu\|^2}{2\sigma^2}\right). \quad [\mu \text{ \& } x = \text{vectors; } \sigma = \text{scalar; } d = \text{dimension}]$$



original PDF

For each class C, suppose known the mean μ_c and σ_c^2, yielding PDF $f_{X|Y=C}(x)$ Y: classes

and a prior $\pi_c = P(Y=C)$, π_c is the same meaning of $P(Y=C)$

$\hookrightarrow f_{X|Y=C}(x)\,\pi_C = f(X|Y=C)\,P(Y=C)$

new Quadratic Func

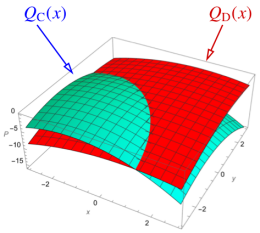
hi

Given x, Bayes decision rule $r^*(x)$ predicts class C that maximizes $f_{X|Y=C}(x)\,\pi_C$.
[Remember our last lecture's main principle: **pick the class with the biggest posterior probability!**]

To find the maximum value of $f_{X|Y=C}(x)\,\pi_C$, we can instead find max value of

$$Q_C(x) = \ln\left(\left(\sqrt{2\pi}^d\right) f_{X|Y=C}(x)\,\pi_C\right)$$

$$= -\frac{\|x-\mu_C\|^2}{2\sigma_C^2} - d\ln\sigma_C + \ln\pi_C$$



h2

Quadratic Discriminant Analysis (QDA)

Suppose only 2 classes C, D. Then the Bayes Classifier is

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

see ref above, or since

$L(C,D)\,Q_D(x) < L(D,C)\,Q_C(x)$

Loss of classify C is smaller

$L(C,D) = L(D,C)$ here.

Bayes Decision Boundary is

$Q_C(x) - Q_D(x) = 0$

To recover posterior probability in 2-class case, use Bayes, and get the Logistic func of when $X=x$ should be classified as $Y=C$.

$$P(Y=C|X) = \frac{P(X|Y=C)\cdot P(Y=C)}{P(X)} = \frac{P_{X|Y=C}(x)\,\pi_C}{P_{X|Y=C}(x)\,\pi_C + P_{X|Y=D}(x)\,\pi_D}$$

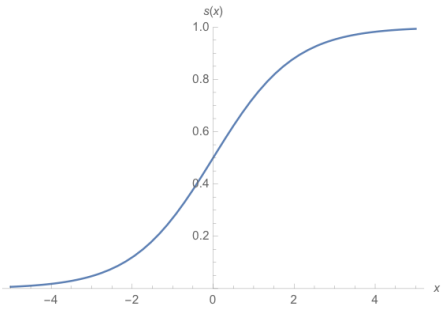
recall $e^{Q_C(x)} = (\sqrt{2\pi})^d f_{X|Y=C}(x)\,\pi_C$ [by definition of Q_C]

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

posterior

$$s(\gamma) = \frac{1}{1+e^{-\gamma}} \quad \Leftarrow \text{logistic fn aka sigmoid fn}$$

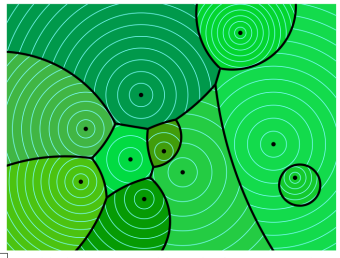
[recall $Q_C - Q_D$ is the decision fn]



logistic.pdf [The logistic function. Write beside it:] $s(0) = \frac{1}{2}$, $s(\infty) \to 1$, $s(-\infty) \to 0$, monotonically increasing.

[We interpret $s(0) = \frac{1}{2}$ as saying that on the decision boundary, there's a 50% chance of class C and a 50% chance of class D.]

Multi-class QDA: [QDA works very naturally with more than 2 classes.]



black points are the peak of each class

multiplicative.pdf [Multi-class QDA partitions the feature space into regions. In two or more dimensions, you typically wind up with multiple decision boundaries that adjoin each other at joints. It looks like a sort of Voronoi diagram. In fact, it's a special kind of Voronoi diagram called a multiplicatively, additively weighted Voronoi diagram.]

h3 Linear Discriminant Analysis (LDA)

Fundamental Assumption: all the Gaussians have same variance σ^2.

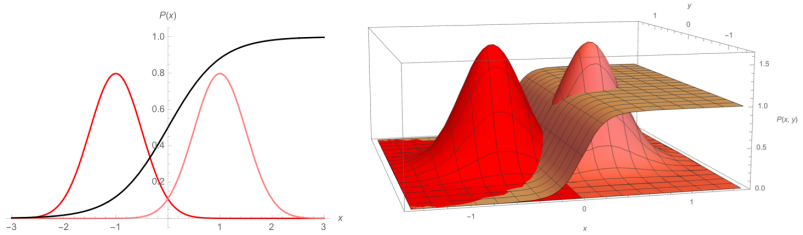
$$Q_C(x) - Q_D(x) = \underbrace{\frac{(\mu_C - \mu_D)\cdot x}{\sigma^2}}_{w\cdot x} \underbrace{- \frac{\|\mu_C\|^2 - \|\mu_D\|^2}{2\sigma^2} + \ln\pi_C - \ln\pi_D}_{+\alpha}$$

It's Linear Classifier.

decision boundary is $w\cdot x + \alpha = 0$

posterior is $P(Y=C \mid X=x) = s(w\cdot x + \alpha)$

[The effect of "$s(w\cdot x+\alpha)$" is to scale and translate the logistic fn in x-space.]



lda1d.pdf, lda2d.pdf [Two Gaussians (red) and the logistic function (black). The logistic function is the right Gaussian divided by the sum of the Gaussians. Observe that even when the Gaussians are 2D, the logistic function still looks 1D.]

need to know parameters: p

necessary for $f(X \mid Y=C)$

h4 Maximum Likelihood Estimation of Parameters

Why we care?

MLE: A method of estimating the parameters of a statistical model by picking parameters that maximize the likelihood function $\mathcal{L}$

like sampling

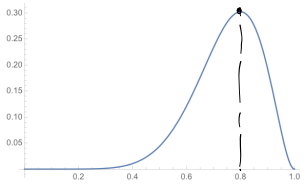
eg: a biased coin, flip 10 times, is head 8 times.

$X \sim B(n, p)$

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

$$P(X=8) = 45\, p^8 (1-p)^2 \overset{\text{def}}{=} \mathcal{L}(p)$$

Find p that maximizes $\mathcal{L}(p)$.



binomlikelihood.pdf [Graph of $\mathcal{L}(p)$ for this example.]

Solve by finding critical point of $\mathcal{L}$:

$$\rightarrow \frac{d\mathcal{L}}{dp} = 360p^7(1-p)^2 - 90p^8(1-p) = 0 \qquad \text{Critical point}$$

$$\Rightarrow \quad 4(1-p) - p = 0 \quad \Rightarrow \quad p = 0.8$$

Suppose our training set has n points, with x points in class C.
Then our estimated prior for class C is $\hat{\pi}_C = \frac{x}{n}$ not true π_C, just estimated π_C.

h5 Likelihood of a Gaussian

Now we transform binomial distribution to Gaussian Distribution

Given $X_1, X_2, \cdots X_n$. known $X \sim$ Normal Distribution (μ, σ^2)

Likelihood of drawing these points is: known $X_1, X_2, \cdots X_n$ points, known they are i.i.d, and are X known normal distribution.

$$\mathcal{L}(\mu, \sigma; X_1, \cdots X_n) = f(X_1) f(X_2) \cdots f(X_n)$$

log on both sides want to know μ, σ^2 of X, estimate $\hat{\mu}$ $\hat{\sigma}^2$

$$\begin{aligned} \ell(\mu, \sigma; X_1, ..., X_n) &= \ln f(X_1) + \ln f(X_2) + ... + \ln f(X_n) \\ &= \sum_{i=1}^{n} \underbrace{\left(-\frac{\|X_i - \mu\|^2}{2\sigma^2} - d \ln \sqrt{2\pi} - d \ln \sigma \right)}_{\text{ln of normal PDF}} \end{aligned}$$

Set $\nabla_\mu \ell = 0, \frac{\partial \ell}{\partial \sigma} = 0$ [Find the critical point of ℓ]

$$\nabla_\mu \ell = \sum_{i=1}^{n} \frac{X_i - \mu}{\sigma^2} = 0 \quad \Rightarrow \quad \hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} X_i$$ [The hats ˆ mean "estimated"]

$$\frac{\partial \ell}{\partial \sigma} = \sum_{i=1}^{n} \frac{\|X_i - \mu\|^2 - d\sigma^2}{\sigma^3} = 0 \quad \Rightarrow \quad \hat{\sigma}^2 = \frac{1}{dn} \sum_{i=1}^{n} \|X_i - \mu\|^2$$

we don't know true μ, so use $\hat{\mu}$ instead

Takeaway: use sample mean & variance of pts in class C to estimate mean & variance of Gaussian for class C.

