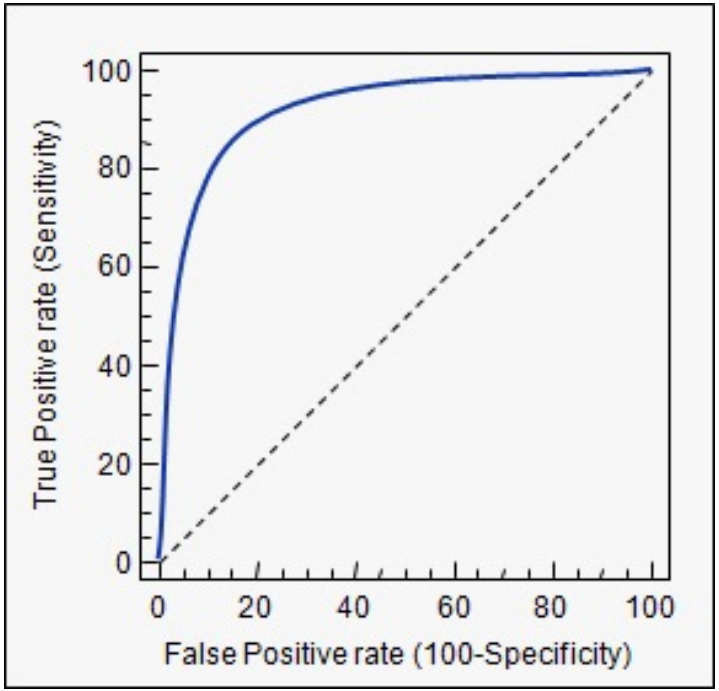
Evaluation, Feature Selection, Support Vector Machines

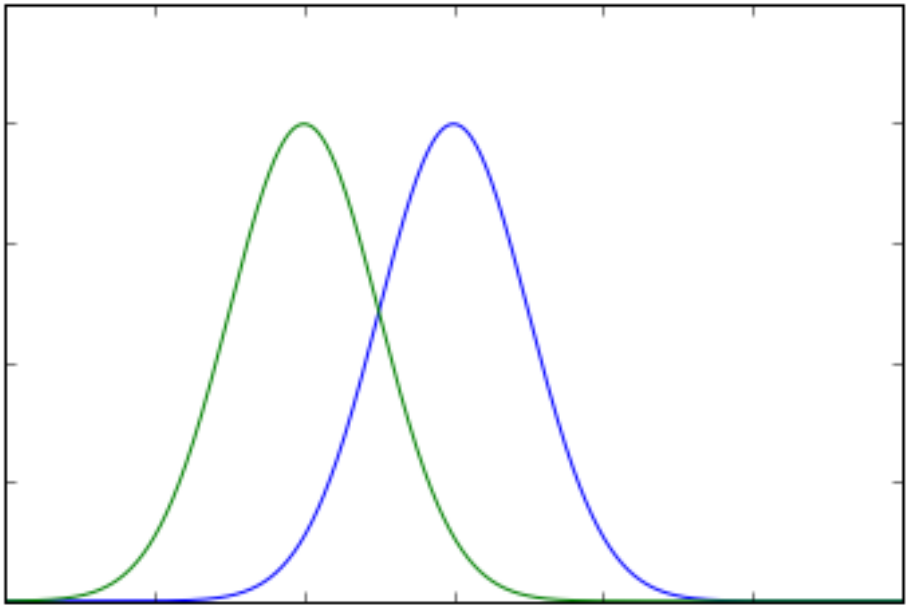
David Quigley
CSCI 5622
2021 Fall

Course Logistics

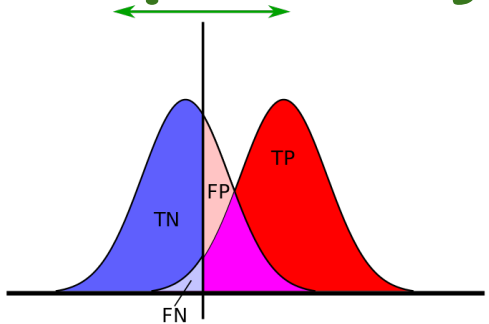
- Problem Set 2: 10/7 @ 11:59PM
- Project Milestone 2 Class Feedback Process on Delay
  - Harder to do Asynchronously
  - Everyone deserves engagement and participation!

Classification Errors – ROC Curve (receiver operating characteristic)

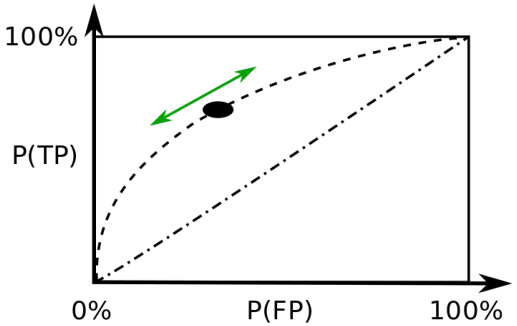




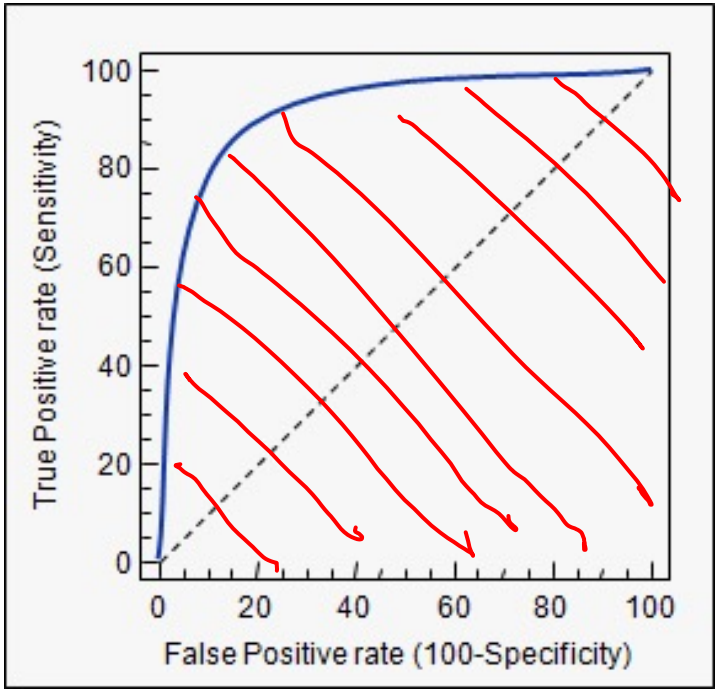
Adapting True Pos rate vs. False Pos rate (sensitivity vs. specificity)