For QDA: estimate conditional mean $\hat{\mu}_C$ & conditional variance $\hat{\sigma}_C^2$ of **each class C separately** [as above] & estimate the priors:

# of points in training set that is class C

$$\hat{\pi}_C = \frac{n_C}{\sum_D n_D}$$ ⇐ total sample points in all classes → # of all training points

[$\hat{\pi}_C$ is the coin flip parameter]

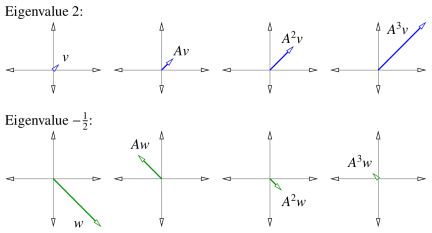
For LDA: same means & priors; one variance for all classes:

$$\hat{\sigma}^2 = \frac{1}{dn} \sum_C \sum_{\{i:y_i=C\}} \|X_i - \hat{\mu}_C\|^2$$ ⇐ pooled within-class variance

[Notice that although LDA is computing one variance for all the data, each sample point contributes with respect to *its own class's mean*. This gives a very different result than if you simply use the global mean! It's usually smaller than the global variance. We say "within-class" because we use each point's distance from its class's mean, but "pooled" because we then pool all the classes together.]

Lec 8 eigenvectors & Multivariate Normal Distribution

Theorem: if v is eigenvector of A w/eigenvalue λ,
then v is eigenvector of A^k w/eigenvalue λ^k



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

Proof: $A^{-1}v = A^{-1}(\frac{1}{\lambda}Av) = \frac{1}{\lambda}v$

hL Spectral Theorem

every real, symmetric nxn matrix has real eigenvalues and n eigenvectors that are mutually orthogonal $v_i^T v_j = 0$ for all $i \neq j$

We can use them as a basis for $\mathbb{R}^n$.

Build Matrix using eigenvectors

Let $V = [v_1\ v_2\ \cdots\ v_n]$ nxn matrix

$V^T V = I$ $\quad \begin{bmatrix} v_1^T v_1 & v_1^T v_j = 0 \\ 0 & \ddots \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \ddots\ 1 \end{bmatrix}$

So $V^T = V^{-1}$

V is orthonormal matrix

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

Then, $AV = V\Lambda$

$AVV^T = V\Lambda V^T$

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

eigendecomposition / matrix factorization

$A^n = V\Lambda V^T V\Lambda V^T V\Lambda V^T \cdots V\Lambda V^T$

$= V\Lambda^n V^T$

Given a symmetric PSD matrix matrix Σ, we can find a symmetric square root $A = \Sigma^{1/2}$, then

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

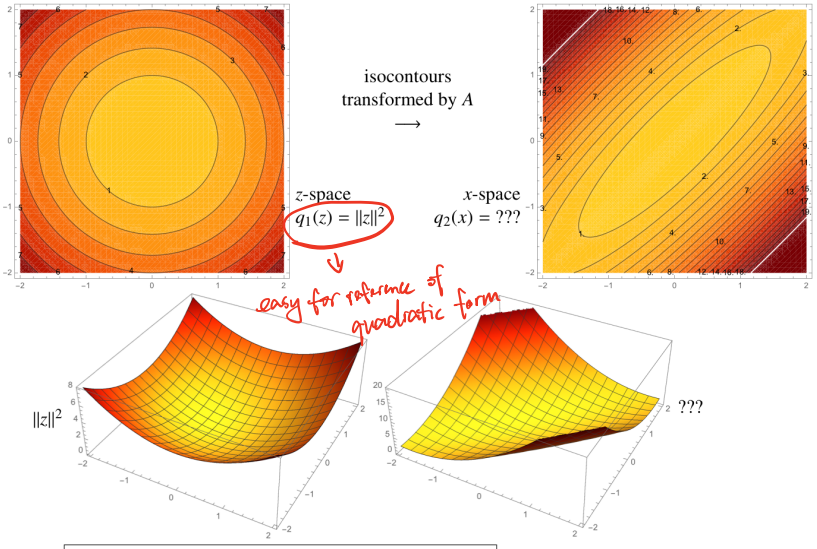
b2 Visualizing Quadratic Form

Quadratic Form of M is $x^T M x$

Why talking about it? ① Introduce a way to graphically understand the effect of a symmetric matrix on the length of a vector

② prerequisite of the next part, Anistropic GAUSSIANS.

Suppose you want a matrix whose quadratic form has the isocontours at right below, which are circles transformed by A. [The same matrix A I've been using, which stretches along the direction with eigenvalue 2 and shrinks along the direction with eigenvalue $-1/2$.]



circles.pdf, ellipses.pdf, circlebowl.pdf, ellipsebowl.pdf

[Both figures at left are plots of $\|z\|^2$, and both figures at right are plots of $x^T A^{-2} x$. (Draw the stretch direction $(1, 1)$ with eigenvalue 2 and the shrink direction $(1, -1)$ with eigenvalue $-\frac{1}{2}$ on the ellipses at right.)]

That is, we want $q_2(Az) = q_1(z)$.
Answer: set $x = Az$.
Then $q_2(x) = q_1(z) = q_1(A^{-1}x) = \|A^{-1}x\|^2 = x^\top A^{-2} x$.

The isocontours of the quadratic form $x^\top A^{-2} x$ are ellipsoids determined by the eigenvectors/values of A.
$\{x : x^\top A^{-2} x = 1\}$ is an ellipsoid with axes $v_1, v_2, \ldots, v_n$ and
radii $\lambda_1, \lambda_2, \ldots, \lambda_n$
because if v_i has length 1 (v_i lies on unit circle), $x = Av_i$ has length λ_i (Av_i lies on the ellipsoid).

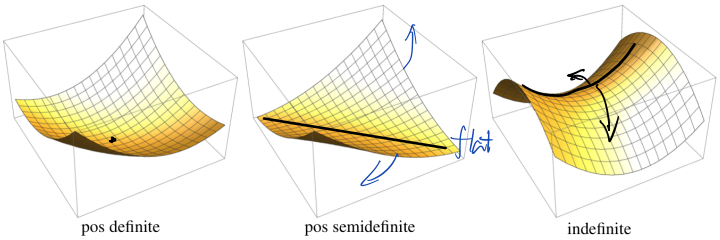
Therefore, contours of $x^\top M x$ are ellipsoids determined by eigenvectors/values of $M^{-1/2}$.
[The eigenvalues of $M^{-1/2}$ are the inverse square roots of the eigenvalues of M.]

Special case: A (or M) is diagonal $\Leftrightarrow$ eigenvectors are coordinate axes
$\Leftrightarrow$ ellipsoids are axis-aligned

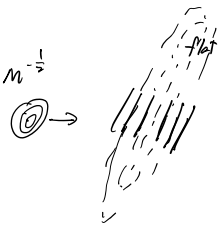
[Draw axis-aligned isocontours for a diagonal metric.]

A symmetric matrix M is

positive definite	if $w^\top M w > 0$ for all $w \neq 0 \Leftrightarrow$ all eigenvalues positive
positive semidefinite	if $w^\top M w \geq 0$ for all $w \Leftrightarrow$ all eigenvalues nonnegative
indefinite	if +ve eigenvalue & −ve eigenvalue
invertible	if no zero eigenvalue



pos definite pos semidefinite indefinite



Lec Anisotropic Gaussians

$X \sim \mathcal{N}(\mu, \Sigma)$,

where μ is $\mathbb{R}^d$ vector,

Σ is covariance $d \times d$ Symmetric Positive Definite Matrix (SPD)

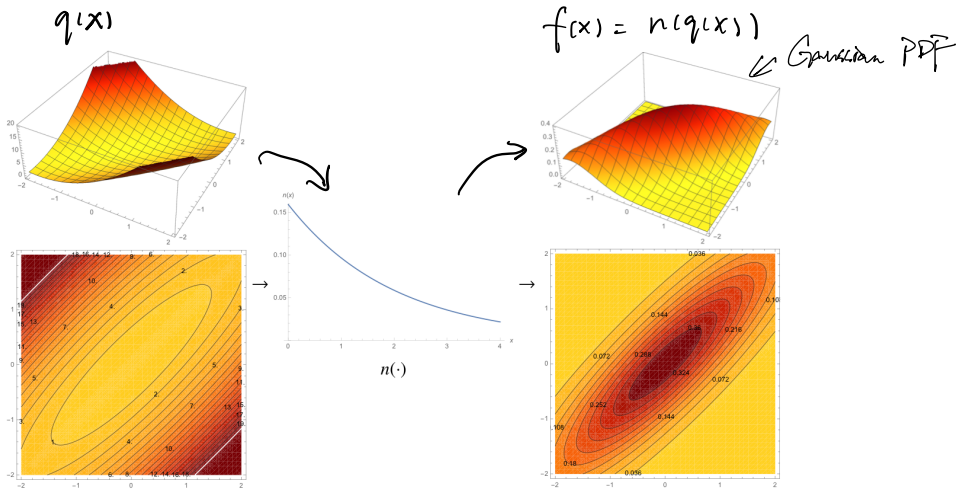
Σ^{-1} is $d \times d$ SPD precision matrix

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

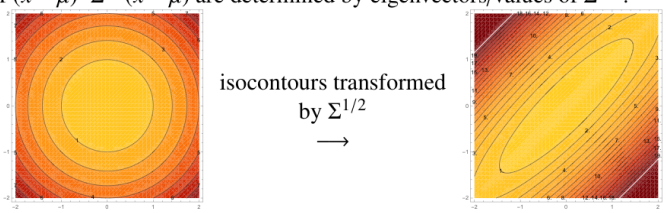
now, write $f(x) = n(q(x))$, where $q(x) = (x-\mu)^\top \Sigma^{-1} (x-\mu)$

Write $f(x) = n(q(x))$, where $q(x) = (x-\mu)^\top \Sigma^{-1} (x-\mu)$
$\uparrow$ $\uparrow$
$\mathbb{R} \to \mathbb{R}$, exponential $\mathbb{R}^d \to \mathbb{R}$, quadratic

[Now $q(x)$ is a function we understand—it's just a quadratic bowl centered at μ, the quadratic form of the precision matrix Σ^{-1}. The other function $n(\cdot)$ is a simple, monotonic, convex function, an exponential of the negation of half its argument. This mapping $n(\cdot)$ does not change the isosurfaces.]



The isocontours of $(x-\mu)^\top \Sigma^{-1}(x-\mu)$ are determined by eigenvectors/values of $\Sigma^{1/2}$.



Covariance Σ:

Let R, S be random variables—column vectors or scalars
$\mathrm{Cov}(R,S) = \mathrm{E}[(R-\mathrm{E}[R])(S-\mathrm{E}[S])^\top] = \mathrm{E}[RS^\top] - \mu_R \mu_S^\top$
$\mathrm{Var}(R) = \mathrm{Cov}(R,R)$
If R is a vector, covariance matrix for R is

$$\mathrm{Var}(R) = \begin{bmatrix} \mathrm{Var}(R_1) & \mathrm{Cov}(R_1,R_2) & \dots & \mathrm{Cov}(R_1,R_d) \\ \mathrm{Cov}(R_2,R_1) & \mathrm{Var}(R_2) & & \mathrm{Cov}(R_2,R_d) \\ \vdots & & \ddots & \vdots \\ \mathrm{Cov}(R_d,R_1) & \mathrm{Cov}(R_d,R_2) & \dots & \mathrm{Var}(R_d) \end{bmatrix} \quad \text{[symmetric; each } R_i \text{ is scalar]}$$

For a Gaussian $R \sim \mathcal{N}(\mu, \Sigma)$, one can show $\mathrm{Var}(R) = \Sigma$. [. . . as you did in Homework 2.]

Lec 9 Anisotropic Gaussians and GDA

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

$\Sigma = V \Gamma V^T$
↑ eigenvalues of Σ
↑

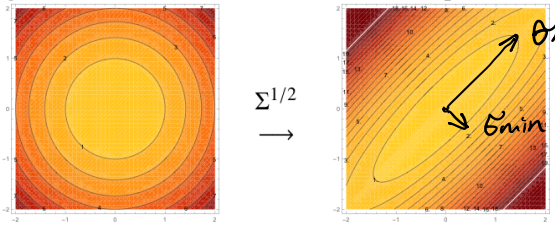
$q(x) = (x-\mu)^T \Sigma^{-1} (x-\mu)$

$\Gamma_{ii} = \sigma_i^2$

$\Sigma^{\frac{1}{2}} = V \Gamma^{\frac{1}{2}} V^T \qquad \Gamma_{ii} = \sigma_i$

$\Sigma^{1/2} = V\Gamma^{1/2}V^\top$ maps spheres to ellipsoids ($\Sigma^{1/2}$ was A in last lecture)

↑ eigenvalues of $\Sigma^{1/2}$ are Gaussian widths / ellipsoid radii / standard deviations, $\sqrt{\Gamma_{ii}} = \sigma_i$

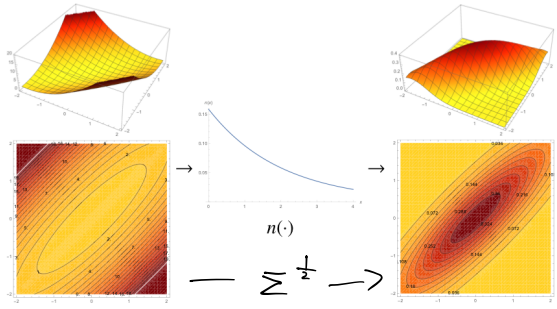


⟵ quadratic form

$q(x) = (x-\mu)^\top \Sigma^{-1} (x-\mu)$

$\Sigma^{-1} = V \Gamma^{-1} V^T$ precision matrix

recall from last lecture.



$n\left((x-\mu)^T \Sigma^{-1} (x-\mu)\right)$

4.1 Maximum Likelihood Estimation for Anisotropic Gaussians

Given training points $X_1, \ldots, X_n$ and classes $y_1, \ldots, y_n$, find best-fit Gaussians.
Let n_C = # of training pts in class C.

[Once again, we want to fit the Gaussian that maximizes the likelihood of generating the training points in a specified class. This time I won't derive the maximum-likelihood Gaussian; I'll just tell you the answer.]

For QDA:

As stated in Lec 7, know some sample pts, need to know the general Normal Distribution's Σ

$$\hat{\Sigma}_C = \frac{1}{n_C} \sum_{i:y_i=C} \underbrace{(X_i - \hat{\mu}_C)(X_i - \hat{\mu}_C)^\top}_{\text{outer product matrix, } d \times d}$$

⇐ conditional covariance for pts in class C

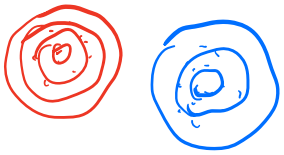
how many X_i of sample pts in class C

$\hat{\mu}_C$: sample mean for pts in class C

X_i : $\begin{bmatrix} a \\ b \\ c \\ \vdots \\ d \end{bmatrix}$ $d \times 1$ $\quad - \quad \begin{bmatrix} \hat{\mu}_1 \\ \hat{\mu}_2 \\ \vdots \\ \hat{\mu}_d \end{bmatrix}$ $d \times 1$ $\quad (X_i - \hat{\mu}_C)^T$

For LDA:

$$\hat{\Sigma} = \frac{1}{n}\sum_{C}\sum_{i:y_i=C}(X_i - \hat{\mu}_C)(X_i - \hat{\mu}_C)^\top \quad \Leftarrow \text{ pooled within-class covariance matrix}$$



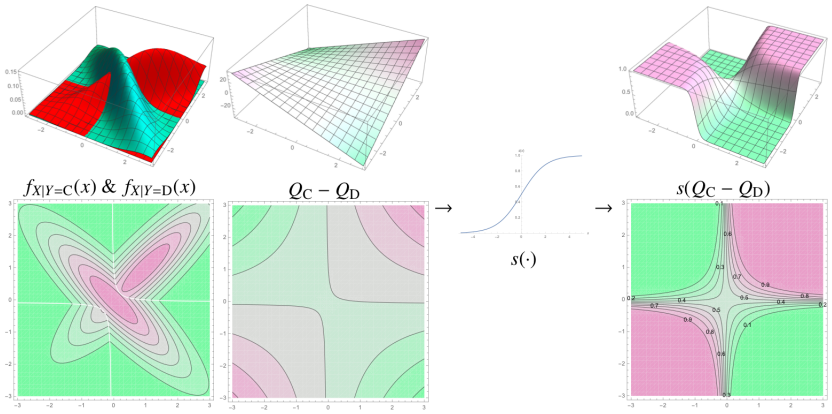
Then, choose class C that maximizes $P(Y=C|X=x)$, is the same as maximizing $QDA(x)$.

$$Q_C(x) = \ln\left((\sqrt{2\pi})^d f_{X|Y=C}(x)\,\pi_C\right)$$

$$= -\frac{1}{2}(x-\mu_C)^\top \Sigma_C^{-1}(x-\mu_C) - \frac{1}{2}\ln|\Sigma_C| + \ln\pi_C$$

$$Q_C(x) - Q_D(x) = LDA(x)$$

2 classes: Decision fn $Q_C(x) - Q_D(x)$ is quadratic, but may be indefinite.
$\Rightarrow$ Decision boundary is a quadric.
Posterior is $P(Y = C|X = x) = s(Q_C(x) - Q_D(x))$ where $s(\cdot)$ is logistic fn.



Recall Lec 7, 1d LDA

$$Q_C(x) - Q_D(x) = \underbrace{\frac{(\mu_C - \mu_D)}{\sigma^2}\cdot x}_{w\cdot x} \underbrace{- \frac{\|\mu_C\|^2 - \|\mu_D\|^2}{2\sigma^2} + \ln\pi_C - \ln\pi_D}_{+\alpha}$$

$$Q_C(x) = -\frac{1}{2}(x-\mu_C)^\top \Sigma_C^{-1}(x-\mu_C) - \frac{1}{2}\ln|\Sigma_C| + \ln\pi_C$$

$$= -\frac{1}{2}\left(x^\top\Sigma_C^{-1}x + \mu_C^\top\Sigma_C^{-1}\mu_C - x^\top\Sigma_C^{-1}\mu_C - \mu_C^\top\Sigma_C^{-1}x\right) - \frac{1}{2}\ln|\Sigma_C| + \ln\pi_C$$

$$- \left(-\frac{1}{2}\left(x^\top\Sigma_D^{-1}x + \mu_D^\top\Sigma_D^{-1}\mu_D - x^\top\Sigma_D^{-1}\mu_D - \mu_D^\top\Sigma_D^{-1}x\right) - \frac{1}{2}\ln|\Sigma_D| + \ln\pi_D\right)$$

same same

$$\frac{1}{2}\mu_C^\top\Sigma^{-1}x - \frac{1}{2}\mu_D^\top\Sigma^{-1}x + \frac{1}{2}x^\top\Sigma_C^{-1}\mu_C - \frac{1}{2}x^\top\Sigma_D^{-1}\mu_D$$

since $\mu_C^\top\Sigma^{-1}x = (\mu_C^\top\Sigma^{-1}x)^\top$
scalar $= x^\top\Sigma^{-1}\mu_C$,