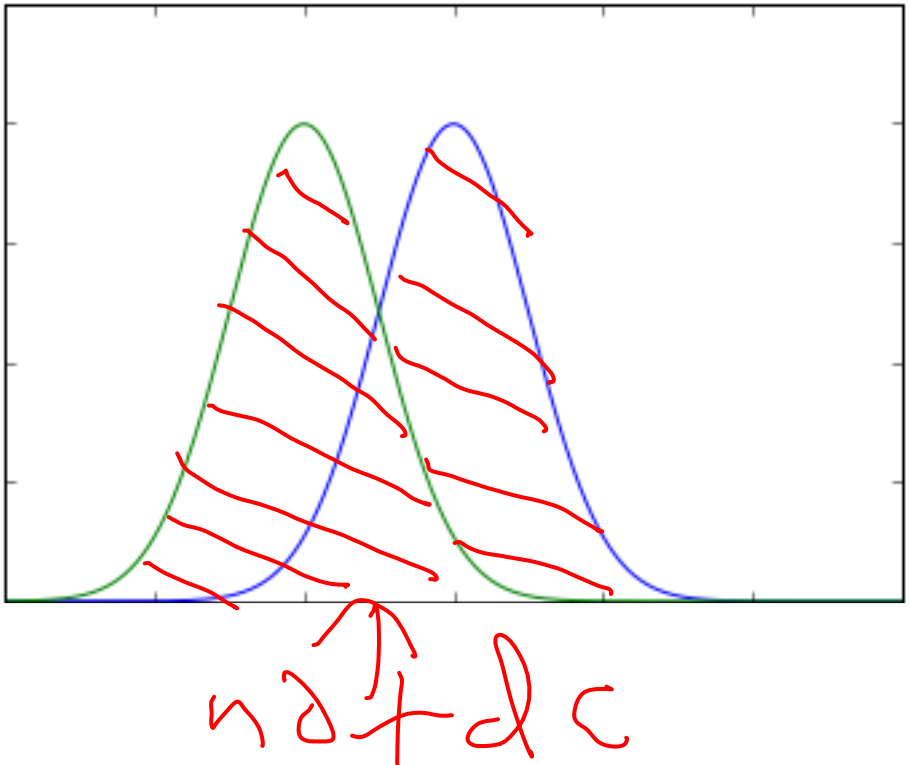


TP	FP
FN	TN

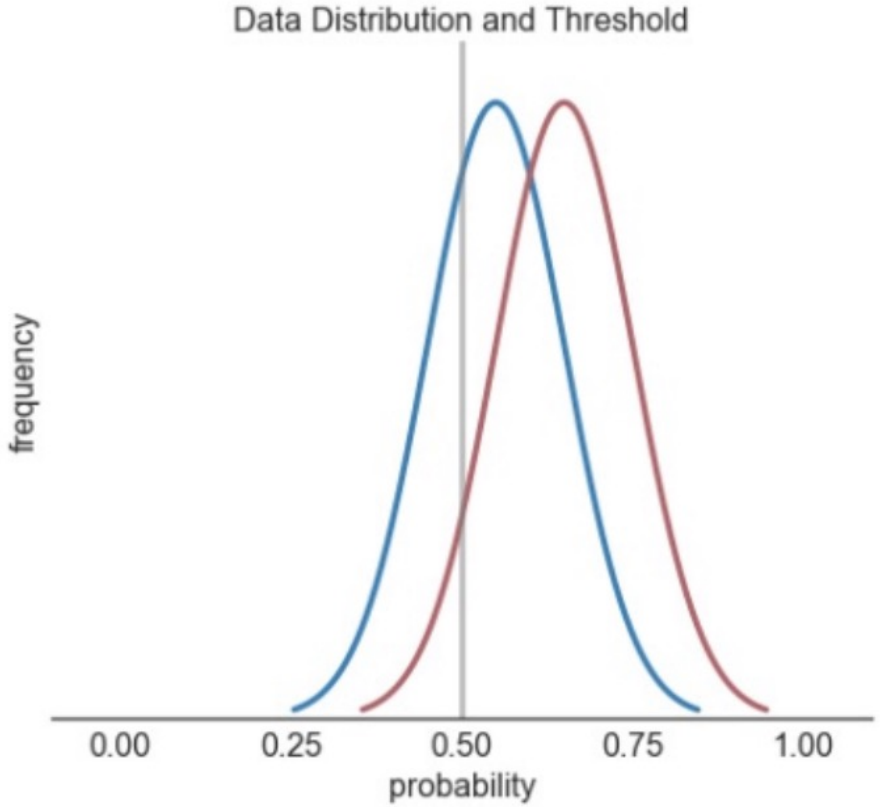


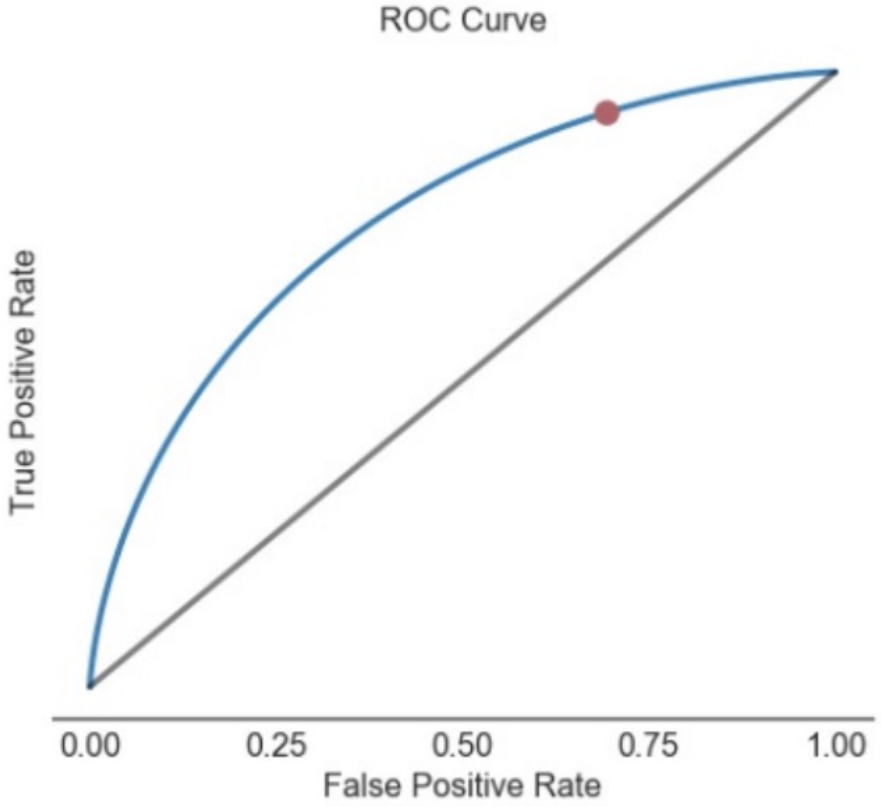
ROC Curve vs AUC



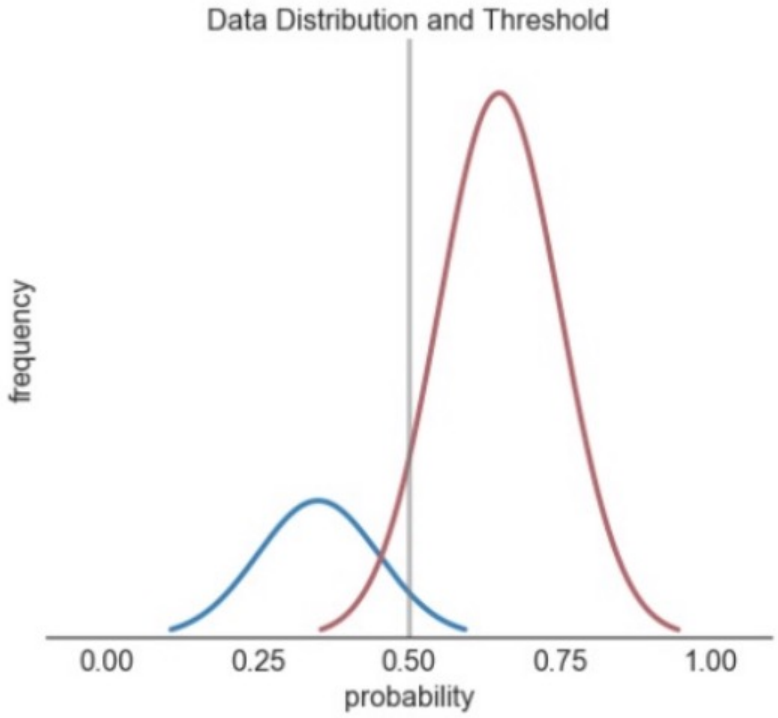


Sharpness of ROC Curve



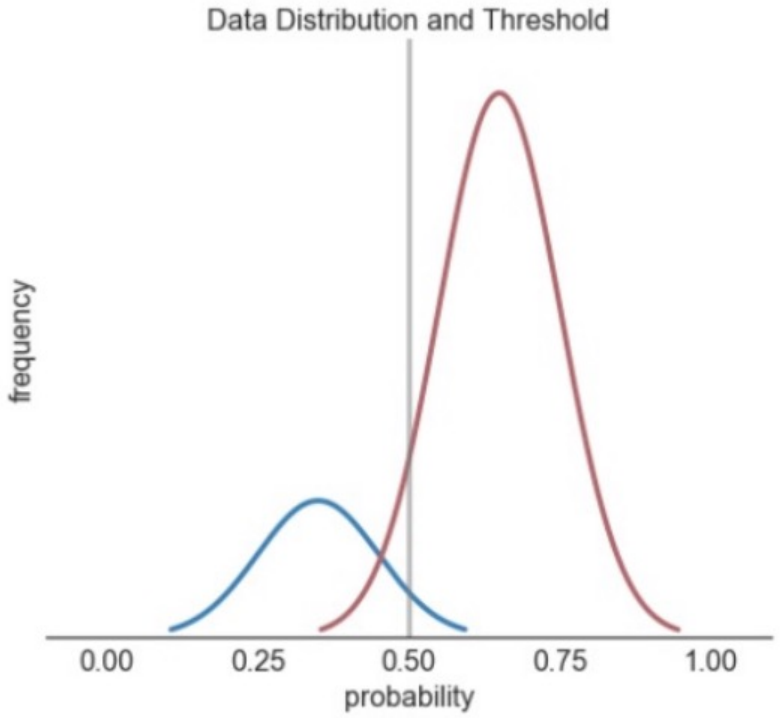


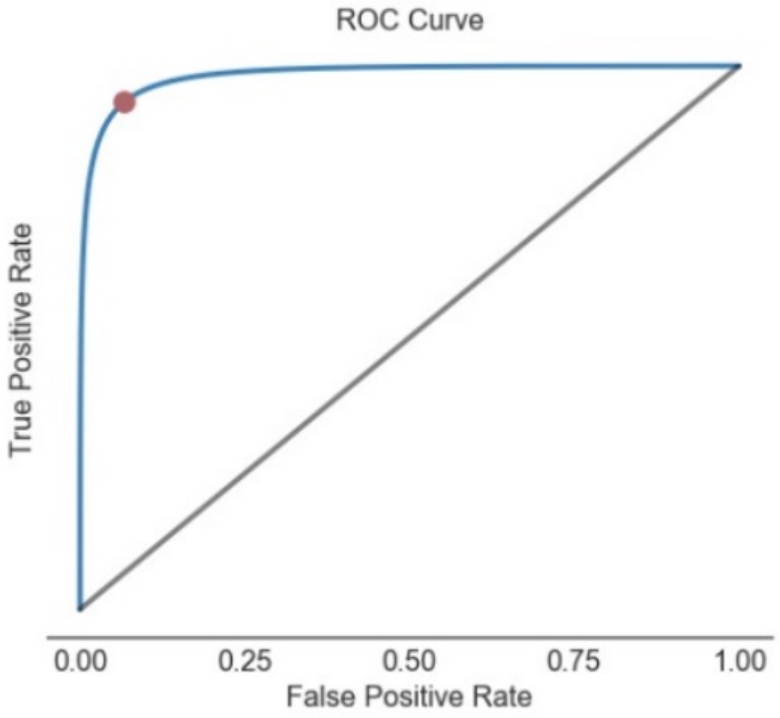
Non-Identical Distributions - Skew



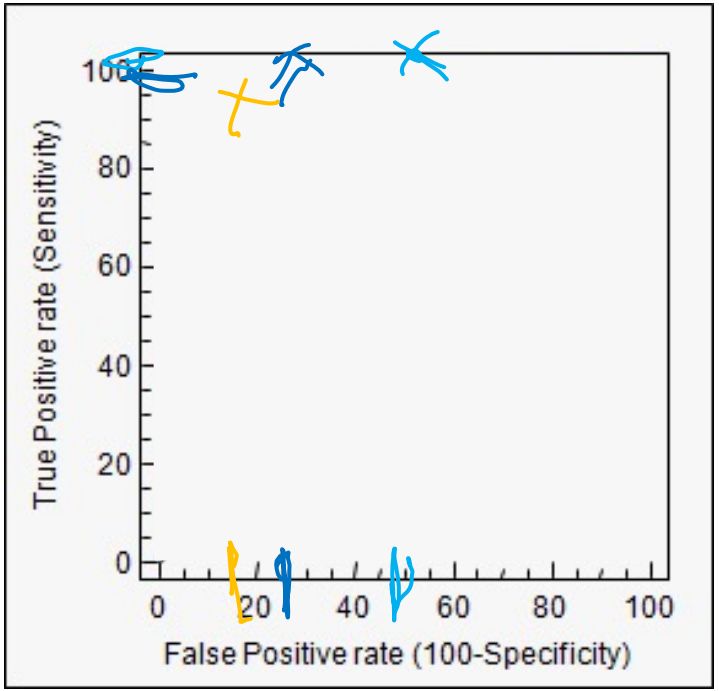
Non-Identical Distributions - Skew

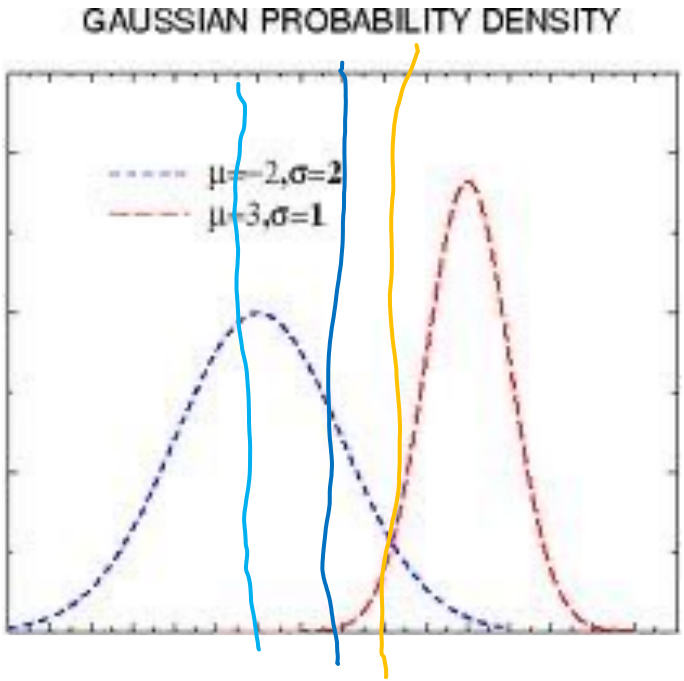
END





Non-Identical Distributions – different dev.



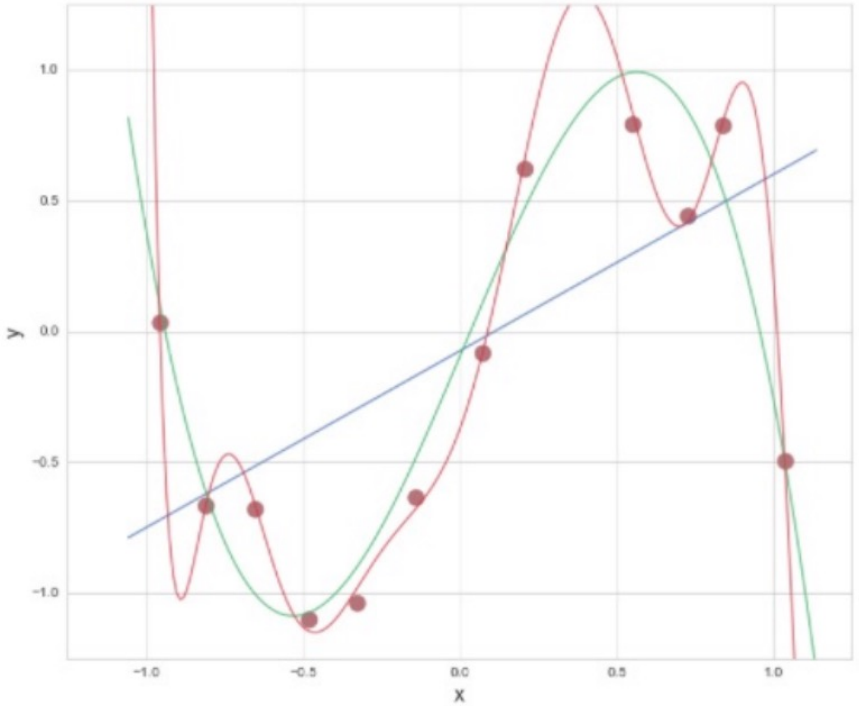


Linear Regression – Housing Market

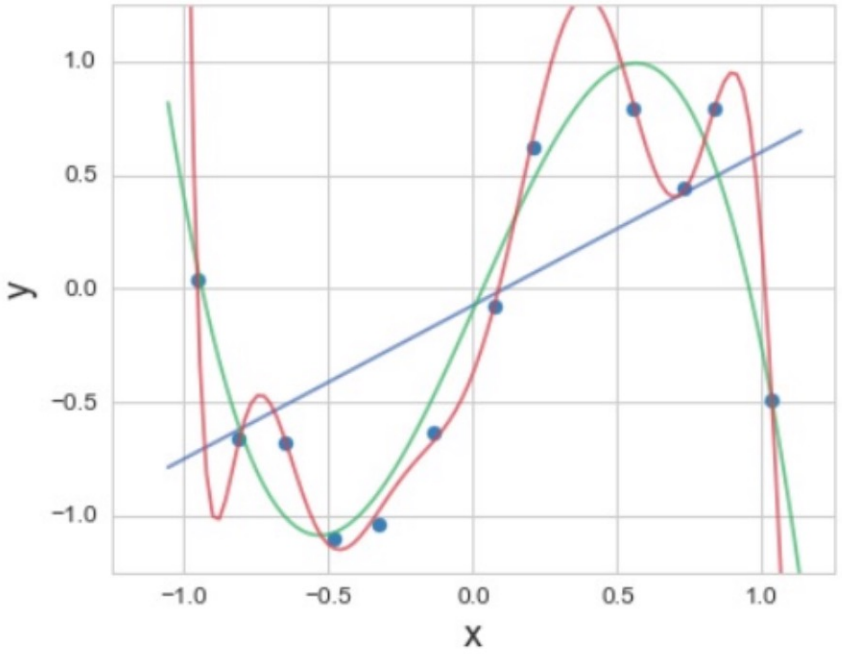
Accuracy of our models

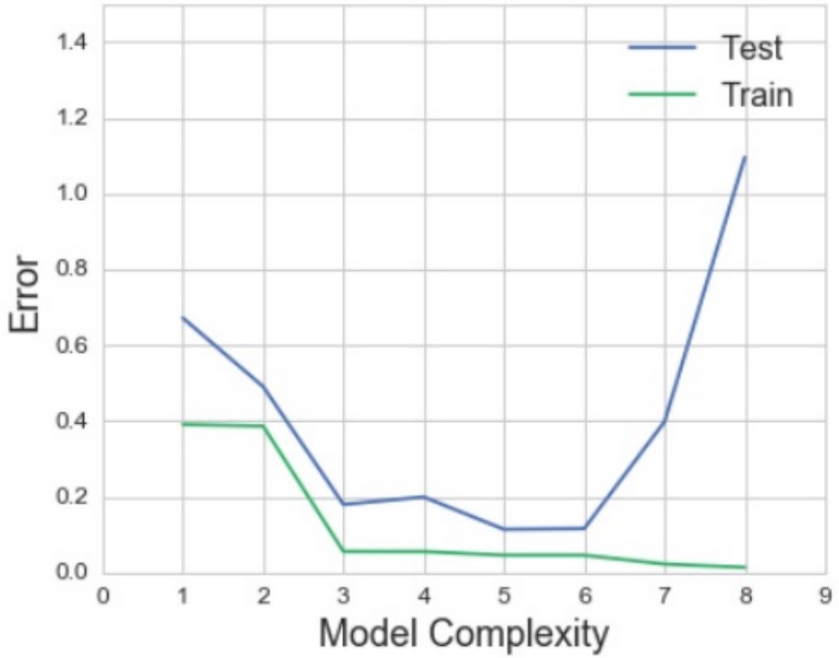
We can form unlimited factors

How do we know when we have enough features to capture the underlying model?



Accuracy of our models



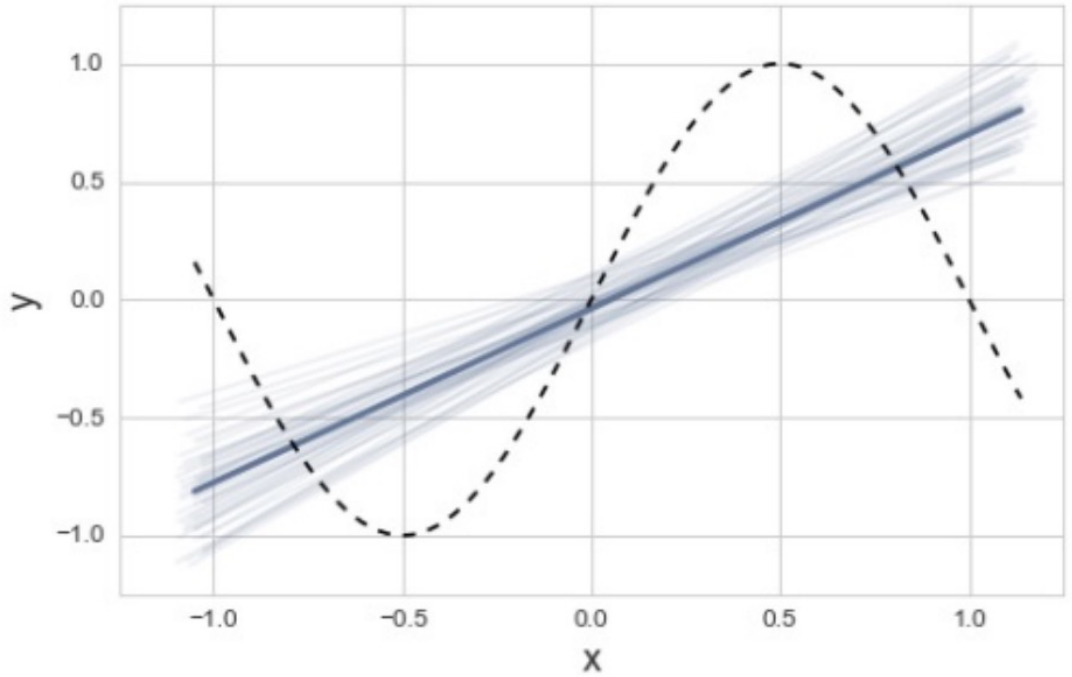


Bias vs. Variance

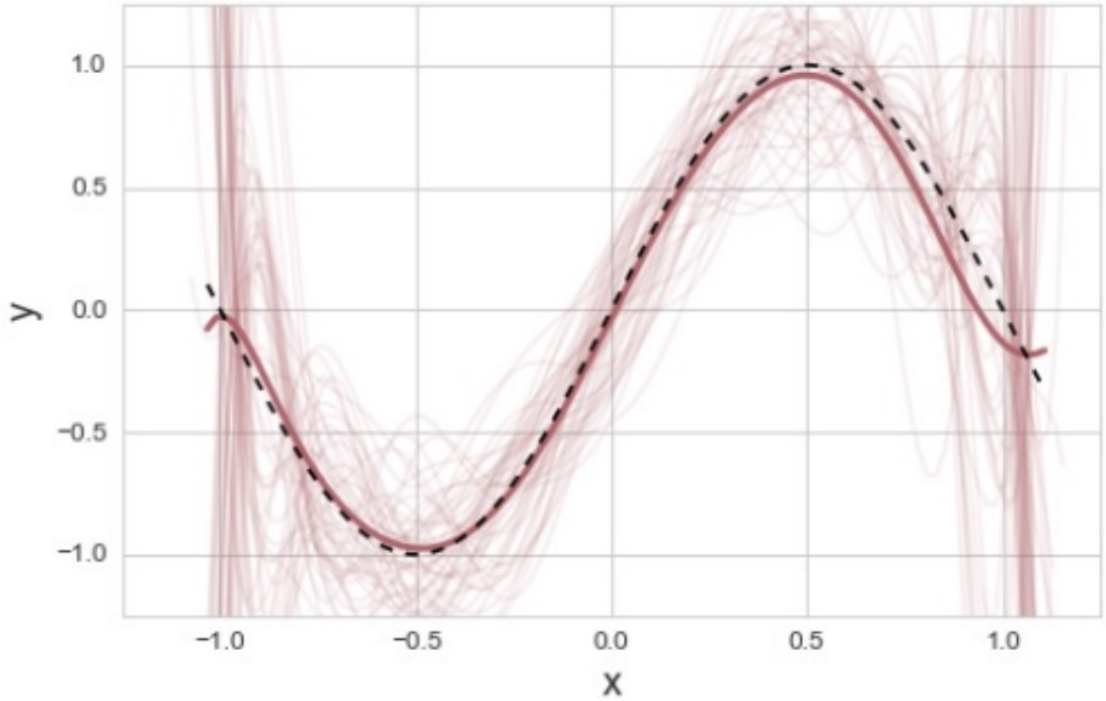
Bias – The inability of the model to represent the underlying data

Variance – The sensitivity of the model to the training data

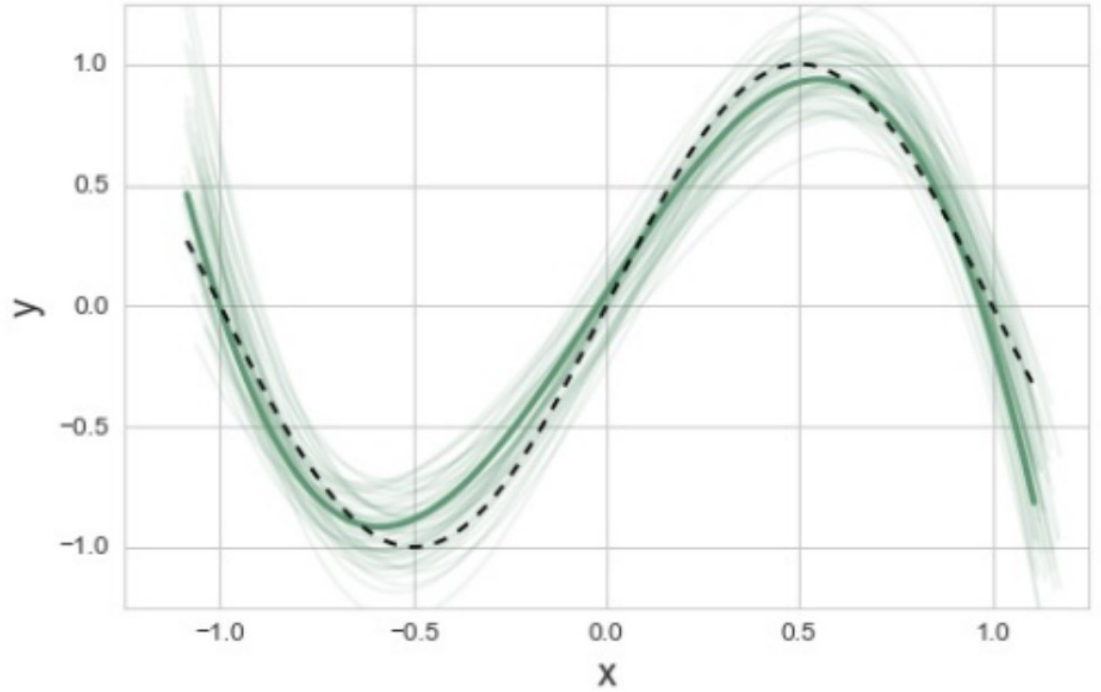
Bias - Variance Tradeoff



Bias – Variance Tradeoff

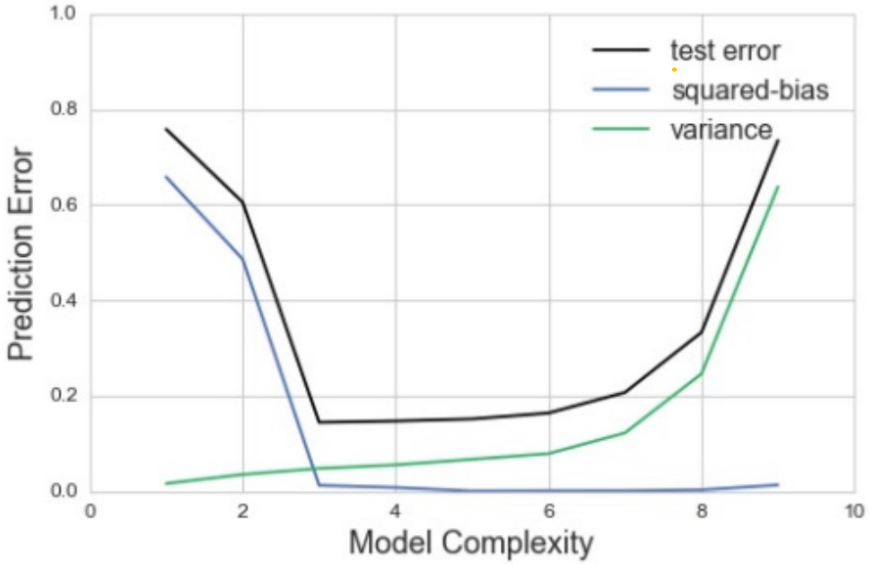


Bias – Variance Tradeoff



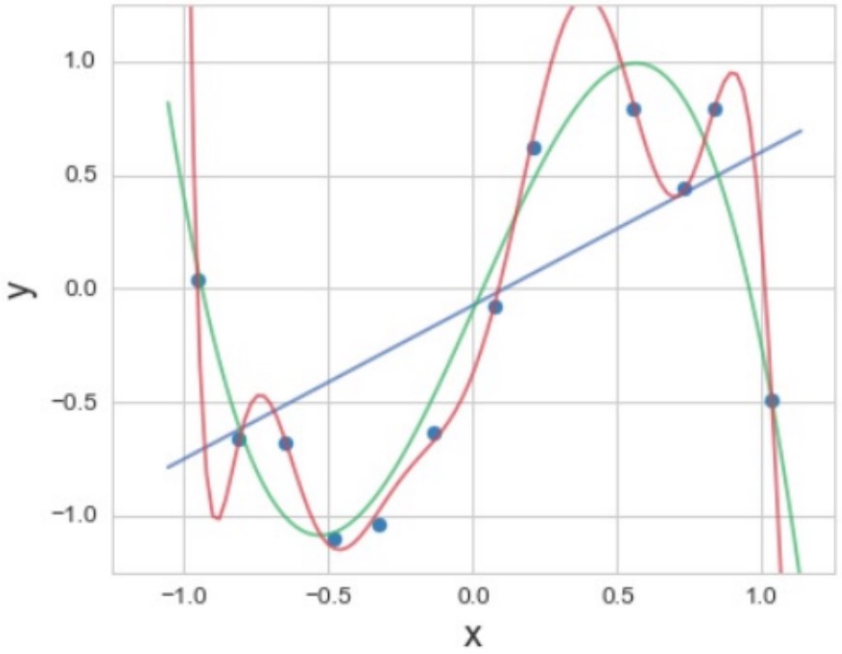
Bias – Variance Tradeoff

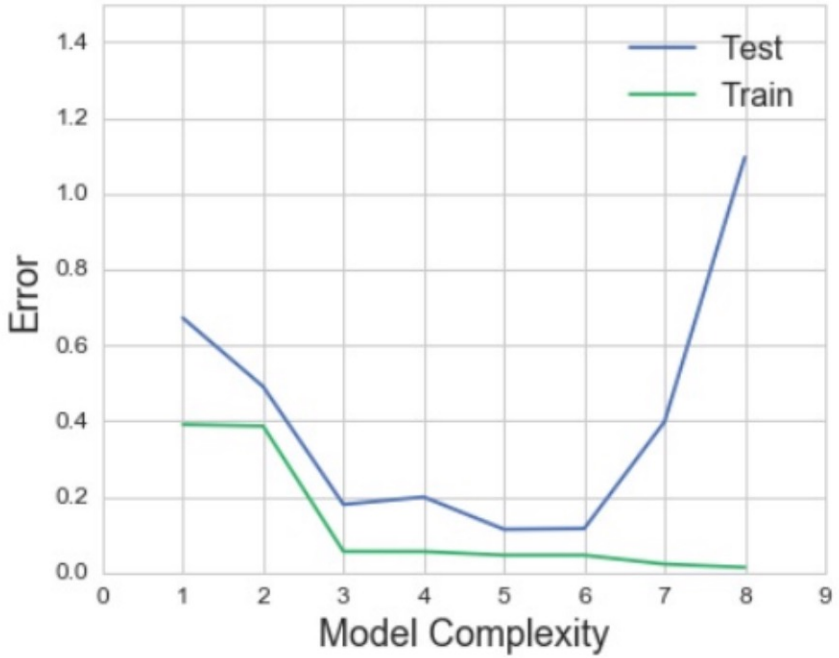
Error = $||Y - \hat{Y}||^2$ can be decomposed into Bias, Variance, and irreducible error



Feature Selection & Shrinkage

Feature Explosion Problem





Two Approaches – Subsets vs. Shrinkage

Subsets

- Easier to interpret

Shrinkage

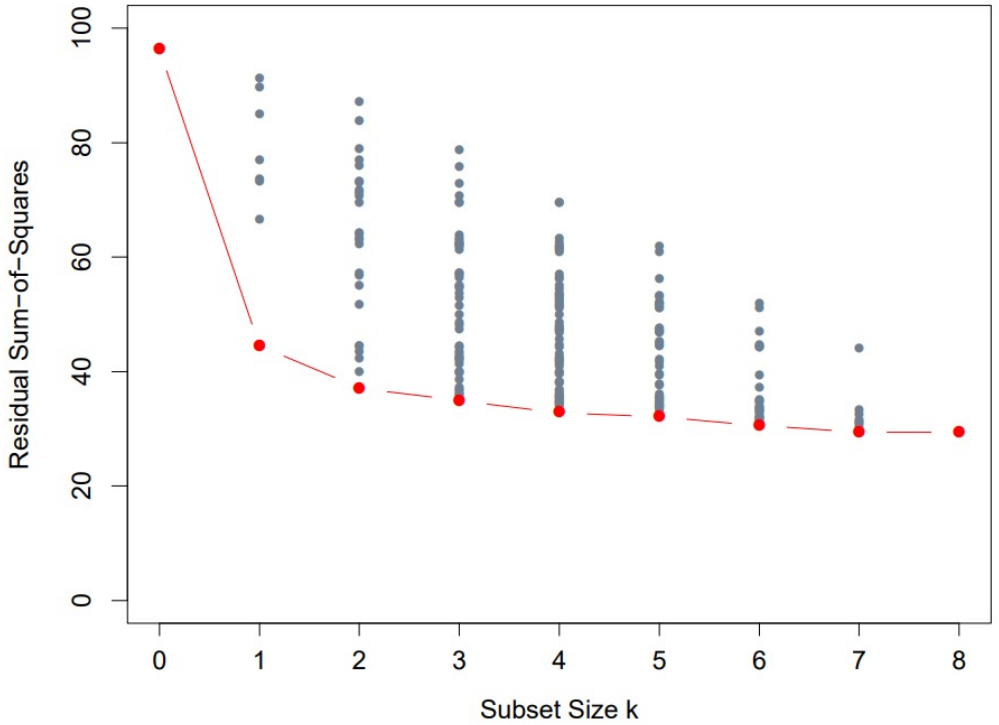
- Help avoid bias

Feature Selection – Best Subset

Choose the k feature set size you want

Create all subsets of size k

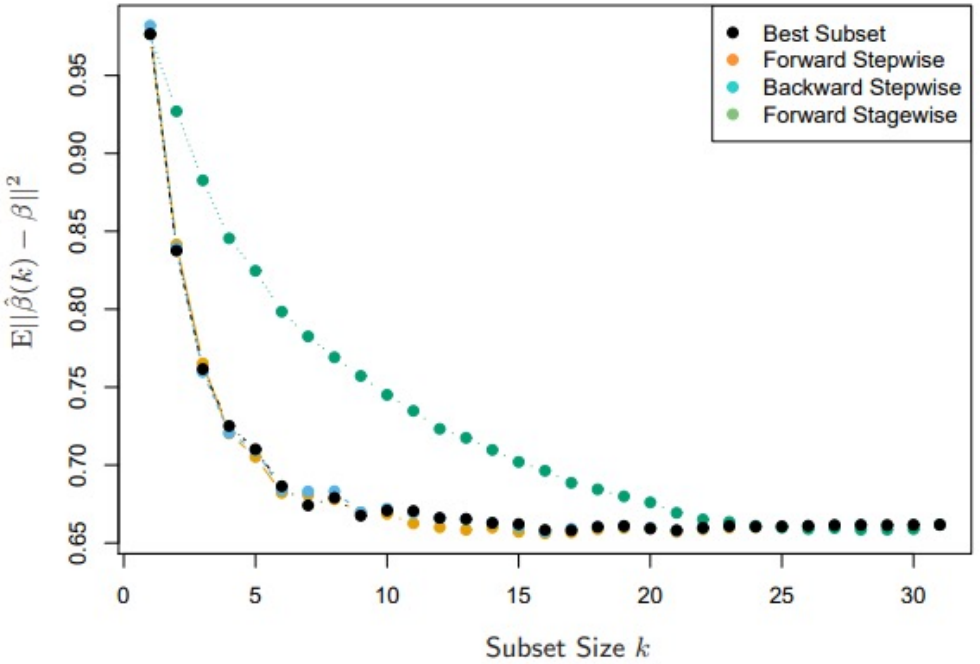
Find the RSS minimization for your subset



Feature Selection – Stepwise Selection

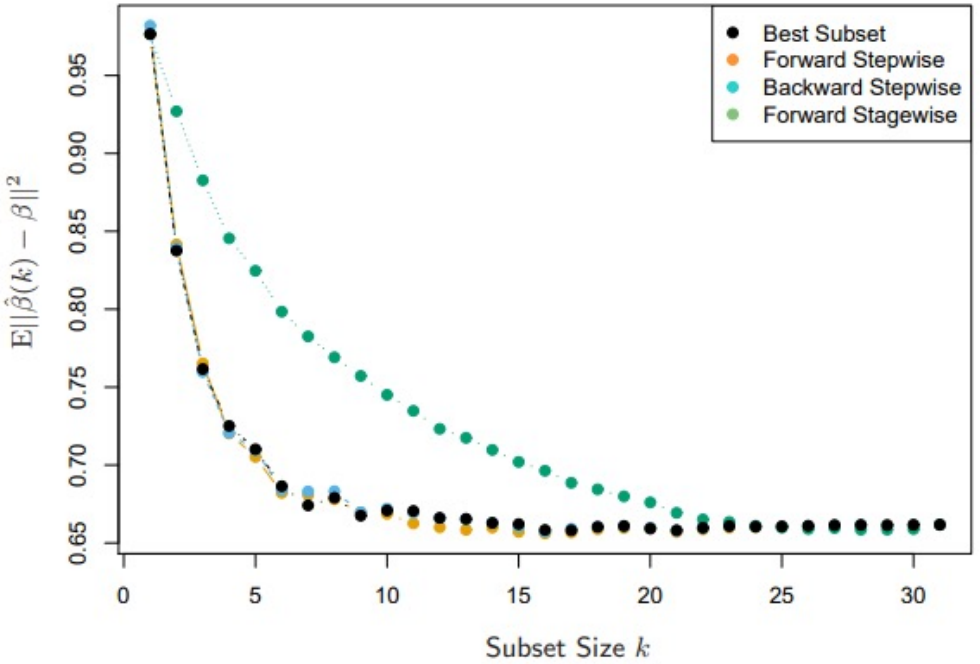
Forward-stepwise:
Start with an empty model, add the most predictive feature, rinse, repeat.

Backward-stepwise:
Start with a full model, remove the least predictive feature, rinse, repeat.



Feature Selection – Stagewise Selection

Start with an empty model, choose the feature that best correlates to your residual, update this feature using the coefficient.



Today's Dataset – Housing Prices

X_1	X_2	X_3	X_4	Y
Size (Sq. Ft.)	**# Bed**	**# Bath**	**Year Built**	**Price ($)**
1200	1	1.5	1998	200,000
1800	2	2	1985	450,000
800	1	1	2017	250,000
2500	3	2	1975	500,000
2800	4	2.5	1983	400,000
…	…	…	…	…