Now Multi dimension LDA

$$\delta_C(x) - \delta_D(x) = \underbrace{(\mu_C - \mu_D)^T \Sigma^{-1} x}_{w^T x} - \underbrace{\frac{\mu_C^T \Sigma^{-1} \mu_C - \mu_D^T \Sigma^{-1} \mu_D}{2} + \ln \pi_C - \ln \pi_D}_{+\alpha}$$

w^T

Decision Boundary is $\delta_C(x) - \delta_D(x) = 0$, $w^T x + \alpha = 0$

Posterior is $P(Y = C \mid X = x) = s(w^T x + \alpha)$

Multi-class LDA: choose class C that maximizes the linear discriminant fn

$$\mu_C^T \Sigma^{-1} x - \frac{\mu_C^T \Sigma^{-1} \mu_C}{2} + \ln \pi_C. \qquad \text{[works for any \# of classes]}$$

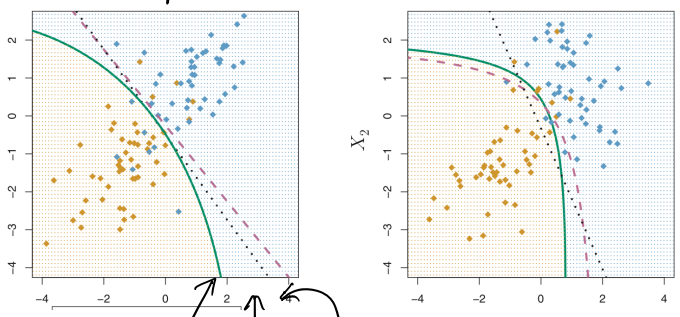
Visualize QDA & LDA.

LDA has $d+1$ parameters, $w \to d$ $\alpha \to 1$

QDA has $\frac{d(d+3)}{2} + 1$ parameters, [$\frac{d^2}{2}$ for $\ln|\Sigma_C|$, $\frac{3}{2}d$ for don't know]

LDA more likely to underfit. guess

QDA overfit. since ↓ parameters. since more parameters



QDA LDA Bayes optimal decision boundary

QDA on data doesn't find true Bayes classifier, because:

1. sample pts. 2. real-world data not perfectly Gaussian

In decorrelate, whiten data

Let X be $n \times d$ design matrix of sample pts. Each row i of X is a sample of X_i^T,

→ centering X, subtract μ^T from each row of X, $X \to \dot{X}$, μ^T is the mean of all the rows of X. Now the mean of $\dot{X}$ is zero.

$\Sigma' = \frac{1}{n} \dot{X}^T \dot{X}$ / $\hat{\Sigma}_C = \frac{1}{n_C} \dot{X}_C^T \dot{X}_C$

→ Decorrelating $\dot{X}$, apply rotation $Z = \dot{X} V$, where $\Sigma' = V \Lambda V^T$, this rotates the sample points to the eigenvector coordinate system.

Then, $\mathrm{Var}(Z) = \Lambda$, which means Z has diagonal covariance

$X_i \sim N(\mu, \Sigma)$, $Z_i \sim N(0, \Lambda)$

proof $\mathrm{Var}(Z) = \Lambda$.

Since $Z = \dot{X}V$, $\mathrm{Var}(Z) = \frac{1}{n} Z^T Z = \frac{1}{n}(\dot{X}V)^T(\dot{X}V)$

$= \frac{1}{n}(V^T \dot{X}^T \dot{X} V)$

$= \frac{1}{n} V^T V \Lambda V^T V$

$= \Lambda$

→ Sphering $\dot{X}$:

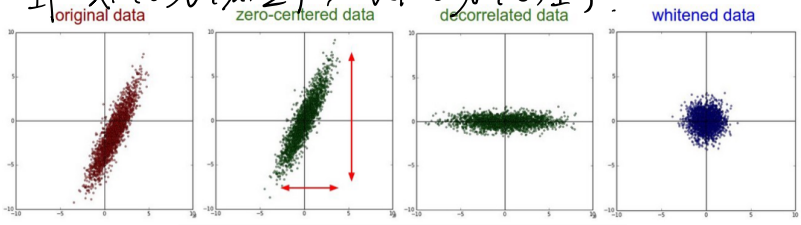
apply transform $W = \dot{X}\Sigma'^{-\frac{1}{2}}$

Recall $\Sigma^{-\frac{1}{2}}$ maps ellipsoid to sphere.

→ whitening X :

center, then sphere $X \to W$, W has covariance matrix I

If $X_i \sim \mathcal{N}(\mu, \Sigma)$, $W_i \sim \mathcal{N}(0, I)$



original.jpg, centered.jpg, decorrelated.jpg, whitened.jpg

Lec 10. Regression, Least-squares Linear and Logistic Regression

Some regression fns:

(1) linear: $h(x; w, \alpha) = w \cdot x + \alpha$

(2) polynomial [equivalent to linear regression with added polynomial features]

(3) logistic: $h(x; w, \alpha) = s(w \cdot x + \alpha)$ recall: logistic fn $s(\gamma) = \frac{1}{1+e^{-\gamma}}$

Some loss fns: let z be prediction $h(x)$; y be true label

(A) $L(z, y) = (z - y)^2$ squared error

(B) $L(z, y) = |z - y|$ absolute error

(C) $L(z, y) = -y \ln z - (1 - y) \ln(1 - z)$ logistic loss, aka cross-entropy: $y \in [0, 1]$, $z \in (0, 1)$

Some cost fns to minimize:

(a) $J(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(X_i), y_i)$ mean loss [you can leave out the "$\frac{1}{n}$"]

(b) $J(h) = \max_{i=1}^{n} L(h(X_i), y_i)$ maximum loss

(c) $J(h) = \sum_{i=1}^{n} \omega_i L(h(X_i), y_i)$ weighted sum [some points are more important than others]

(d) $J(h) =$ (a), (b), or (c) $+\lambda \|w\|^2$ ℓ_2 penalized/regularized

(e) $J(h) =$ (a), (b), or (c) $+\lambda \|w\|_{\ell_1}$ ℓ_1 penalized/regularized

Some famous regression methods:

Least-squares linear regr.:	(1) + (A) + (a)	} quadratic cost; minimize w/calculus
Weighted least-squ. linear:	(1) + (A) + (c)	
Ridge regression:	(1) + (A) + (a) + (d)	
Lasso:	(1) + (A) + (a) + (e)	quadratic program
Logistic regr.:	(3) + (C) + (a)	convex cost; minimize w/gradient descent
Least absolute deviations:	(1) + (B) + (a)	} linear program
Chebyshev criterion:	(1) + (B) + (b)	

hi Linear Regression (1) + (A) + (a)

Convention: X is $n \times d$ design matrix of sample pts
y is n-vector of scalar labels

$$\begin{bmatrix} X_{11} & X_{12} & \dots & X_{1j} & \dots & X_{1d} \\ X_{21} & X_{22} & & X_{2j} & & X_{2d} \\ \vdots & & & & & \\ X_{i1} & X_{i2} & & X_{ij} & & X_{id} \\ \vdots & & & & & \\ X_{n1} & X_{n2} & & X_{nj} & & X_{nd} \end{bmatrix} \leftarrow \text{point } X_i^\top \qquad \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$\uparrow$ feature column X_{*j} $\qquad\qquad$ $\uparrow$ y

Usually $n > d$. [But not always.]

Recall fictitious dimension trick [from Lecture 3]: rewrite $h(x) = x \cdot w + \alpha$ as

$$[x_1 \quad x_2 \quad 1] \cdot \begin{bmatrix} w_1 \\ w_2 \\ \alpha \end{bmatrix}. \qquad X : n \times (d+1)$$

Find w that minimizes $\|Xw - y\|^2$, $\|Xw-y\|^2 = RSS(w)$, which is residual sum of squares

minimize $RSS(w) = (Xw-y)^T(Xw-y)$

$= (w^TX^T - y^T)(Xw-y)$

$= w^TX^TXw - y^TXw - w^TX^Ty + y^Ty$

$= w^TX^TXw - 2y^TXw + y^Ty$

$\nabla_w RSS = \underline{2X^TXw} - 2X^Ty$

$(w+\Delta w)^TX^TX(w+\Delta w)$
$= (w^TX^TX + \Delta w^TX^TX)(w+\Delta w)$
$= \cancel{w^TX^TXw} + w^TX^TX\Delta w + \Delta w^TX^TXw + \cancel{\Delta w^TX^TX\Delta w}$
$= w^TX^TX\Delta w + (X^TXw)^T\Delta w$
$= w^TX^TX\Delta w + w^TX^TX\Delta w$
$= \underbrace{2w^TX^TX}_{k^T}\Delta w$, since $\nabla w = 0$, $\therefore 2X^TXw$

Let $2X^TXw - 2X^Ty = 0$

$X^TXw = X^Ty$

$w = (X^TX)^{-1}X^Ty$

Notice: X^TX is always PSD, but not always positive definite. If X^TX is singular, can't directly use $(X^TX)^{-1}$

X^+, which is $(X^TX)^{-1}X^T$, is pseudoinverse of X, $(d+1) \times n$, ~~$(d+1)\times n \cdot n \times (d+1)$~~, $(d+1) \times n$
which is also left inverse, since $(X^TX)^{-1}X^TX = I$