Two Approaches – Subsets vs. Shrinkage

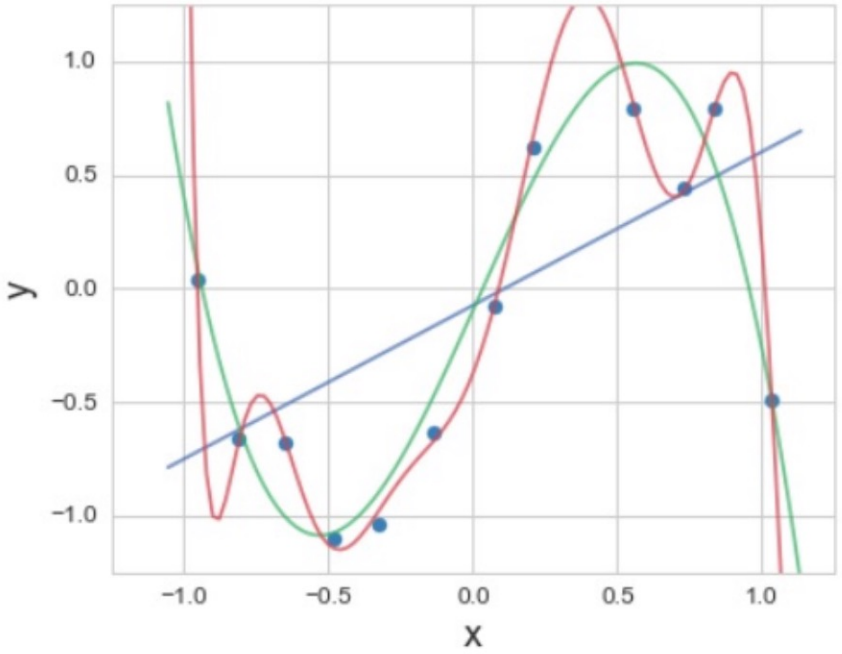
Subsets

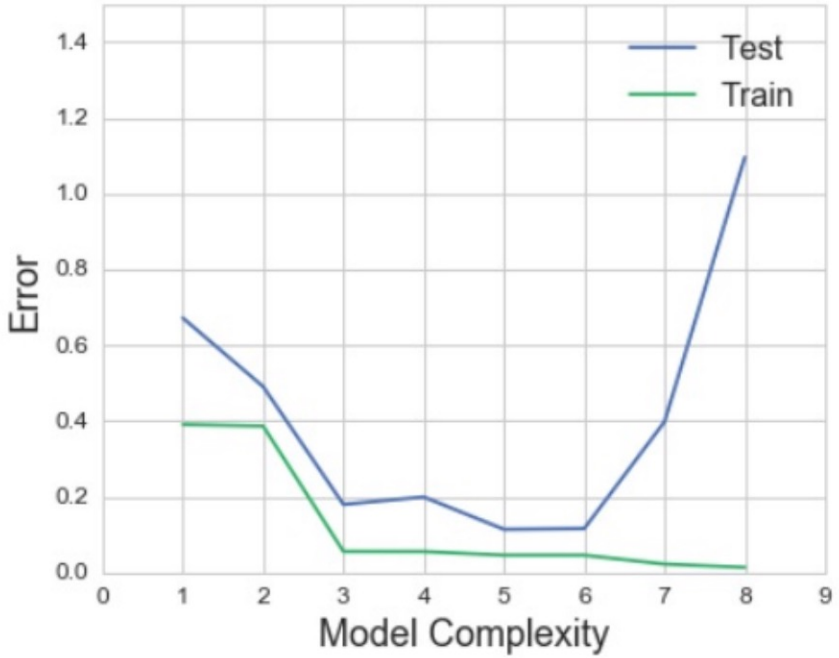
- Easier to interpret

Shrinkage

- Help avoid bias

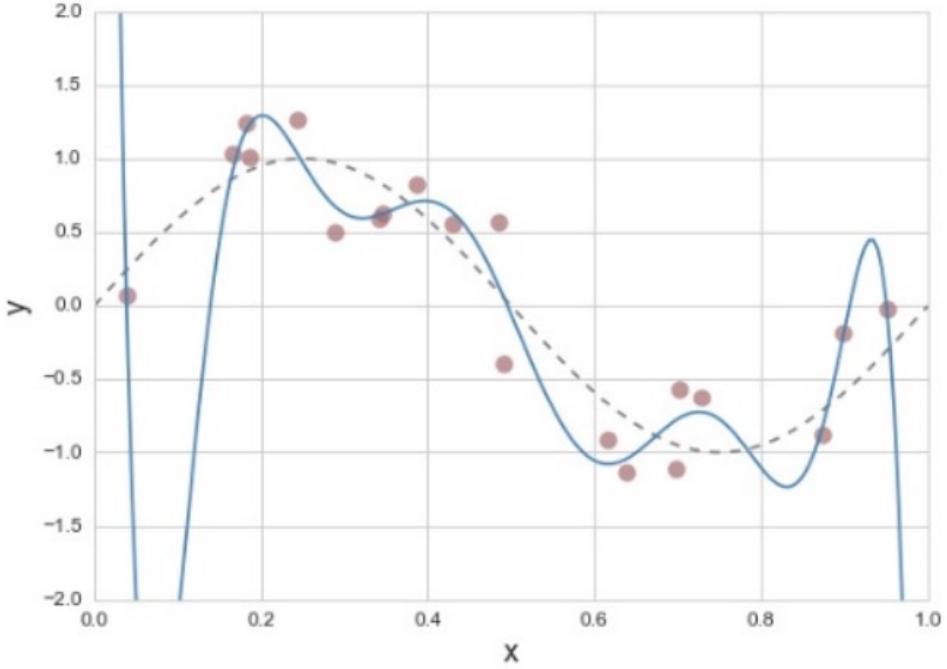
Feature Explosion Problem

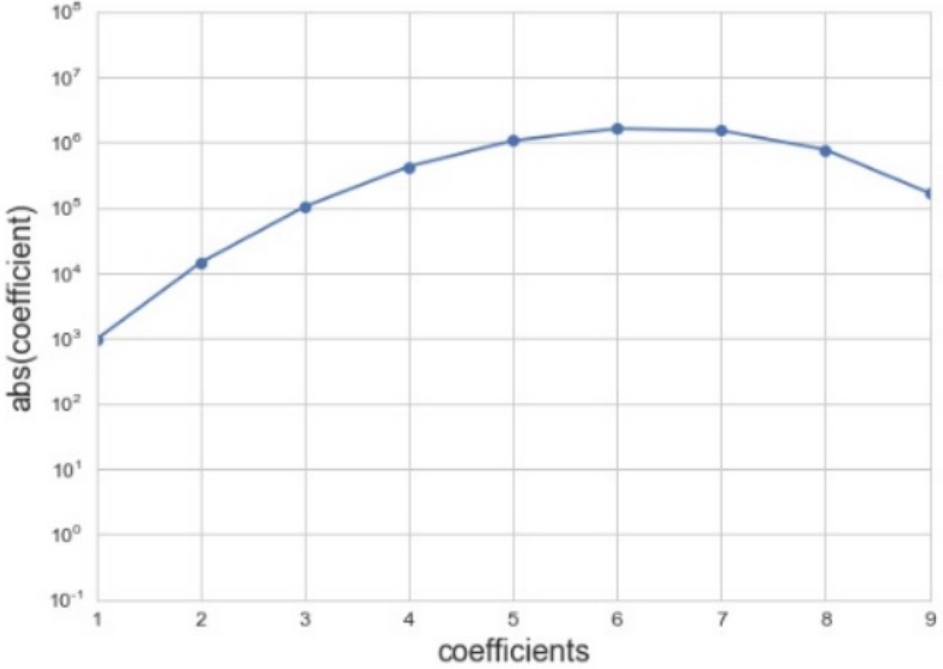




Feature Explosion Problem

True distribution y = sin(2*π*x)





Minimizing RSS

$RSS = ||Y - Xw||^2$

$\hat{w} = \text{argmin}_w(||Y - Xw||^2)$

How have we solved large weights / overfitting problems before?

Optimizing Weights – Penalty Term (Log Reg, Week 5)

$$NLL(w) = -\Sigma_{i=1 \rightarrow n}[y_i * \log[(1/1+e^{-(w \cdot xi)})] + (1-y_i) * \log[(1-(1/1+e^{-w \cdot xi}))]]$$

Add a penalty term

$$\lambda * \Sigma_{k=1 \rightarrow D} w^2_k$$

$$NLL(w) = -\Sigma_{i=1 \rightarrow n}[y_i * \log[(1/1+e^{-(w \cdot xi)})] + (1-y_i) * \log[(1-(1/1+e^{-w \cdot xi}))]] + \lambda * \Sigma_{k=1 \rightarrow D} w^2_k$$

NOTE: Don't penalize the bias weight w_0

Minimizing RSS – With Penalty

$RSS = ||Y - Xw||^2$

$\hat{w} = \text{argmin}_w(||Y - Xw||^2)$

Add a penalty term

$\hat{w} = \text{argmin}_w(||Y - Xw||^2 + \lambda * \Sigma_{k=1 \rightarrow D} \text{penalty}(w_k))$

How do we choose our penalty function?
How do we choose our tuning parameter λ?

Choosing Penalty Function

$\lambda * \Sigma_{k=1 \rightarrow D} w^2_k$

Choosing Penalty Function

$\lambda * \Sigma_{k=1 \rightarrow D} w^2_k$

The same penalty we used in Logistic Regression

Choosing Penalty Function

$\lambda * \Sigma_{k=1 \rightarrow D} w^2_k$

The same penalty we used in Logistic Regression

L2 Norm (related to Euclidian distance from origin)

Choosing Penalty Function

$\lambda * \Sigma_{k=1 \rightarrow D} w^2_k$

The same penalty we used in Logistic Regression

L2 Norm (related to Euclidian distance from origin)

Ridge Regression

Side Note: Feature Scales

Build some 1D models (manually. Yes, you're guesstimating. Yes they are not well trained models.)

Model 1) Sq ft → Price

Model 2) # Bed → Price

What is your weight?

(What is the scale of your weight, how many trailing 0's?)

X_1	X_2	X_3	X_4	Y
Size (Sq. Ft.)	**# Bed**	**# Bath**	**Year Built**	**Price ($)**
1200	1	1.5	1998	200,000
1800	2	2	1985	450,000
800	1	1	2017	250,000
2500	3	2	1975	500,000
2800	4	2.5	1983	400,000
…	…	…	…	…

Side Note: Feature Scales

Build some 1D models (manually. Yes, you're guesstimating. Yes they are not well trained models.)

X_1	X_2	X_3	X_4	Y
Size (Sq. Ft.)	**# Bed**	**# Bath**	**Year Built**	**Price ($)**
1200	1	1.5	1998	200,000
1800	2	2	1985	450,000
800	1	1	2017	250,000
2500	3	2	1975	500,000
2800	4	2.5	1983	400,000
…	…	…	…	…

Model 1) Sq ft → Price

w = 1500?

Model 2) # Bed → Price

w = 150,000?

What is your weight?

What is your RSS? $||Y - Xw||^2$

Side Note: Feature Scales

Build some 1D models, but add a penalty term, w^2

X_1	X_2	X_3	X_4	Y
Size (Sq. Ft.)	**# Bed**	**# Bath**	**Year Built**	**Price ($)**
1200	1	1.5	1998	200,000
1800	2	2	1985	450,000
800	1	1	2017	250,000
2500	3	2	1975	500,000
2800	4	2.5	1983	400,000
…	…	…	…	…

Model 1) Sq ft → Price

Model 2) # Bed → Price

What is your weight?

What is your RSS? $||Y - Xw||^2 + w^2$

Side Note: Feature Scaling

X_1	X_2	X_3	X_4	Y
Size (Sq. Ft.)	**# Bed**	**# Bath**	**Year Built**	**Price ($)**
1200	1	1.5	1998	200,000
1800	2	2	1985	450,000
800	1	1	2017	250,000
2500	3	2	1975	500,000
2800	4	2.5	1983	400,000
…	…	…	…	…

→

X_1	X_2	X_3	X_4	Y
Size / 1,000	**# Bed**	**# Bath**	**Years Ago / 10**	**Price ($)**
1.2	1	1.5	2.1	200,000
1.8	2	2	3.4	450,000
.8	1	1	.2	250,000
2.5	3	2	4.4	500,000
2.8	4	2.5	3.7	400,000
…	…	…	…	…

Side^2 Note: Negative Weights

X_1	X_2	X_3	X_4	Y
Size / 1,000	**# Bed**	**# Bath**	**Years Ago / 10**	**Price ($)**
1.2	1	1.5	2.1	200,000
1.8	2	2	3.4	450,000
.8	1	1	.2	250,000
2.5	3	2	4.4	500,000
2.8	4	2.5	3.7	400,000
…	…	…	…	…

Side Note: Feature Scaling

X_1	X_2	X_3	X_4	Y
Size (Sq. Ft.)	**# Bed**	**# Bath**	**Year Built**	**Price ($)**
1200	1	1.5	1998	200,000
1800	2	2	1985	450,000
800	1	1	2017	250,000
2500	3	2	1975	500,000
2800	4	2.5	1983	400,000
…	…	…	…	…

→

X_1	X_2	X_3	X_4	Y
Size / 1,000	**# Bed**	**# Bath**	**Years Ago / 10**	**Price ($)**
1.2	1	1.5	2.1	200,000
1.8	2	2	3.4	450,000
.8	1	1	.2	250,000
2.5	3	2	4.4	500,000
2.8	4	2.5	3.7	400,000
…	…	…	…	…

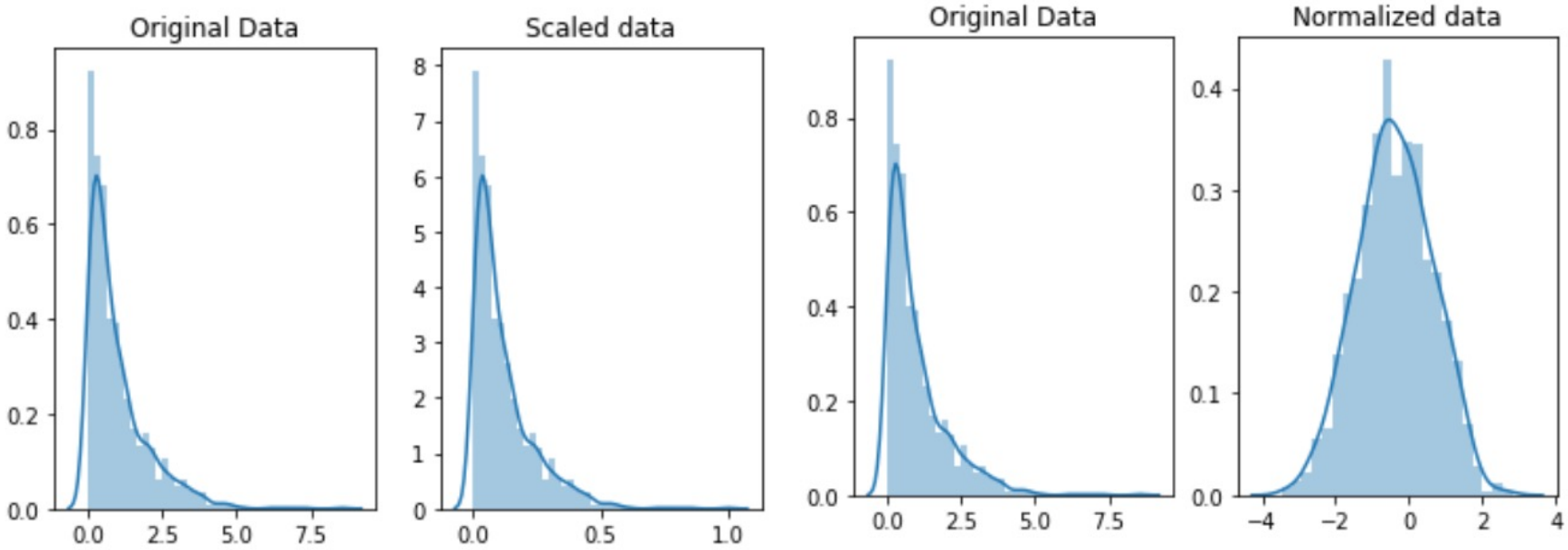
Side Note: Feature Scaling

X_1	X_2	X_3	X_4	Y
Size (Sq. Ft.)	**# Bed**	**# Bath**	**Year Built**	**Price ($)**
1200	1	1.5	1998	200,000
1800	2	2	1985	450,000
800	1	1	2017	250,000
2500	3	2	1975	500,000
2800	4	2.5	1983	400,000
…	…	…	…	…

→

X_1	X_2	X_3	X_4	Y
Size / 1,000	**# Bed**	**# Bath**	**Years Ago / 10**	**Price ($) / 100,000**
1.2	1	1.5	2.1	2
1.8	2	2	3.4	4.5
.8	1	1	.2	2.5
2.5	3	2	4.4	5
2.8	4	2.5	3.7	4
…	…	…	…	…

Side² Note: Feature Scaling vs. Normalizing



Minimizing RSS – With Penalty

$RSS = ||Y - Xw||^2$

$\hat{w} = \text{argmin}_w(||Y - Xw||^2)$

Add a penalty term

$\hat{w} = \text{argmin}_w(||Y - Xw||^2 + \lambda * \Sigma_{k=1 \rightarrow D} \text{penalty}(w_k))$

How do we choose our penalty function?
How do we choose our tuning parameter λ?

Choosing a tuning parameter λ

$\hat{w} = \text{argmin}_w (||Y - Xw||^2 + \lambda * \Sigma_{k=1 \rightarrow D} w^2_k)$

$\lambda = -\infty$ (or a big negative):

$\lambda = -.0000001$ (or a small negative):

$\lambda = 0$:

$\lambda = +.0000001$ (or a small positive):

$\lambda = +\infty$ (or a big positive):

Choosing a tuning parameter λ

$\hat{w} = \text{argmin}_w(||Y - Xw||^2 + \lambda * \Sigma_{k=1 \rightarrow D} w^2_k)$

$\lambda = -\infty$ (or a big negative): *We seek out large weights, spirals*
$\lambda = -.001$ (or a small negative): *We seek out large weights, spirals*
$\lambda = 0$: *We're just removing the penalty term!*
$\lambda = +.001$ (or a small positive): *We seek out small weights*
$\lambda = +\infty$ (or a big positive): *We make our weights = 0, penalty outweighs accuracy tuning*

How have we chosen a scaling factor λ in the past?

Choosing a Learning Rate (Week 5)

Try some out!