$\hat{y}_i$ is predicted values of y_i, $\hat{y}_i = w \cdot X_i$ (X_i: data point i) $\qquad \hat{y} = Xw = X(X^+y) = Hy = X(X^TX)^{-1}X^Ty$

where H is hat matrix, $H = XX^+$

Ideally, H is Identity matrix, $y = \hat{y}$, if $n > d+1$, H is singular

Linear Regression:

Pros: easy to complete, unique & stable solution

Cons: Sensitive to outliers (errors are squared), fails if X^TX is singular

(From ee16b) $\begin{bmatrix} \vec{x}[t+1] \\ \vec{x}[t+2] \\ \vdots \\ \vec{x}[t+n+1] \end{bmatrix} = [A \; B] \begin{bmatrix} \vec{x}[t] \\ \vec{x}[t+1] \\ \vdots \\ \vec{x}[t+n] \end{bmatrix}$ $\qquad \begin{bmatrix} \vec{x}^T[t] \\ \vec{x}^T[t+1] \\ \vdots \\ \vec{x}^T[t+m] \end{bmatrix} \begin{bmatrix} A^T \\ B^T \end{bmatrix} = \begin{bmatrix} \vec{x}[t+1]^T \\ \vdots \\ \vec{x}[t+n+1]^T \end{bmatrix}$)

$n \times 2$ $\quad 2 \times 2$ $\quad n \times 2$

Same: $P = (D^TD)^{-1}D^TS$

Logistic Regression (3) + (C) + (a)

$$s(w \cdot x + \alpha) = \frac{1}{1 + e^{-(w \cdot x + \alpha)}}$$

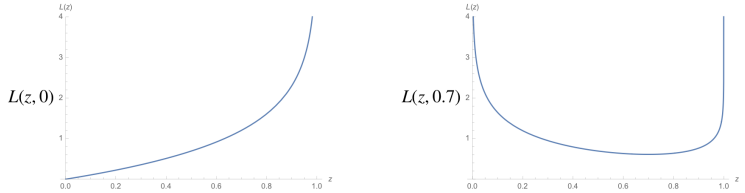
With X and w including the fictitious dimension; α is w's last component ...

Find w that minimizes

$$J = \sum_{i=1}^{n} L(s(X_i \cdot w), y_i) = -\sum_{i=1}^{n} \Big(y_i \ln s(X_i \cdot w) + (1 - y_i) \ln (1 - s(X_i \cdot w)) \Big).$$

Since we're using the loss func (C), which is $L(z, y) = -y \ln z - (1-y) \ln (1-z)$

logistic loss, a.k.a cross entropy

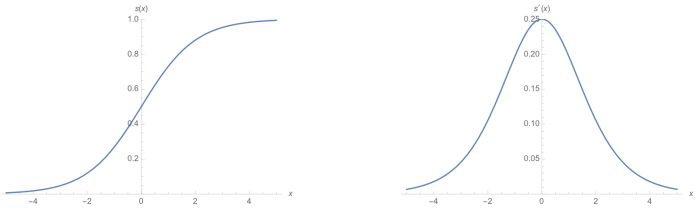


logloss0.pdf, loglosspt7.pdf [Plots of the loss $L(z, y)$ for $y = 0$ (left) and $y = 0.7$ (right). As you might guess, the left function is minimized at $z = 0$, and the right function is minimized at $z = 0.7$. These loss functions are always convex.]

$J(w)$ is convex, so can solve by gradient descent.

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

$$s'(\gamma) = \frac{e^{-\gamma}}{(1 + e^{-\gamma})^2} = s(\gamma)(1 - s(\gamma))$$



logistic.pdf, dlogistic.pdf [Plots of $s(\gamma)$ (left) and $s'(\gamma)$ (right).]

Let $s_i = s(X_i \cdot w) = \frac{1}{1 + e^{-X_i \cdot w}}$

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

$$= \nabla_w \Big(-\sum_{i=1}^{n} \big(y_i \ln s_i + (1 - y_i) \ln (1 - s_i) \big) \Big)$$

$$= -\sum y_i \frac{1}{s_i} \nabla_w s_i - \frac{1 - y_i}{1 - s_i} \nabla_w s_i$$

$$= -\sum \Big(\frac{y_i}{s_i} - \frac{1 - y_i}{1 - s_i} \Big) \nabla_w s_i \quad \Rightarrow \quad s'(\gamma) = s(\gamma)(1 - s(\gamma)), \ \gamma = X_i \cdot w$$

$$= -\sum \Big(\frac{y_i}{s_i} - \frac{1 - y_i}{1 - s_i} \Big) s_i (1 - s_i) X_i \quad \nabla_w s(\gamma) = s'(\gamma) \cdot \gamma' = s(\gamma)(1 - s(\gamma)) \cdot X_i$$

$$= -\sum \big(y_i (1 - s_i) - (1 - y_i) s_i \big) X_i$$

$$= -\sum (y_i - y_i s_i - s_i + y_i s_i) X_i$$

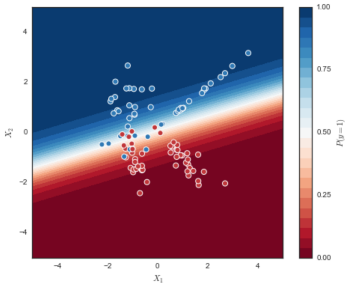
$$= \boxed{-\sum (y_i - s_i) X_i}$$

$$= \boxed{-X^T (y - s(Xw))} \quad \text{where} \quad s(Xw) = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix}$$

$$\begin{bmatrix} — & X_1^T & — \\ & \vdots & \\ — & X_n^T & — \end{bmatrix} \begin{bmatrix} y_1 - s_1 \\ \vdots \\ y_n - s_n \end{bmatrix}$$

- gradient descent : $w \leftarrow w + \varepsilon X^T(y - s(Xw))$; $w - (\varepsilon \cdot -X^T(y - s(Xw)))$
  - learning rate
  - (Gradient Descent, Note 3)
- Stochastic gradient descent : $w \leftarrow w + \varepsilon (y_i - s(x_i \cdot w)) x_i$; $w - (\varepsilon \cdot -(y_i - s_i) x_i)$
  - (no Σ, because it's a single point ∇.)
  - ↓ works best if shuffling points

Logistic Regression always separates linearly separable pts.



problogistic.png, by "mwascom" of Stack Overflow
http://stackoverflow.com/questions/28256058/plotting-decision-boundary-of-logistic-regression
[An example of logistic regression.]

Lec 11 More Regression: Newton's Method, ROC Curves

h1 Least-Squares polynomial regression

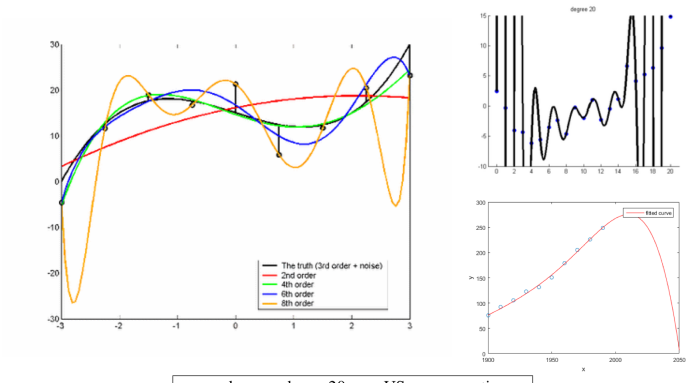
Replace each x_i with feature vector $\phi(x_i)$ with all terms of degree $0, \ldots, p$

eg. degree $0, \ldots, 2$.

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

Log. reg. + quadratic features = same form of posteriors as QDA.

Very easy to overfit!



overunder.png, degree20.png, UScensusquartic.png

h2 weighted least-square regression (1) + (A) + (C)

$\Omega = \begin{bmatrix} w_1 & & 0 \\ & w_2 & \\ 0 & & \ddots \\ & & & w_n \end{bmatrix}$

Remember in Hw2: $\langle A, B \rangle = \operatorname{trace}(A^\top B) = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ij}$.

$\langle A \cdot x, y \rangle = A(yx^\top)$

$= x^\top A y$

$\sum_{i=1}^{n} w_i (x_i \cdot w - y_i)^2 = \langle w, k_i \cdot w - y_i \rangle^2$

$= \langle w, (x_i \cdot w - y_i)^\top (x_i w - y_i) \rangle$

Assign each sample pt a weight ω_i; collect ω_i's in $n \times n$ diagonal matrix Ω.

Greater $\omega_i \rightarrow$ work harder to minimize $(\hat{y}_i - y_i)^2$ recall: $\hat{y} = Xw$ [$\hat{y}_i$ is predicted label for X_i]

Find w that minimizes $(Xw - y)^\top \Omega (Xw - y)$ $= \sum_{i=1}^{n} \omega_i (X_i \cdot w - y_i)^2$

$\|Xw - y\|^2 = (Xw - y)^\top (Xw - y)$

[As with ordinary least-squares regression, we find the minimum by setting the gradient to zero, which leads us to the normal equations.]

Solve for w in normal equations: $X^\top \Omega X w = X^\top \Omega y$

$X_i w - y_i$

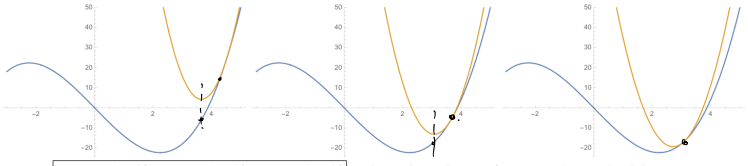
$X w - y$

$\begin{bmatrix} X_1 w - y_1 \\ X_2 w - y_2 \\ \vdots \\ X_n w - y_n \end{bmatrix}$

h3 Newton's Method

Iterative optimization method for smooth fn $J(w)$.
Often much faster than gradient descent. [We'll use Newton's method for logistic regression.]

Idea: You're at point v. Approximate $J(w)$ near v by quadratic fn.
Jump to its unique critical pt. Repeat until bored.



newton1.pdf, newton2.pdf, newton3.pdf [Three iterations of Newton's method in one-dimensional space. We seek the minimum of the blue curve, J. Each brown curve is a local quadratic approximation to J. Each iteration, we jump to the bottom of the brown parabola.]

Appendix:

Taylor Series:

$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots$

$\nabla J(w) = \nabla J(v) + \frac{\nabla^2 J(v)}{1!}(w - v)$

How to approximate $J(w)$ near v by quadratic fn?

Taylor series about v:

$$\nabla J(w) = \nabla J(v) + (\nabla^2 J(v))(w - v) + O(\|w - v\|^2)$$

approximated $J(w)$ (brown parabola)

where $\nabla^2 J(v)$ is the Hessian matrix of J at v.

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a function taking as input a vector $\mathbf{x} \in \mathbb{R}^n$ and outputting a scalar $f(\mathbf{x}) \in \mathbb{R}$. If all second-order partial derivatives of f exist, then the Hessian matrix $\mathbf{H}$ of f is a square $n \times n$ matrix, usually defined and arranged as

$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

That is, the entry of the ith row and the jth column is

$$(\mathbf{H}_f)_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

Jump to the bottom of brown parabola ← Find critical pt w by setting $\nabla J(w) = 0$:

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

Newton's Method: (New type of Gradient Descent)

pick starting point w, repeat until convergence:

$w = w + e$, where e is

$(\nabla^2 J(w)) e = -\nabla J(w)$

Warning: Doesn't know difference between minima, maxima, saddle pts.
Starting pt must be "close enough" to desired critical pt.

[If the objective function J is actually quadratic, Newton's method needs only one step to find the exact solution. The closer J is to quadratic, the faster Newton's method tends to converge.]

[Newton's method is superior to blind gradient descent for some optimization problems for several reasons. First, it tries to find the right step length to reach the minimum, rather than just walking an arbitrary distance downhill. Second, rather than follow the direction of steepest descent, it tries to choose a better descent direction.]

[Nevertheless, it has some major disadvantages. The biggest one is that computing the Hessian can be quite expensive, and it has to be recomputed every iteration. It can work well for low-dimensional weight spaces, but you would never use it for a neural network, because there are too many weights. Newton's method also doesn't work for most nonsmooth functions. It particularly fails for the perceptron risk function, whose Hessian is zero, except where the Hessian is not even defined.]

h4 Logistic Regression Using Newton's Method

Since $w = w + e$, where $(\nabla^2 J(w))\, e = -\nabla J(w)$
need to solve $\nabla_w J$ and $\nabla_w^2 J(w)$

Recall: $s'(\gamma) = s(\gamma)(1 - s(\gamma))$, $\quad s_i = s(X_i \cdot w)$, $\quad s = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix}$,

$$\nabla_w J = -\sum_{i=1}^{n} (y_i - s_i) X_i = -X^\top (y - s)$$

[Now let's derive the Hessian too, so we can use Newton's method.]

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

Ω is (+ve) definite $\forall w \;\Rightarrow\; X^\top \Omega X$ is (+ve) semidefinite $\forall w \;\Rightarrow\; J(w)$ is convex.
[The logistic regression cost function is convex, so Newton's method finds a globally optimal point if it converges at all.]

Newton's method:

$w \leftarrow 0$
repeat until convergence
$e \leftarrow$ solution to normal equations $(X^\top \Omega X)\, e = X^\top (y - s)$ Recall: Ω, s are fns of w
$w \leftarrow w + e$

[Notice that this looks a lot like weighted least squares, but the weight matrix Ω and the right-hand-side vector $y - s$ change every iteration. So we call it . . .]
An example of iteratively reweighted least squares.

[Misclassified points far from the decision boundary have the most influence on the step e, and correctly classified points far from the decision boundary have the least (because $y_i - s_i$ is small for such a point). Points near the decision boundary have medium influence. But if there are no misclassified points far from the decision boundary, then points near the decision boundary have most of the influence.]

[Here's one more idea for speeding up logistic regression.] Idea: If n very large, save time by using a random subsample of the pts per iteration. Increase sample size as you go.

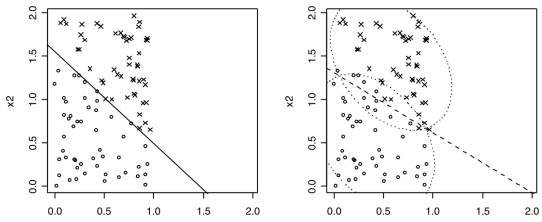
[The principle is that the first iteration isn't going to take you all the way to the optimal point, so why waste time looking at *all* the sample points? Whereas the last iteration should be the most accurate one.]

LDA vs. Logistic Regression

LDA pros: 1. for well separated class, stable. Logistic regression unstable
2. > 2 classes easy to classify. Logistic regression needs modifying
3. LDA slightly more accurate when classes nearly normal, and if n small.

Logistic Regression pros: 1. More emphasis on decision boundary, always separates linearly separable pts.
2. – More robust on some non-Gaussian distributions (e.g., dists. w/large skew)
3. – Naturally fits labels between 0 and 1 [usually probabilities]

[Correctly classified points far from the decision boundary have a small effect on logistic regression—albeit a bigger effect than they have on SVMs—whereas misclassified points far from the decision boundary have the biggest effect. By contrast, LDA gives all the sample points equal weight when fitting Gaussians to them. Weighting points according to how badly they're misclassified is good for reducing training error, but it can also be bad if you want stability or insensitivity to bad data.]



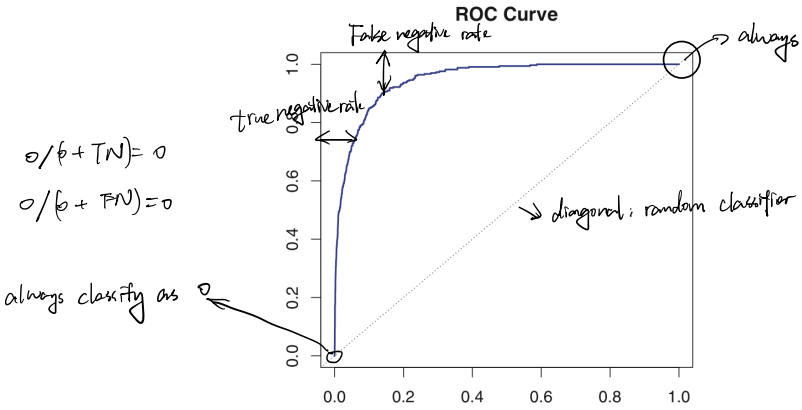
logregvsLDAuni.pdf [Logistic regression vs. LDA for a linearly separable data set with a very narrow margin. Logistic regression (center) always succeeds in separating linearly separable classes, because the cost function approaches zero for a correct linear separator. In this example, LDA (right) misclassifies some of the training points.]

h6 ROC curve (Receiver Operating characteristics)

x-axis : false positive rate = FP / (FP + TN) aka. specificity
(+/- , -/-)

y-axis : true positive rate = TP / (TP + FN) aka sensitivity
(+/+ , -/+)

Generate the curve by trying every probability threshold.



always classify as 1
FP/(FP+TN) = FP/FP = N/N = 1
TP/(TP+FN) = TP/TP = P/P = 1

0/(0+TN) = 0
0/(0+FN) = 0

Area Under Curve (AUC), if AUC = 1, good classifier, AUC = 1/2, random classifier

Why?

Since if we have a threshold that achieves TPR = 1, then "adding" threshold will remain TPR = 1, since all p will be classified as p, p/p = 1.

↓ is because FP should always be 0, if it > 0, it never decreases. Because the threshold will "increase"

Lec 12 Statistical Justifications for Regression & Bias-Variance Decomposition

Typical model of reality:
- sample points come from unknown prob. distribution: $X_i \sim D$.
- y-values are sum of unknown, non-random fn + random noise:

$$\forall X_i, \quad y_i = g(X_i) + \epsilon_i, \qquad \epsilon_i \sim D', \quad D' \text{ has mean zero.}$$

Goal of regression: find h that estimates g.
Ideal approach: choose $h(x) = \mathrm{E}_Y[Y|X = x] = g(x) + \mathrm{E}[\epsilon] = g(x)$. ($\mathrm{E}[\epsilon] = 0$)

Statistical Justifications for Regression:

h1 Least-Square Cost Function from Maximum Likelihood

Suppose $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$; then $y_i \sim \mathcal{N}(g(X_i), \sigma^2)$.
Recall that log of normal PDF is (remember $f(y_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i-\mu_i)^2}{2\sigma^2}}$)

$$\ln f(y_i) = -\frac{(y_i - \mu_i)^2}{2\sigma^2} - \text{constant} \qquad \Leftarrow \mu_i = g(X_i)$$

& log likelihood is (negative sign, so minimize)

$$\ell(g; X, y) = \ln(f(y_1) f(y_2) \cdots f(y_n)) = \ln f(y_1) + \ldots + \ln f(y_n) = \boxed{-\frac{1}{2\sigma^2}} \sum (y_i - g(X_i))^2 - \text{constant}.$$

Takeaway: Max likelihood on "parameter" $g \Rightarrow$ choose a g that minimizes $\sum(y_i - g(X_i))^2$. $\Rightarrow$ loss func (A)

h2 Empirical Risk (A type of Cost Function)

The risk for hypothesis h is:
the expected loss $R(h)$, which $R(h) = E[h]$ (recall Lec 6/7)
over all (X, Y) in some joint distribution

But now, we don't know X's distribution D. How can we minimize risk?
We use Empirical Distribution!