- Do a few rounds of SGD on a few small rates (1, .5, .1, .01, perhaps)
- Use the one that is showing the best improvement

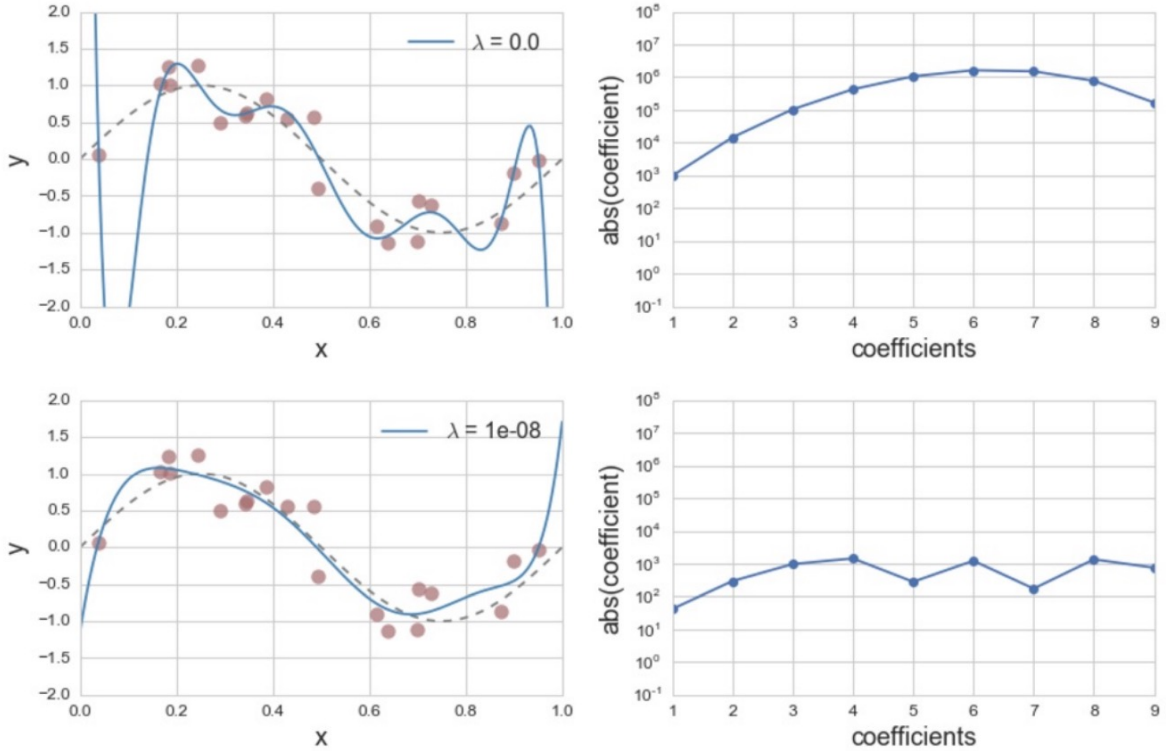
Scale back over time

- You can easily start bouncing around the true minimum

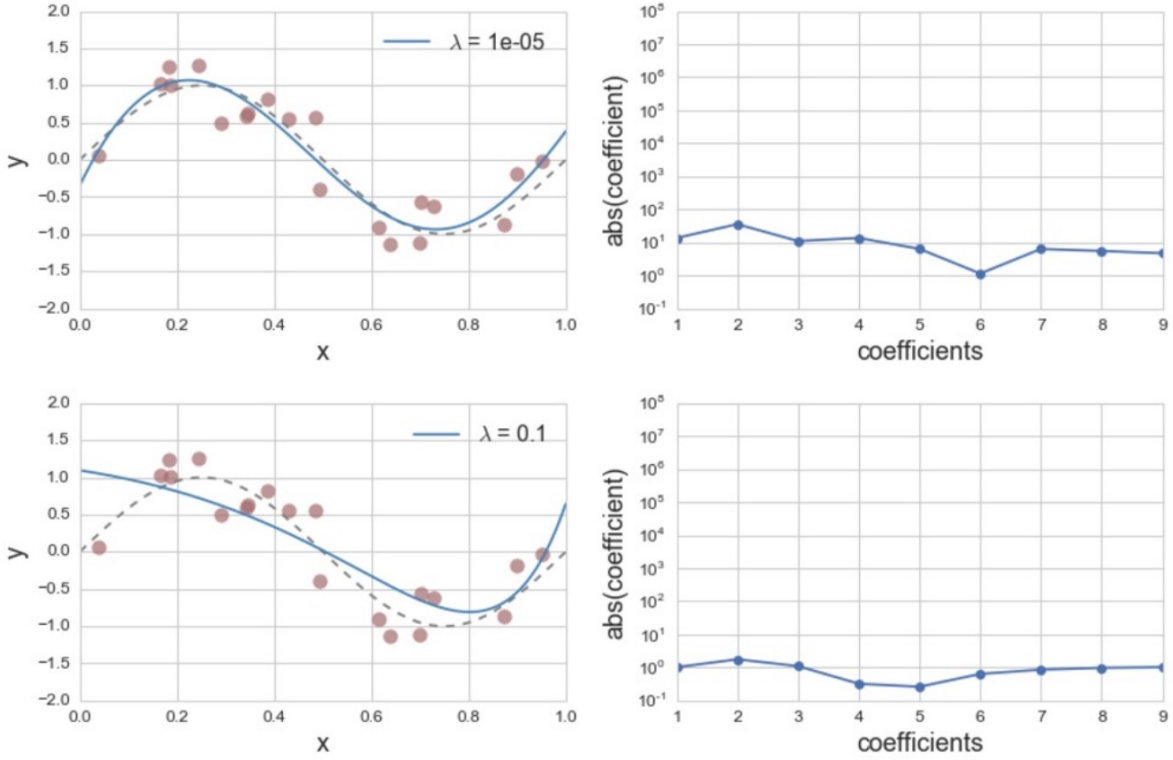
Scale rates for each feature

- *this is an advanced technique, we won't get into it today.*

Choosing a Tuning Parameter



Choosing a Tuning Parameter



Choosing a Tuning Parameter λ

Try some out!

- Do a few rounds of λ on a few small rates (1, .1, .01, .001, .0001, .00001, .000001, .0000001)
- Use cross-validation

How do we determine the “best” λ?

Bias vs. Variance vs. λ

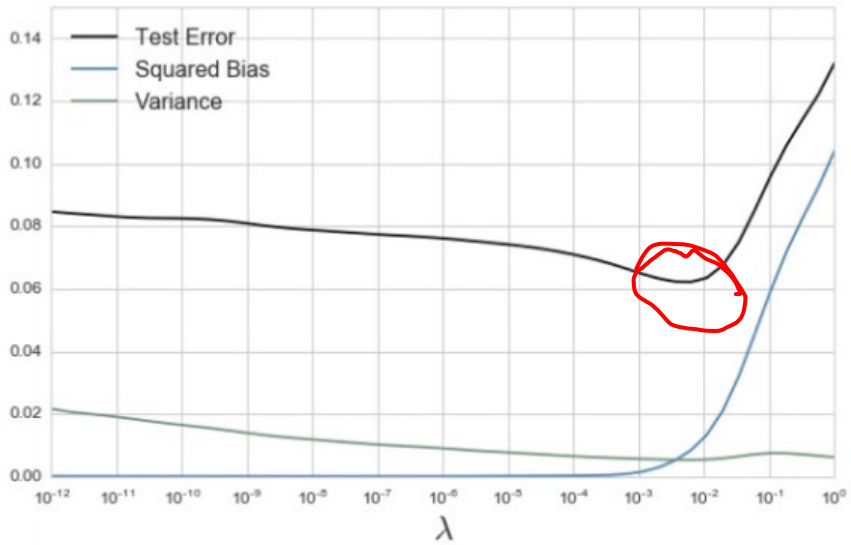
Bias vs. Variance vs. λ

Bias increases with simpler models, therefore with higher λ

Variance increases with complex models, therefore with lower λ

Bias vs. Variance vs. λ

True distribution y = sin(2*π*x)



Choosing Penalty Function

$\lambda * \Sigma_{k=1 \rightarrow D} w^2_k$

The same penalty we used in Logistic Regression

L2 Norm (related to Euclidian distance from origin)

Ridge Regression

Choosing Penalty Function

$\lambda * \Sigma_{k=1 \rightarrow D} |w_k|$

Choosing Penalty Function

$\lambda * \Sigma_{k=1 \rightarrow D} |w_k|$

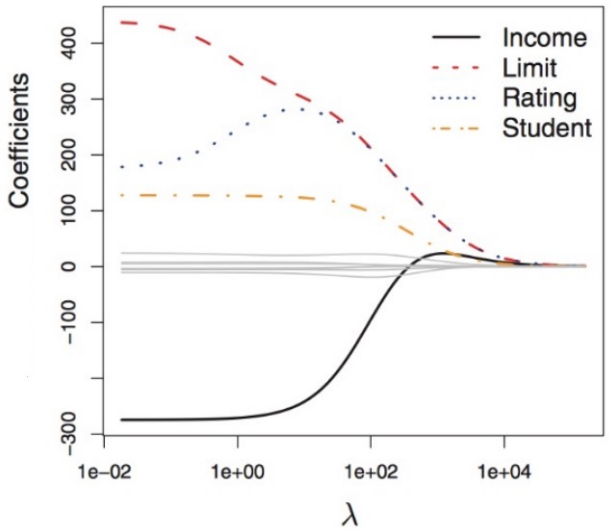
L1 Norm (Manhattan distance from origin)

Lasso Regression

Model Behavior – Ridge vs. Lasso

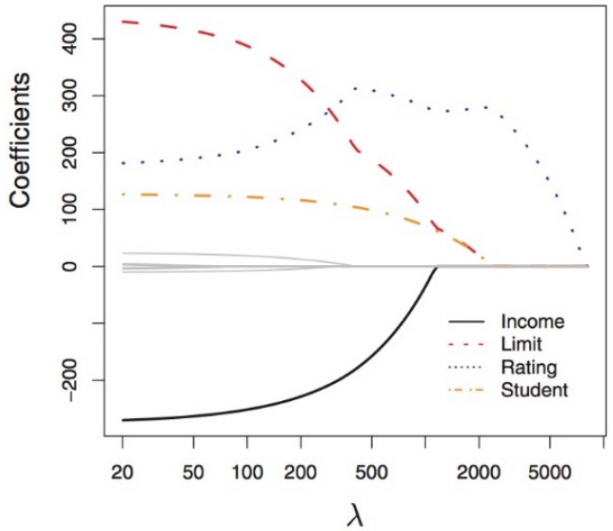
Ridge Regression

- uniform converge



Lasso Regression

- different convergence
- collapses at 0



Understanding Model Behavior

$\min_w(||Y-Xw||^2 + \lambda^*\Sigma_{k=1\rightarrow D}w^2_k)$ or $\min_w(||Y-Xw||^2 + \lambda^*\Sigma_{k=1\rightarrow D}|w_k|)$

Alternative Perspective

$\min_w(||Y-Xw||^2)$ such that

$\Sigma_{k=1\rightarrow D}w^2_k <= s$ (Ridge)

$\Sigma_{k=1\rightarrow D}|w_k| <= s$ (Lasso)

Understanding Model Behavior

Lasso Regression

$\min_w(||Y-Xw||^2)$ such that

$\Sigma_{k=1 \rightarrow D}|w_k| <= s$ (Lasso)

In the 2D Case...

$|w_1| + |w_2| <= s$

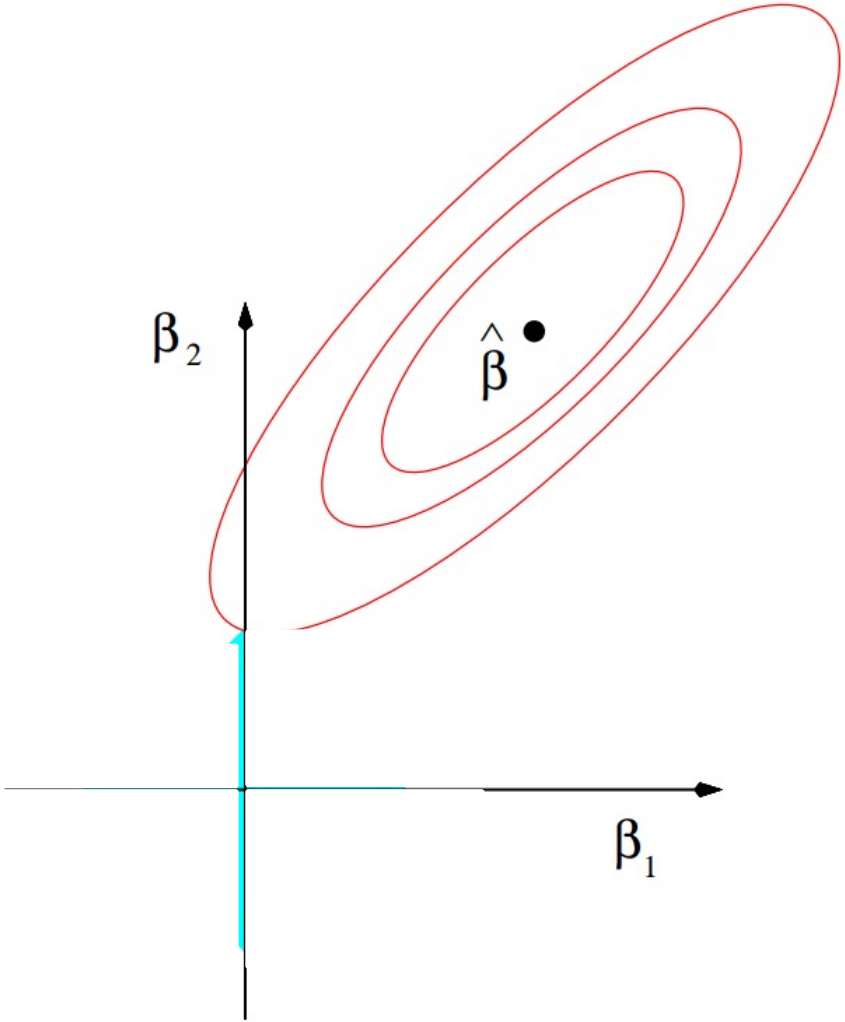
Ridge Regression

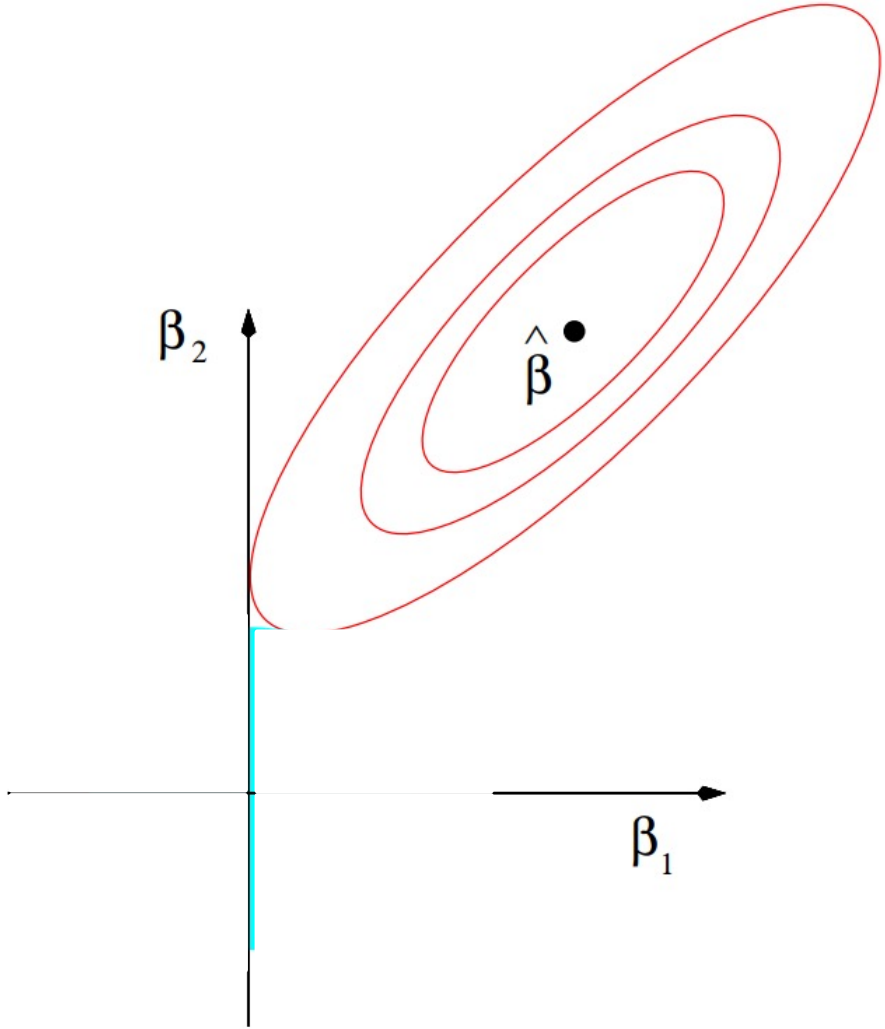
$\min_w(||Y-Xw||^2)$ such that

$\Sigma_{k=1 \rightarrow D} w_k <= s$ (Ridge)

In the 2D Case...

$w^2_1 + w^2_2 <= s$

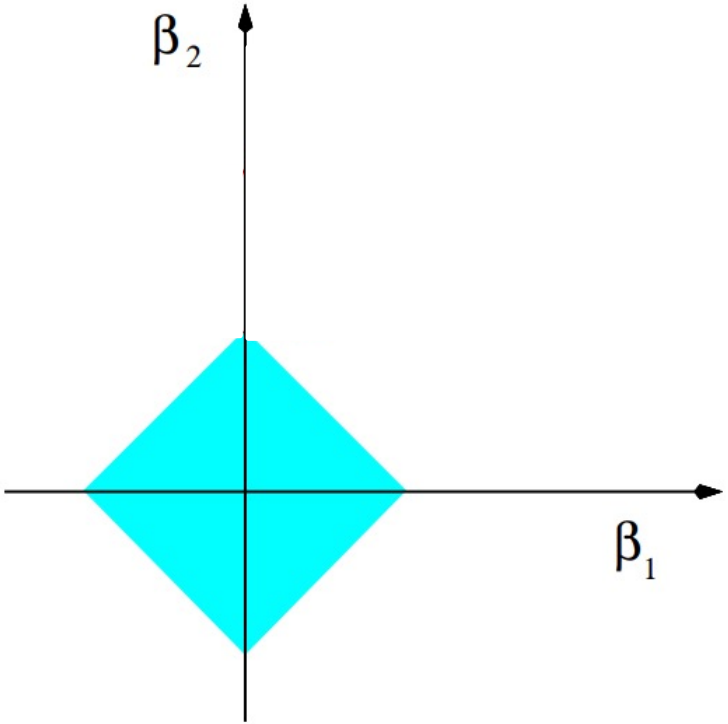

β_2
$\hat{\beta}$
β_1


β_2
$\hat{\beta}$
β_1

Lasso Regression

In the 2D Case…

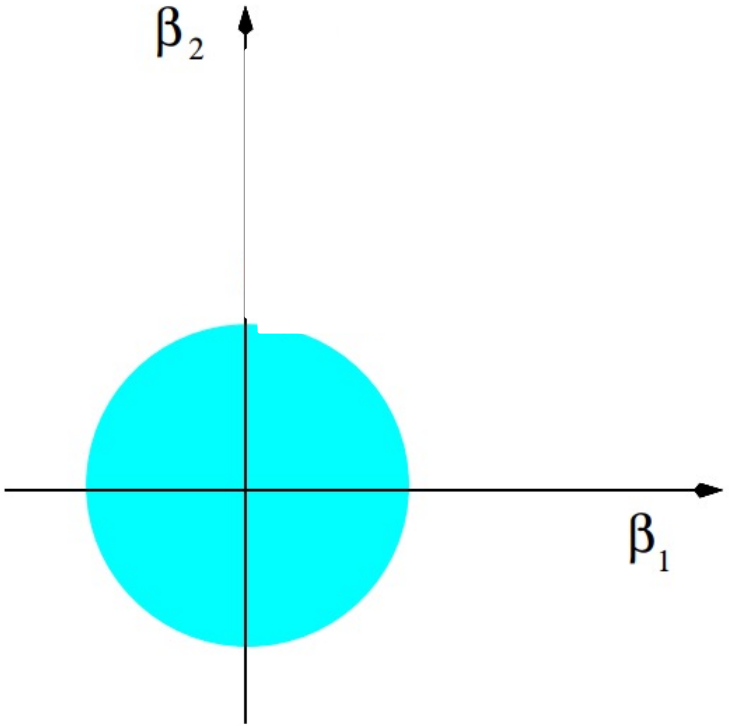
$|w_1| + |w_2| <= s$



Ridge Regression

In the 2D Case…

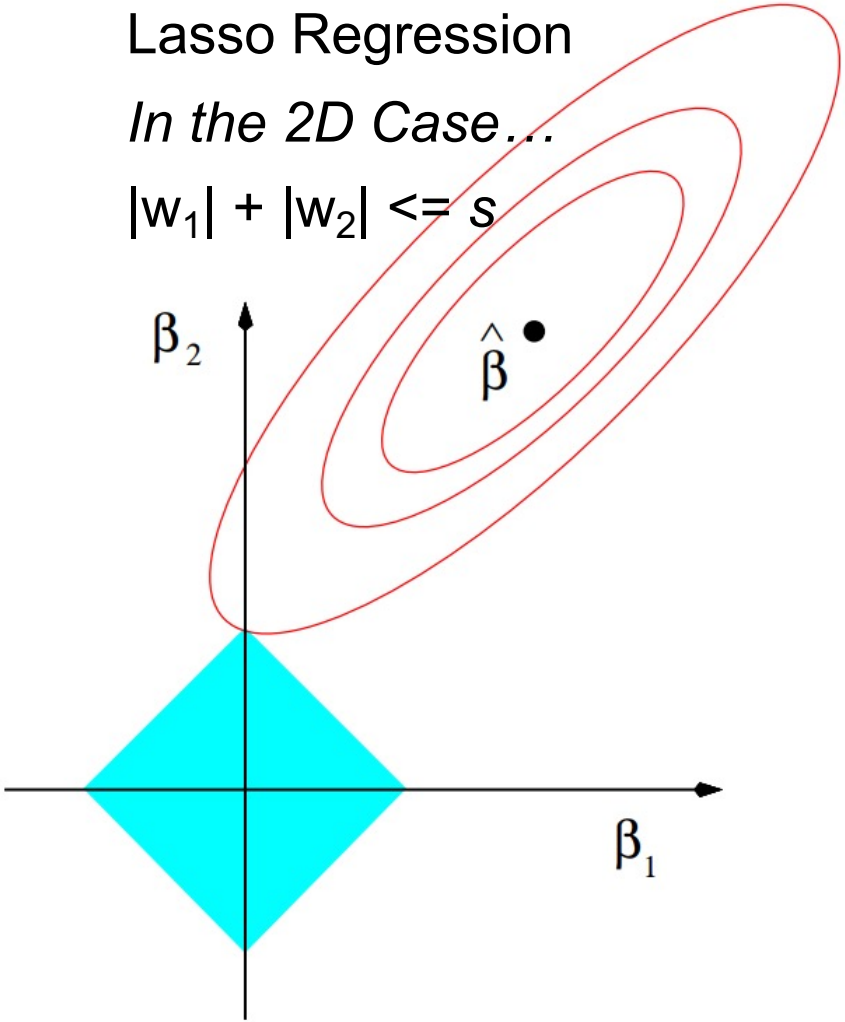
$w^2_1 + w^2_2 <= s$



Lasso Regression

In the 2D Case…

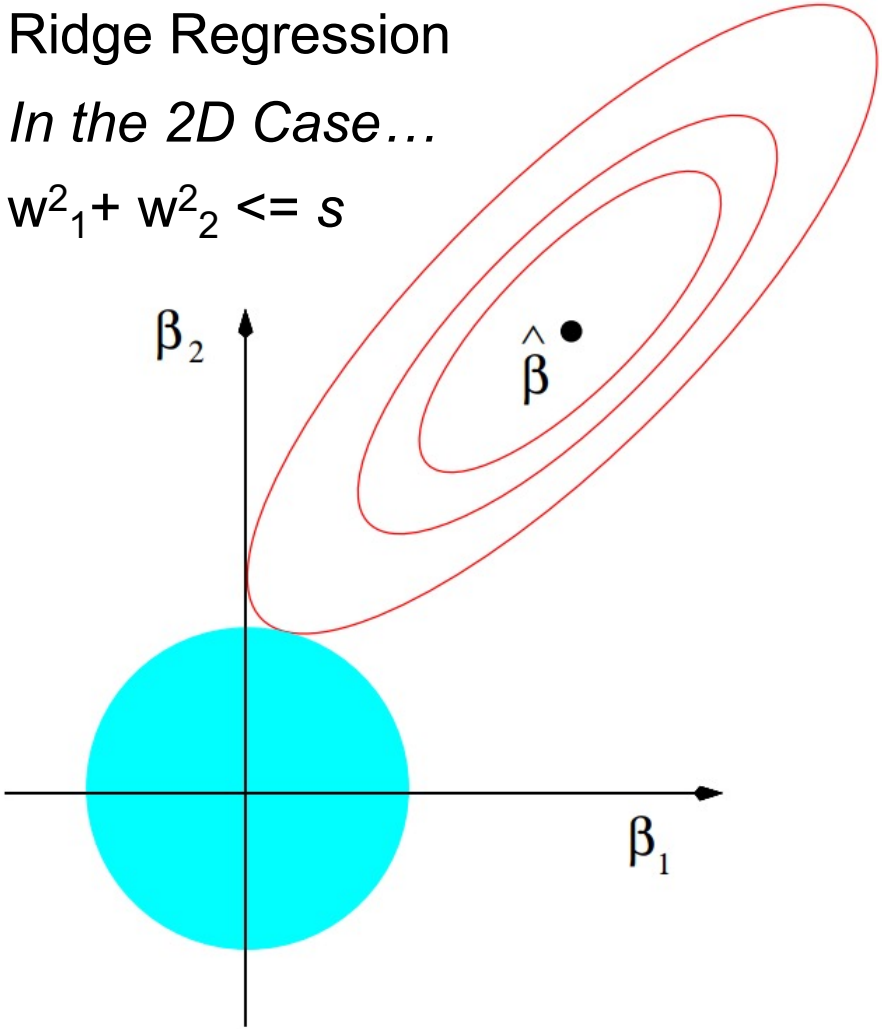
$|w_1| + |w_2| <= s$



Ridge Regression

In the 2D Case…

$w^2_1 + w^2_2 <= s$



Model Behavior – Lasso vs. Ridge Regression

Lasso Regression
- Tends to have optimum weights hit at 0 for some dimensions

Ridge Regression
- Creates low weights, (almost) never has optimum at 0

Other perspectives in reading...

Thursday

Course Logistics

- Problem Set 2: 10/7 @ 11:59PM
- Project Milestone 2 Class Feedback Process on Delay
  - Harder to do Asynchronously
  - Everyone deserves engagement and participation!

Reporting Model Goodness

Various approaches explored in this class

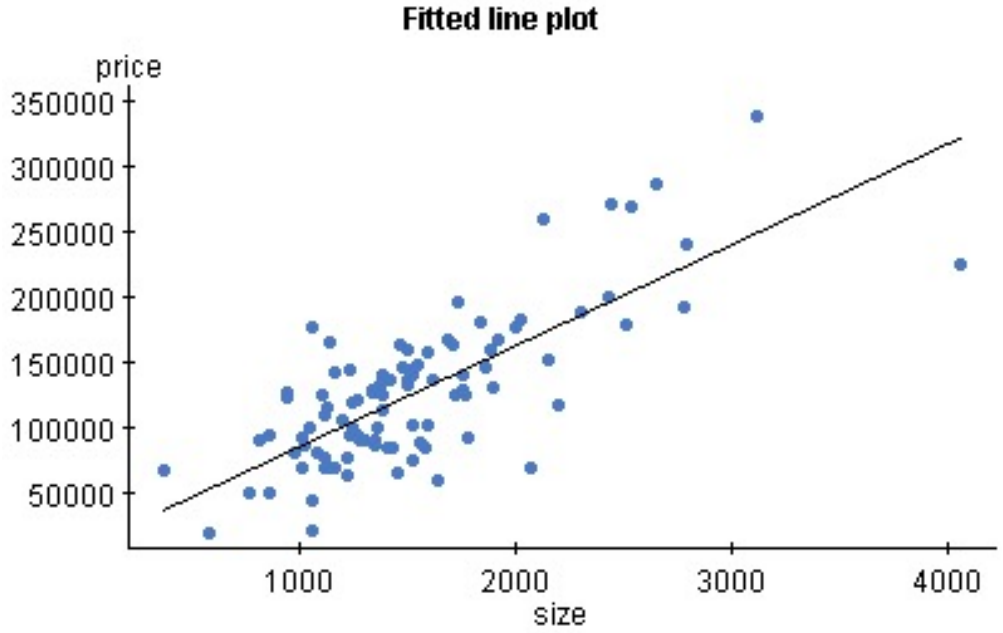
- Accuracy (correct vs. incorrect classification)
- ROC Curve (false positives & false negatives)

Are these relevant to a regression problem?

What have we been using so far to measure goodness?

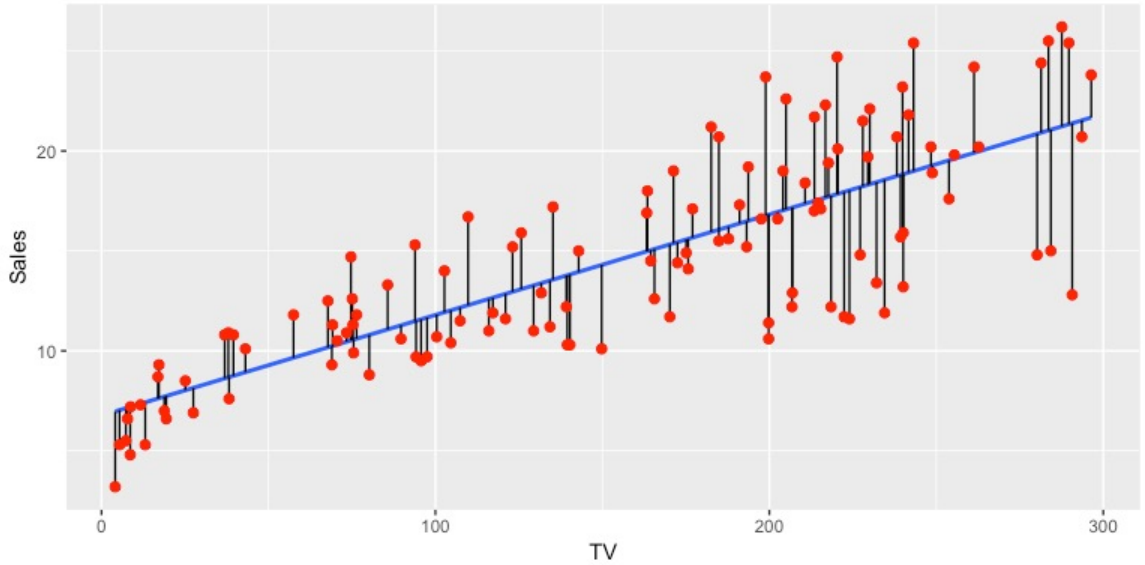
Error for Regression

- “Correct vs. Incorrect” is an odd notion
    - My linear regression here has (approximately) 0% accuracy!



Residual Sum of Squares

$$RSS(w) = ||Y - Xw||^2 = \Sigma_{i=1 \to m} (y_i - w \cdot x_i)^2$$



We're looking for the *least-squares solution*

Reporting goodness using RSS

RSS is a general notion of goodness of fit

Transformations of RSS get us to a notion like accuracy

- “I have 95% confidence that I am within $5,000 of offer price”

Reporting goodness using RSS

RSS is a general notion of goodness of fit

Transformations of RSS get us to a notion like accuracy

- “I have 95% confidence that I am within $5,000 of offer price”
- “I have 95% confidence that I am within 5,000 miles of home”

We shouldn’t need to interpret appropriate goodness for each problem

Developing goodness – a baseline

Yuma weather: sunny = True - 84 % Accuracy

- Using no features, I simply predict the majority label

Can we do a similar thing with a regression problem?

Developing goodness – a baseline

Yuma weather: sunny = True - 84 % Accuracy

- Using no features, I simply predict the majority label

Can we do a similar thing with a regression problem?

House Cost = $1

House Cost = $1,000,000

Developing goodness – a baseline

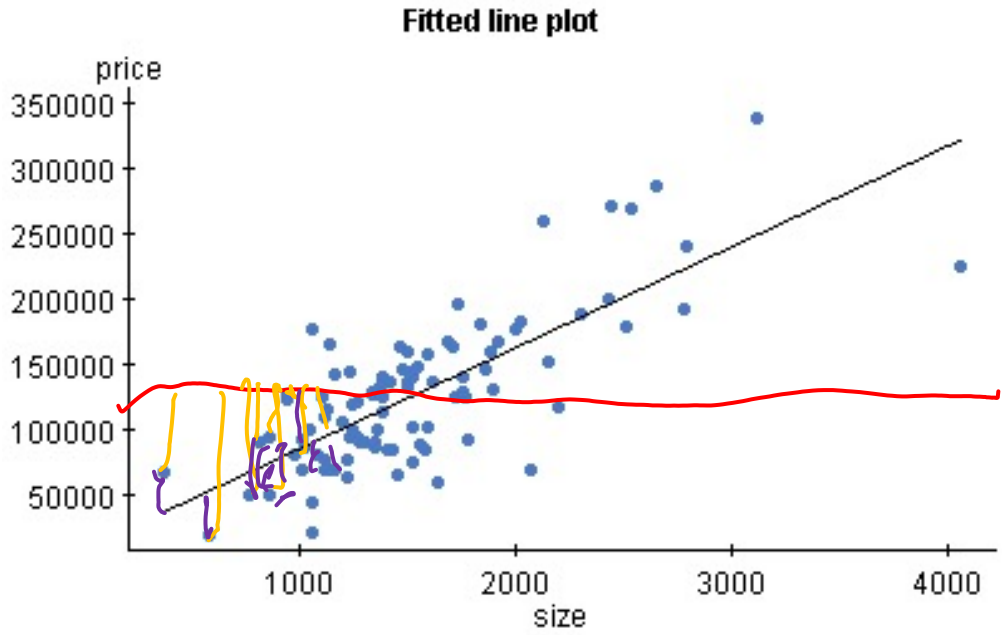
Yuma weather: sunny = True - 84 % Accuracy

- Using no features, I simply predict the majority label

Can we do a similar thing with a regression problem?

House Cost = <the average cost of a house>

Developing Goodness – a baseline



Developing goodness – alternative view

Yuma weather: sunny = True - 84 % Accuracy

- “There is no variance, assume the average state”

Can we do a similar thing with a regression problem?