Empirical Distribution:
The discrete uniform distribution over the sample points.

Empirical Risk: ↑
Expected loss under empirical distribution

$$\hat{R}(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(X_i), y_i) \Longrightarrow \text{Cost function (A) mean loss}$$

Diff between risk and empirical risk: choose n samples instead of knowing the whole distribution.

also, $L(h(X_i), y_i)$ can be $\rightarrow (h(X_i) - y_i)^2$

h3 Logistic Loss from Maximum Likelihood (A type of Cost Function)

Also, we don't know the distribution of X, what cost function for Logistic Regression?
Now, given the actual probability x_i is in class C is y_i,
our predicted probability of it is $h(X_i)$
Imagine β duplicate copies of x_i, with $y_i\beta$ in class C and $(1-y_i)\beta$ not in class C

$$\text{Likelihood is } \mathcal{L}(h; X, y) = \prod_{i=1}^{n} h(X_i)^{\overbrace{y_i\beta}^{\text{fixed}}} (1 - h(X_i))^{\overbrace{(1-y_i)\beta}^{\text{fixed}}}$$

$\binom{\beta}{y_i\beta}$ fixed, so can be discarded. $h(X_i)$ not fixed.

e.g. $y_i = \frac{2}{3}$, $\beta = 6$: 6 in C, 3 not in C, $h(X_i) \Rightarrow C = 50\%$.

$\mathcal{L}(h; X, y) = 0.5^6 \cdot 0.5^3$

recall Binomial distribution, given $P[X=C] = P$, $P[X \neq C] = (1-P)$, there're 6 C, 3 not C, probability of that happens? $\binom{9}{3} P^6 (1-P)^3$

with all X_i, $\prod_{i=1}^{n} p^k(1-p)^{n-k}$

Log likelihood is $\ell(h) = \ln \mathcal{L}(h)$

$$= \beta \sum_i \Big(y_i \ln h(X_i) + (1 - y_i) \ln(1 - h(X_i)) \Big) \Longrightarrow \text{Loss func (C)}$$

$$= \underbrace{(-\beta)}_{\text{negative}} \sum \text{logistic loss fn } L(h(X_i), y_i).$$

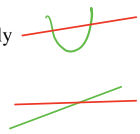
Takeaway: Max likelihood ⇒ minimize ∑ logistic losses.

4 The Bias – Variance Decomposition

There are 2 sources of error in a hypothesis h:

Not accurate → bias: error due to inability of hypothesis h to fit g perfectly
e.g., fitting quadratic g with a linear h

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



→ Notice Not normal distribution Like h1 Least square.

Model: $X_i \sim D$, $\epsilon_i \sim D'$, $y_i = g(X_i) + \epsilon_i$ [remember that D' has mean zero]
fit hypothesis h to X, y

Now h is a random variable: i.e., its weights are random

Consider arbitrary pt $z \in \mathbb{R}^d$ (not necessarily a sample pt!) & $\gamma = g(z) + \epsilon$, $\epsilon \sim D'$

Note: $E[\gamma] = g(z)$; $\mathrm{Var}(\gamma) = \mathrm{Var}(\epsilon)$ [the mean comes from g, and the variance comes from ϵ]

Risk fn when loss = squared error:

Risk function when loss is squared error:

$R(h) = E[L(h(z), \gamma)]$, it means taking expectation over possible training set X, y & values of γ.

$$= E[(h(z) - \gamma)^2]$$

independent to each other, why? don't know

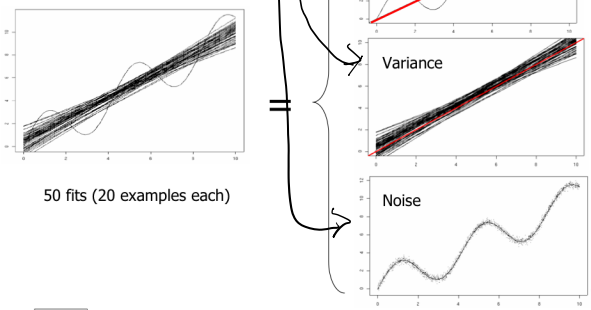
$$= E[h(z)^2] + E[\gamma^2] - 2E[\gamma h(z)]$$

$$= \mathrm{Var}(h(z)) + E[h(z)]^2 + \mathrm{Var}(\gamma) + E[\gamma]^2 - 2E[\gamma]E[h(z)]$$

$$= (E[h(z)] - E[\gamma])^2 + \mathrm{Var}(h(z)) + \mathrm{Var}(\epsilon)$$

$$= (E[h(z)] - g(z))^2 + \mathrm{Var}(h(z)) + \mathrm{Var}(\epsilon)$$

bias² of method | variance of method | irreducible error



50 fits (20 examples each)

← black lines are our predicted h(z), not always the same. 50 different lines

bvn.pdf [In this example, we're trying to fit a sine wave with lines, which obviously aren't going to be accurate. At left, we have generated 50 different hypotheses (lines). Each line was generated from 20 random training points by least-squares linear regression. At upper right, the red line is the *expected hypothesis*—an average over infinitely many hypotheses. The black curve illustrates test points on the true function g. We see that most test points have a large bias (difference between the black and red curves), because lines don't fit sine waves well. However, some of the test points happen to have a small bias—where the sine wave crosses the red line. At center right, the variance is the expected squared difference between a random black line and the red line (at a test point z). At lower right, the irreducible error is the expected squared difference between a random test point and the sine wave.]

- √ – Underfitting = too much bias
- √ – Most overfitting caused by too much variance → 
- √ – Training error reflects bias but not much variance; test error reflects both
  [which is why low training error can fool you when you've overfitted]
- √ – For many distributions, variance → 0 as $n \to \infty$ ←
  - If h can fit g exactly, for many distributions bias → 0 as $n \to \infty$
  - If h cannot fit g well, bias is large at "most" points
- √ – Adding a good feature reduces bias; adding a bad feature rarely increases it
- √ – Adding a feature usually increases variance [don't add a feature unless it reduces bias more]
- √ – Can't reduce irreducible error [hence its name]
  - Noise in test set affects only Var(ϵ);
    noise in training set affects only bias & Var(h)
  - We can't precisely measure bias or variance of real-world data
    [because we cannot know g exactly and our noise model might be wrong]
  - But we can test learning algs by choosing g & making synthetic data

Variance is high when $x^2, x^3, x^4, \ldots$
$\mathrm{Var}[h(z)])$ $\mathrm{Var}(\theta x^2), \mathrm{Var}(\theta x^3), \ldots$ ↑↑

example of Bias-Variance decomposition : Least-Square Linear Reg.

Model: $g(z) = v^\top z$ (ground truth is linear)
[So we could fit g perfectly with a linear h if not for the noise in the training set.]
Let e be noise n-vector, $e_i \sim \mathcal{N}(0, \sigma^2)$
Training labels: $y = Xv + e$
[X & y are the inputs to linear regression. We don't know v or e.]

Lin. reg. computes weights

$$w = X^+ y = X^+(Xv + e) = v + \underbrace{X^+ e}_{\text{noise in weights}}$$ [We want $w = v$, but the noise in y becomes noise in w.]

Bias :

BIAS is $\mathrm{E}[h(z)] - g(z) = \mathrm{E}[w^\top z] - v^\top z = z^\top \mathrm{E}[w - v] = z^\top \mathrm{E}[X^+ e] = z^\top \mathrm{E}[X^+]\mathrm{E}[e] = 0$

Variance :

VARIANCE is $\mathrm{Var}(h(z)) = \mathrm{Var}(w^\top z) = \mathrm{Var}(v^\top z + (X^+ e)^\top z) = \mathrm{Var}(z^\top X^+ e)$

Takeaways: Bias can be zero when hypothesis function can fit the real one!
[This is a nice property of the squared error loss function.]
Variance portion of RSS (overfitting) decreases as $1/n$ (sample points),
increases as d (features)
or $O(d^p)$ if you use degree-p polynomials.

review note 12 for more details !!

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

h1

(1) + (A) + ℓ_2 penalized mean loss (d). Linear $w \cdot x + \alpha$ + Squared error + l_2 regulation cost func

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

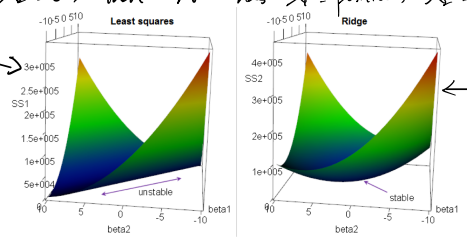
where w' is w with component α replaced by 0.
X has fictitious dimension but we DON'T penalize α.

Adds a regularization term, aka a penalty term, for shrinkage: to encourage small $\|w'\|$. Why?