House Cost = “There is no variance, assume the average state”

Developing goodness – a baseline

Baseline (Total Sum of Squares) = $||Y - avg(Y)||^2$

Our Error (Residual Sum of Squares) = $||Y - Xw||^2$

Improvement: $||Y - Xw||^2$ vs $||Y - avg(Y)||^2$

Developing goodness – coefficient of determination R^2

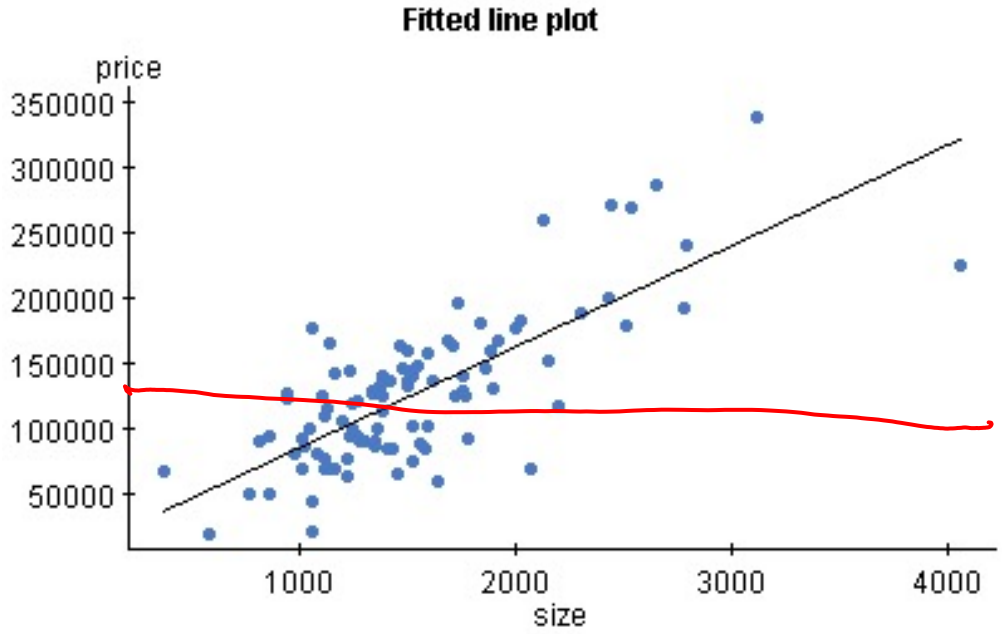
Baseline (Total Sum of Squares) = $||Y - avg(Y)||^2$

Our Error (Residual Sum of Squares) = $||Y - Xw||^2$

Improvement: $||Y - Xw||^2$ vs $||Y - avg(Y)||^2$

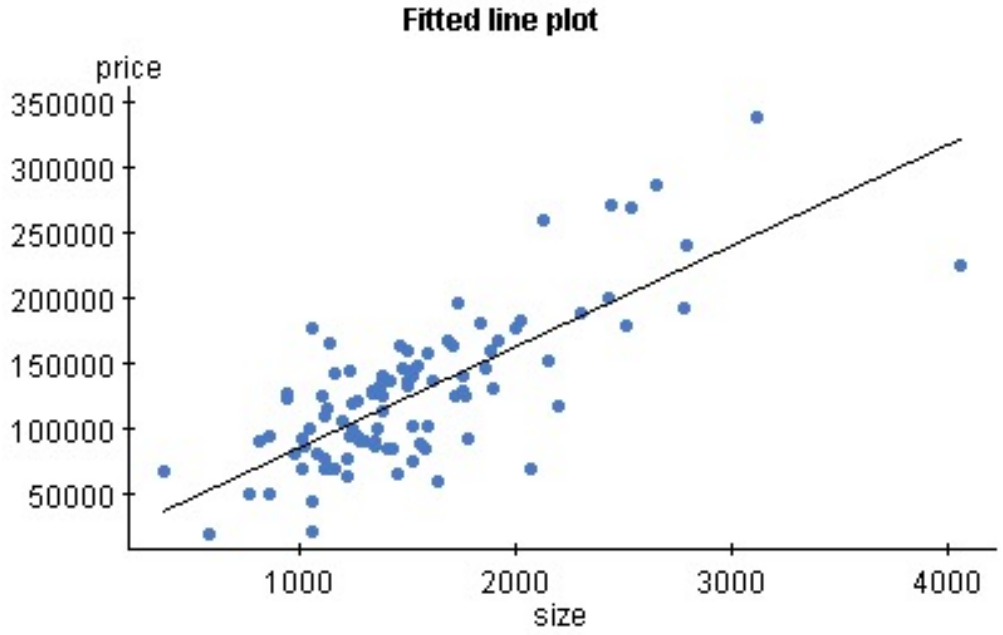
$R^2 = 1 - RSS/TSS$

Developing Goodness – R^2



Linear Regression (Improved)

Linear Regression (Week 2) – The Basics



$Y \approx \beta_0 + \beta_1 X.$

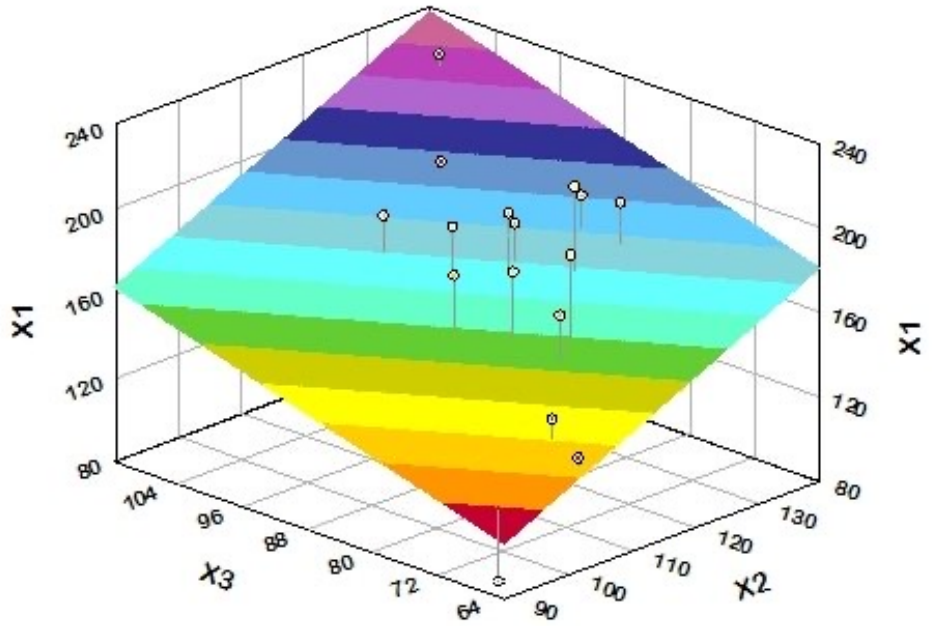
$y = w_0 + w_1 * x$

But how do we

But What happens if there's more than one feature?

Multiple Dimensions

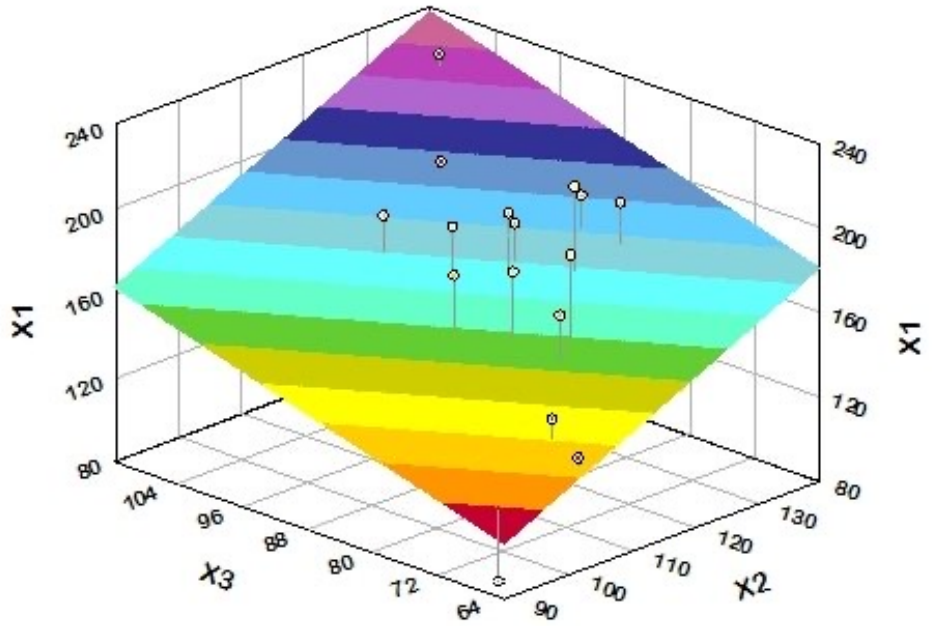
$y = w_0 + w_1 * x_1$



Multiple Dimensions

$y = w_0 + w_1 * x_1$

$y = w_0 + w_1 * x_1 + w_2 * x_2$

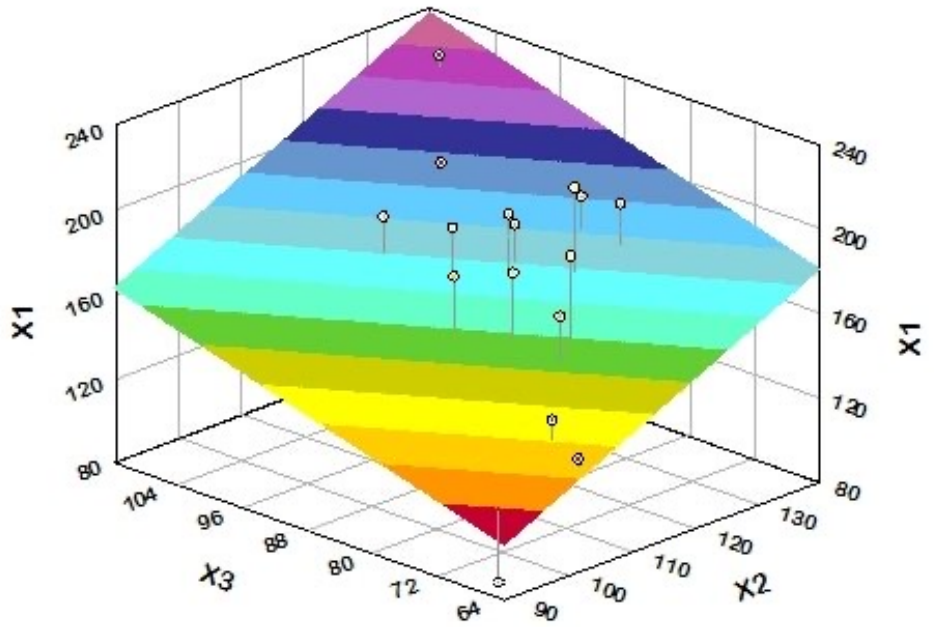


Multiple Dimensions

$y = w_0 + w_1 * x_1$

$y = w_0 + w_1 * x_1 + w_2 * x_2$

$y = w_0(*x_0) + w_1 * x_1 + w_2 * x_2 \ldots$



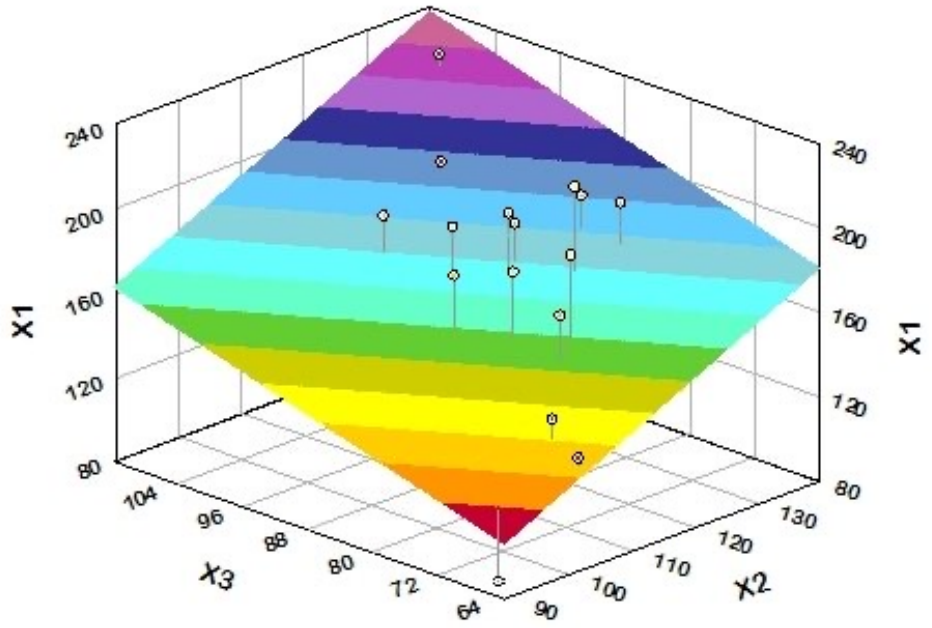
Multiple Dimensions

$y = w_0 + w_1 * x_1$

$y = w_0 + w_1 * x_1 + w_2 * x_2$

$y = w_0(*x_0) + w_1 * x_1 + w_2 * x_2 \ldots$

$y = w^T x$



Multiple Dimensions

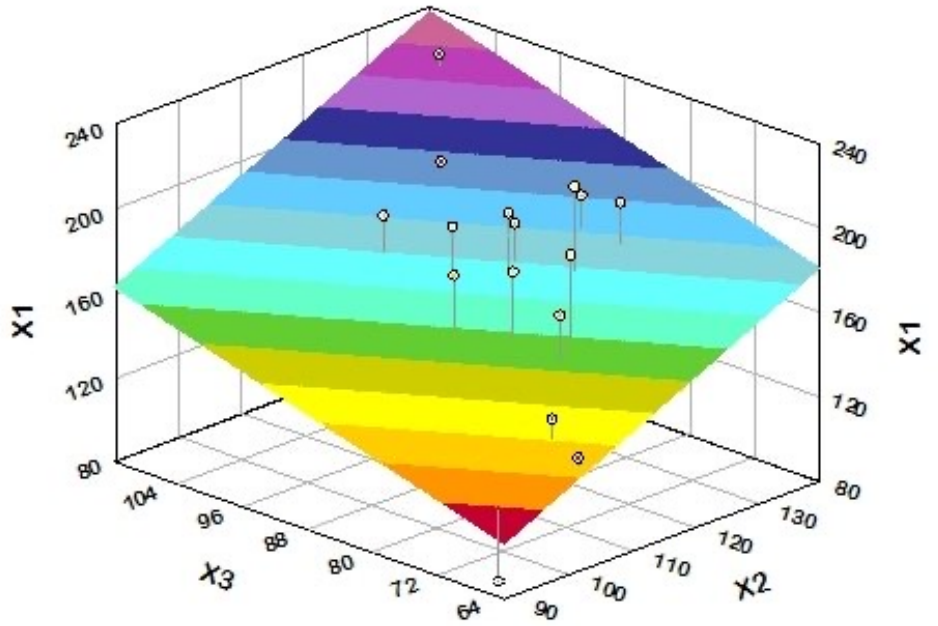
$y = w_0 + w_1 * x_1$

$y = w_0 + w_1 * x_1 + w_2 * x_2$

$y = w_0(*x_0) + w_1 * x_1 + w_2 * x_2 \ldots$

$y = w^T x$

$y = w^T x + \epsilon$



Finding our weights (w)

$y = w^T x$

For a given feature set x_i, $y_i = w^T \cdot x_i$

Combine all features into a matrix (features x examples), transpose,

$Xw = Y$

Finding our weights (w)

$Xw = Y$

But our actual scores don't map to our function...