① – Guarantees positive definite normal eq'ns; always unique solution.
[Standard least-squares linear regression yields singular normal equations when the sample points lie on a common hyperplane in feature space.] E.g., when $d > n$.

Image there's a matrix M that $\|Mw\|^2 = \|Xw - y\|^2$, if M is PSD matrix,
λ_1 = postive, λ_2 = "0", then $M^{-\frac{1}{2}}$ has λ_1 = positive, λ_2 = "∞", then $\|Xw - y\|^2$ is the left graph.

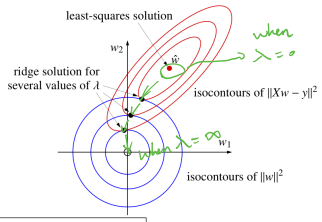
$\|Xw - y\|^2$



$\|Xw - y\|^2 + \lambda\|w'\|^2$

ridgequad.png [The cost function $J(w)$ with and without regularization.]

– Reduces overfitting by reducing variance. Why?
Imagine: $500x_1 - 500x_2$ is best fit for well-separated points with $y_i \in [0, 1]$.
Small change in x $\Rightarrow$ big change in y!



ridgeterms2.pdf (redrawing of ISL, Figure 6.7) [In this plot of weight space, for simplicity, we're not using a bias term α (we set it to zero). $\hat{w}$ is the least-squares solution. The red ellipses are isocontours of $\|Xw - y\|^2$. The blue circles are isocontours of $\|w\|^2$, centered at the origin. The ridge regression solution lies where a red isocontour just touches a blue isocontour tangentially. As λ increases, the solution will occur at a more outer red isocontour and a more inner blue isocontour. This helps to reduce overfitting.]

Recall for Linear Regression : $w = (X^T X)^{-1} X^T y$, for Ridge Regression, it becomes:

Setting $\nabla J = 0$ gives normal eq'ns

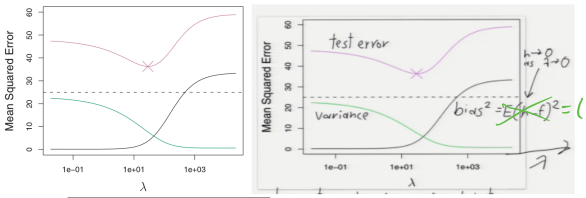
$$(X^\top X + \lambda I')\, w = X^\top y$$

where I' is identity matrix w/bottom right set to zero. [Don't penalize the bias term α.]
[Don't worry; $X^\top X + \lambda I'$ is always positive definite for $\lambda > 0$, assuming X ends with a column of 1's.]

Algorithm: Solve for w. Return $h(z) = w^\top z$. hypothesis what is $h(z)$?? guess $z = x_i$

Increasing λ $\Rightarrow$ more regularization; smaller $\|w'\|$

~~Recall [from the previous lecture] our data model $y = Xw + e$, where e is noise.~~
~~Variance of ridge regr. is $\mathrm{Var}(c^\top (X^\top X + \lambda I')^{-1} X^\top e)$.~~
As $\lambda \to \infty$, variance $\to 0$, but bias increases.



ridgebiasvar.pdf (ISL, Figure 6.5) [Plot of bias^2 & variance as λ increases.]

[So, as usual for the bias-variance trade-off, the test error as a function of λ is a U-shaped curve. We find the bottom by validation.]

λ is a hyperparameter; tune by (cross-)validation.

Ideally, features should be "normalized" to have same variance.
Alternative: use asymmetric penalty by replacing I' w/other diagonal matrix. [For example, if you use polynomial features, you could use different penalties for monomials of different degrees.]

h2

Bayesian Justification for Ridge Reg.

$$f(y_i|\mathbf{x}_i, \mathbf{w}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2/(2\sigma^2)}$$

Assign a prior probability on w': $w' \sim \mathcal{N}(0, \varsigma^2)$. Apply MLE to maximize the posterior prob.

w' has PDF $f(w') \propto e^{-\|w'\|/2\varsigma^2}$

$L(w)$ likelihood function $= \prod_{i=1}^{n} f(y_i|x_i, w)$

$= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-(y_i - w\cdot x_i)^2/2\sigma^2}$

Log.

$$\begin{aligned}
\text{Bayes' Theorem: posterior } f(w|X, y) &= \frac{f(y|X, w) \times \text{ prior } f(w')}{f(y|X)} = \frac{\mathcal{L}(w)\, f(w')}{f(y|X)} \\
\text{Maximize log posterior} &= \ln \mathcal{L}(w) + \ln f(w') - \text{const} \\
&= -\text{const}\, \|Xw - y\|^2 - \text{const}\, \|w'\|^2 - \text{const} \\
&\Rightarrow \text{Minimize } \|Xw - y\|^2 + \lambda \|w'\|^2
\end{aligned}$$

joint $e^{-(\|w'\|^2/2\varsigma^2)}$ not $e^{-\|w\|/2\varsigma^2}$

[We are treating w and y as random variables, but X as a fixed constant—it's not random.]
This method (using likelihood, but maximizing posterior) is called maximum *a posteriori* (MAP).

h3

FEATURE SUBSET SELECTION

[Some of you may have noticed as early as Homework 1 that you can sometimes get better performance on a spam classifier simply by dropping some useless features.]

All features increase variance, but not all features reduce bias.

Idea: Identify poorly predictive features, ignore them (weight zero).
Less overfitting, smaller test errors.
2nd motivation: Inference. Simpler models convey interpretable wisdom.

Useful in all classification & regression methods.

Sometimes it's hard: Different features can partly encode same information.
Combinatorially hard to choose best feature subset.

Alg: Best subset selection. Try all $2^d - 1$ nonempty subsets of features. [Train one classifier per subset.]
Choose best classifier by (cross-)validation. Slow.

[Obviously, best subset selection isn't feasible if we have a lot of features. But it gives us an "ideal" algorithm to compare practical algorithms with. If d is large, there is no algorithm that's guaranteed to find the best subset and that runs in acceptable time. But heuristics often work well.]

Heuristic 1: Forward stepwise selection.
Start with null model (0 features); repeatedly add best feature until validation errors start increasing (due to overfitting) instead of decreasing. At each outer iteration, inner loop tries every feature & chooses the best by validation. Requires training $O(d^2)$ models instead of $O(2^d)$.
Not perfect: e.g., won't find the best 2-feature model if neither of those features yields the best 1-feature model. [That's why it's a heuristic.]

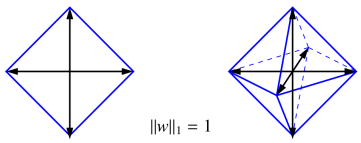
Heuristic 2: Backward stepwise selection.
Start with all d features; repeatedly remove feature whose removal gives best reduction in validation error.
Also trains $O(d^2)$ models.

[Forward stepwise is a better choice when you suspect only a few features will be good predictors; e.g., spam. Backward stepwise is better when most features are important. If you're lucky, you'll stop early.]

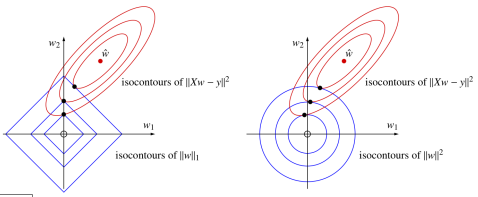
h4 LASSO (least Absolute shrinkage and selection operator) pro: often naturally set some features to 0.
(1) + (A) + L1 penalized mean loss (e)

Find w that minimizes $\|Xw - y\|^2 + \lambda \|w'\|_1$ where $\|w'\|_1 = \sum_{i=1}^{d} |w_i|$ (Don't penalize α.)

The isosurfaces of $\|w'\|_1$ are cross-polytopes.
The unit cross-polytope is the convex hull of all the positive & negative unit coordinate vectors.

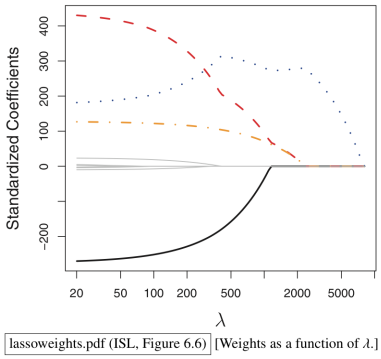


[Draw this figure by hand crosspolys.png]



lassoridge2.pdf [Isocontours of the terms of the objective function for the Lasso appear at left. Compare with the ridge regression isocontours at right.]

[The red ellipses are the isocontours of $\|Xw - y\|^2$, and the least-squares solution lies at their center. The isocontours of $\|w'\|_1$ are diamonds centered at the origin (blue). The solution lies where a red isocontour just touches a blue diamond. What's interesting here is that in this example, the red isocontour touches just the tip of the diamond. So the weight w_1 gets set to zero. That's what we want to happen to weights that don't have enough influence. This doesn't always happen—for instance, the red isosurface could touch a side of the diamond instead of a tip of the diamond.]



lassoweights.pdf (ISL, Figure 6.6) [Weights as a function of λ.]

[This shows the weights for a typical linear regression problem with about 10 variables. You can see that as lambda increases, more and more of the weights become zero. Only four of the weights are really useful for prediction; they're in color. Statisticians used to choose λ by looking at a chart like this and trying to eyeball a spot where there aren't too many predictors and the weights aren't changing too fast. But nowadays they prefer validation.]

Sometimes sets some weights to zero, especially for large λ.
Algs: subgradient descent, least-angle regression (LARS), forward stagewise