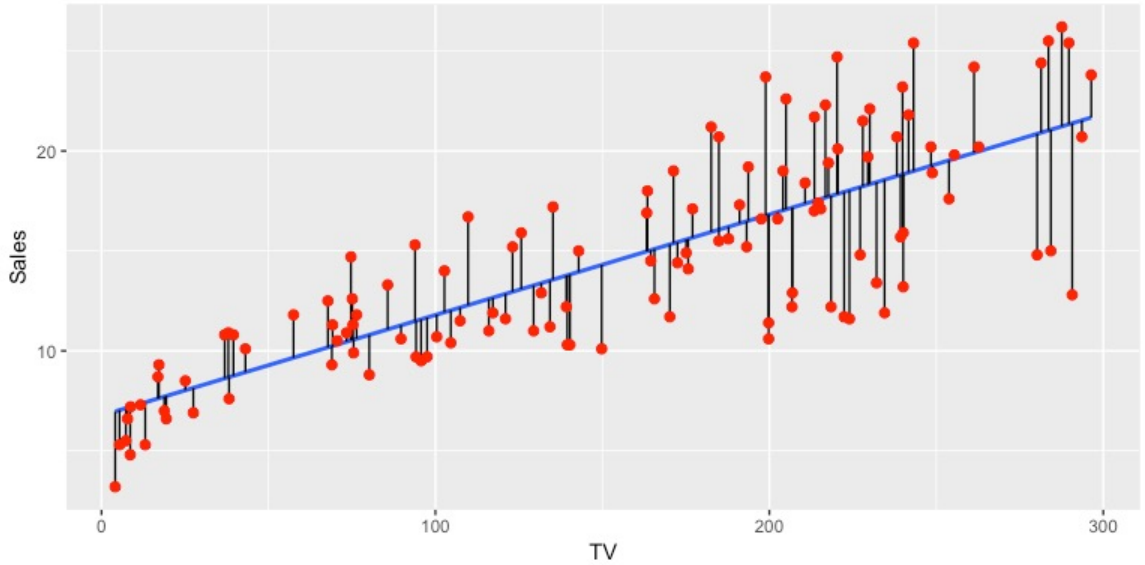
$Error = ||Y - Xw||^{\$ 2*}$

X, Y are given values of our data, so we need to find w to minimize error

$ || *a* || denotes the L2 Norm (the Euclidian distance from the origin) of *a*
*We typically look at the square of the error

Residual Sum of Squares

$$RSS(w) = ||Y - Xw||^2 = \Sigma_{i=1 \to m} (y_i - w \cdot x_i)^2$$



We're looking for the *least-squares solution*

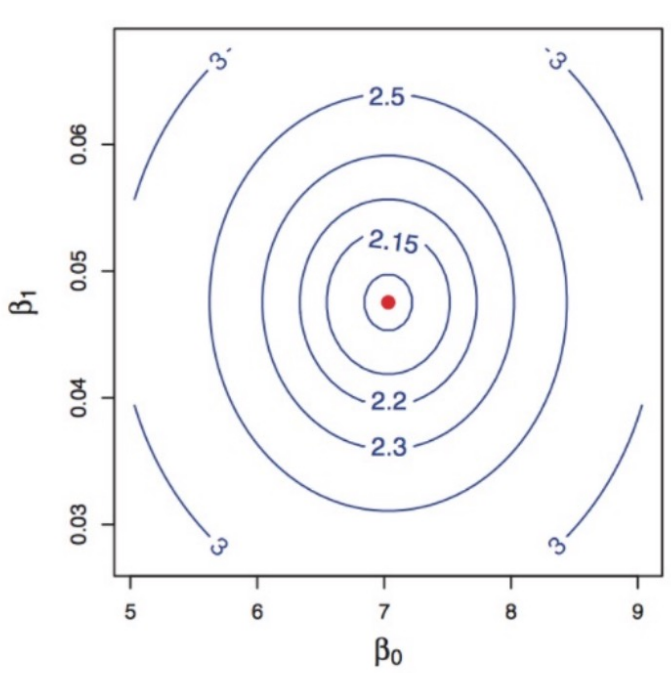
Minimizing RSS

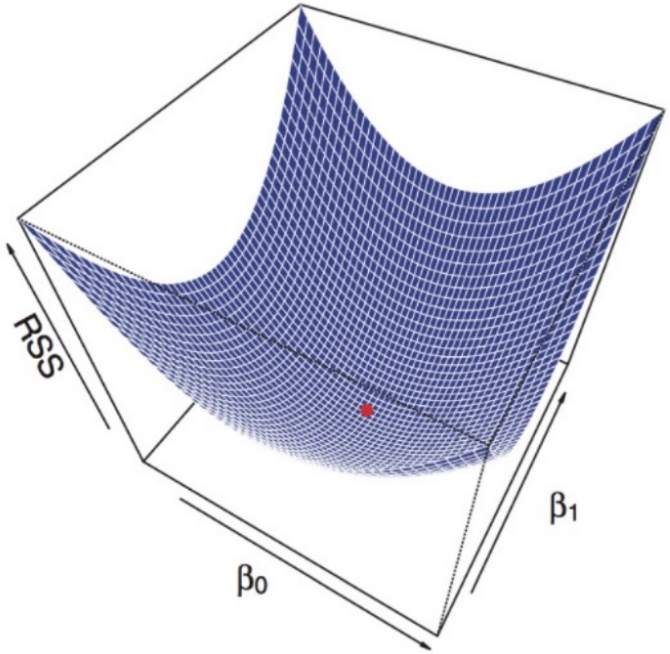
RSS is *convex*:

Unique argmin

Gradients work

$\nabla_w \text{RSS}(w) = 0$





Minimizing RSS

$$\nabla_w \text{RSS}(w) = \nabla_w ||Y - Xw||^2 = 0$$

$$X^T X w - X^t y = 0$$

$$X^T X w = X^t y \qquad \text{OR} \qquad X^T X w = (X^T X)^{-1} X^t y$$

These are our *Normal Equations*

Minimizing RSS

$X^TXw = X^ty$ OR $X^TXw = (X^TX)^{-1}X^ty$

Normal Equations are prohibitively slow...

For large data (i.e. does not fit in memory), use **Gradient Descent**

Minimizing RSS

$X^TXw = X^ty$ OR $X^TXw = (X^TX)^{-1}X^ty$

Normal Equations are prohibitively slow...

For large data (i.e. does not fit in memory), use **Gradient Descent**

Otherwise, we'll use **QR Factorization** or **Singular Value Decomposition**

QR Factorization

X is a matrix (m features x n examples)

There exist matrices Q (an n x n matrix) and R (an m x n upper triangle matrix) such that

$X = Q \cdot R$

QR Factorization

$$X = [q_1, q_2, \ldots q_n] \cdot \begin{bmatrix} R_{11} & R_{12} & \ldots & R_{1n} \\ 0 & R_{22} & \ldots & R_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & R_{nn} \end{bmatrix}$$

q_1 to q_n are *orthonormal vectors*

$||q_i|| = 1$, $q^T_i \, q_j = 0$ if $i \neq j$

R is a triangle matrix, R_{ii} is nonzero

Gram-Schmidt Process

$$X = [x_1 \mid x_2 \mid \cdots \mid x_n]$$

$$u_1 = x_1 \qquad e_1 = u_1 / \|u_1\|$$

$$u_2 = x_2 - (x_2 \cdot e_1)e_1 \qquad e_2 = u_2 / \|u_2\|$$

$$u_{k+1} = x_{k+1} - (x_{k+1} \cdot e_1)e_1 - \cdots - (x_{k+1} \cdot e_k)e_k \qquad e_{k+1} = u_{k+1} / \|u_{k+1}\|$$

$$X = [e_1, e_2, \ldots e_n] \cdot \begin{bmatrix} x_1 \cdot e_1 & x_2 \cdot e_1 & \ldots & x_n \cdot e_1 \\ 0 & x_2 \cdot e_2 & \ldots & x_n \cdot e_2 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & x_n \cdot e_n \end{bmatrix}$$

Nonlinear Mappings

Bedrooms in a house

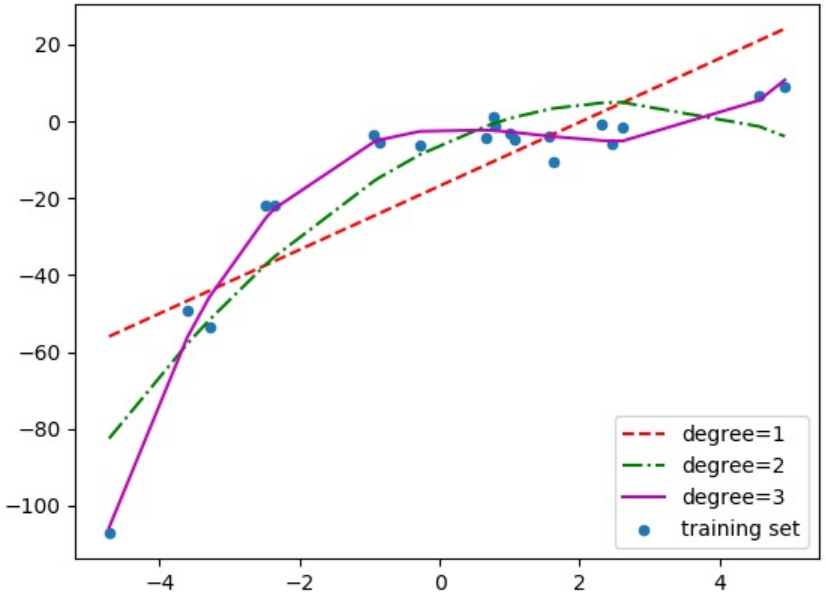
- going from 1 to 2 bedrooms increments price by X
- going from 2 to 3? 7 to 8? 20 to 21?

Beyond Linear Regression

$y = w_0(*x_0) + w_1*x_1 + w_2*x_2...$

Say x_1 is better represented as a cubic function

w_1*x_1 becomes
$w_{1,1}*x_1+w_{1,2}*x^2_1+w_{1,3}*x^3_1$ or
$w_1*x_1+w_2*x^2_1+w_3*x^3_1$



Beyond Linear Regression

$y = w_0(*x_0) + w_1*x_1 + w_2*x_2...$

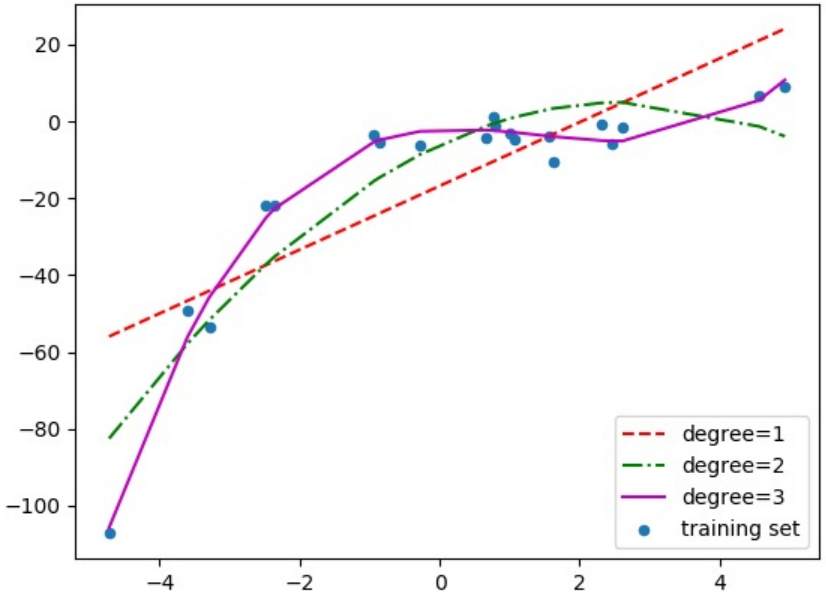
$w_1*x_1+w_2*x^2_1+w_3*x^3_1$

Create new variables

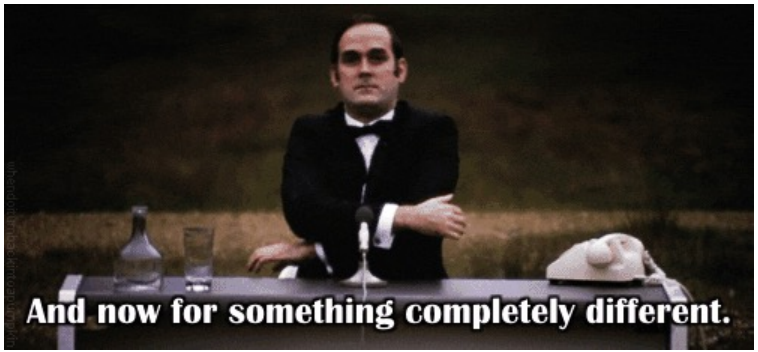
$x_2 = x^2_1$ & $x_3 = x^3_1$

(move old x_2, w_2 down to 4s)

It's linear again!



Support Vector Machines



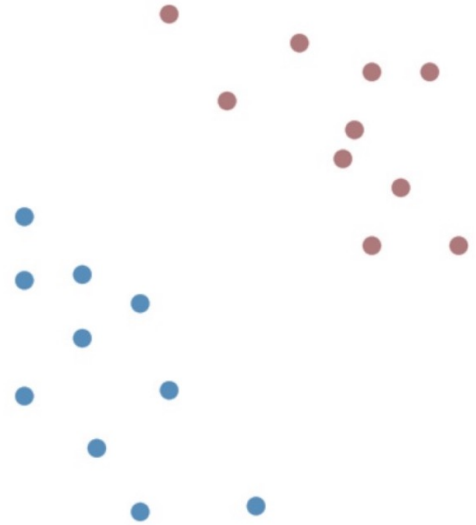
Classifying Points

Intuitively, how would we do this in...

KNN?

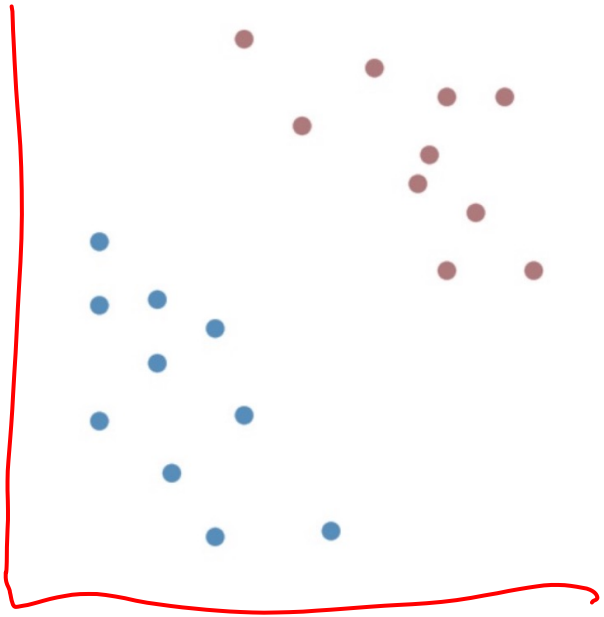
Naïve Bayes?

Logistic Regression?



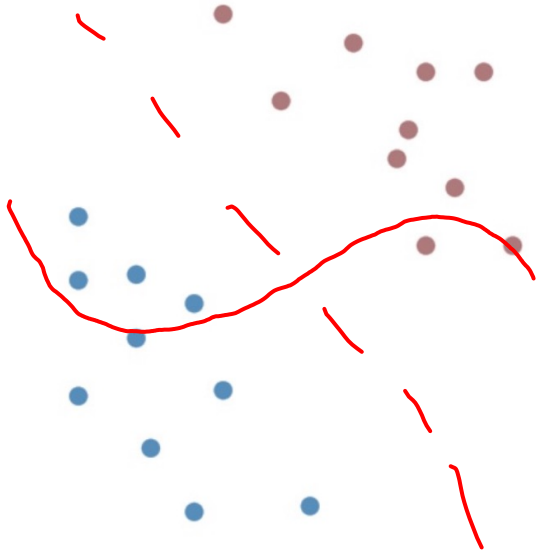
Classifying Points

- Naïve Bayes
  - Priors are equal
    - 10 points vs. 10 points
  - Class-Conditional
    - Blue is more likely if x_1, x_2 are low, inverse
  - Evidence is a wash
    - The same across Red vs. Blue



Classifying Points

- Logistic Regression
  - If the Logistic is < .5 it's one, otherwise the other
  - Weights to x_1, x_2 to find decision point
  - Decision boundary is orthogonal to regression line



Linear Classifier

A linear classifier:

- calculates a weighted sum ($w^T x$)
- classifies based on $w^T x$ >= *threshold*

Linear Classifier

A linear classifier:

- calculates a weighted sum ($w^T x + b$)
- classifies based on $w^T x + b >= 0$

Linear Classifier

A linear classifier:

- calculates a weighted sum ($w^T x + b$)
- classifies based on $w^T x + b >= 0$

New Representation

$C = \{-1, 1\}$

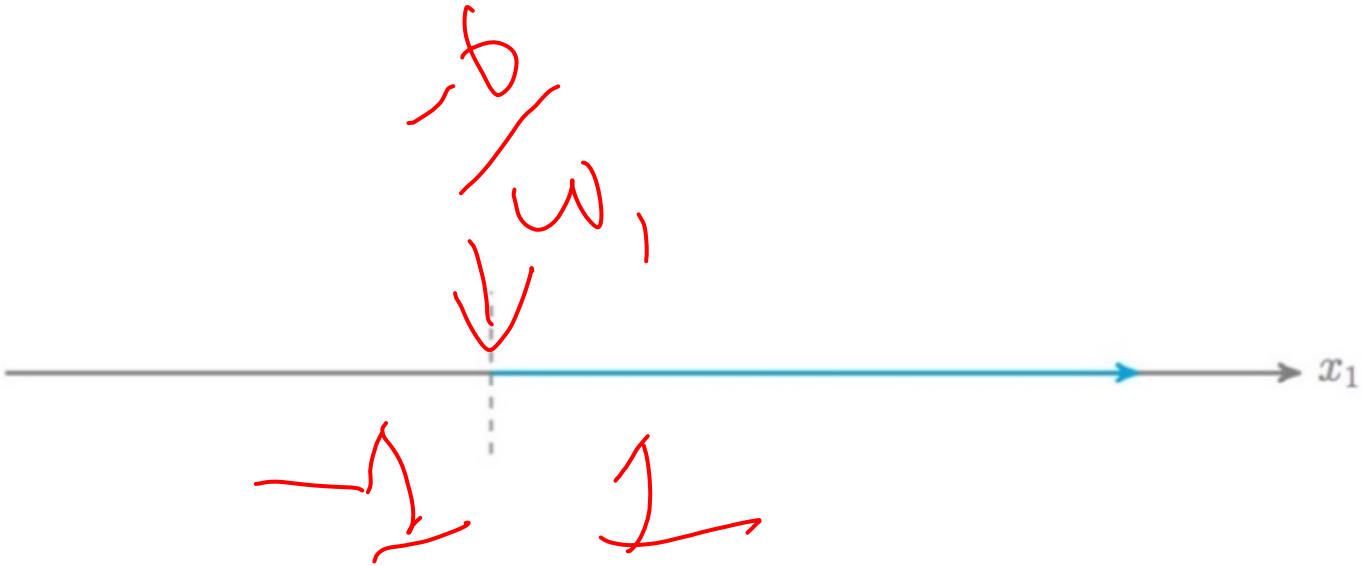
Classifying Points – 1D

$w^T x + b >= 0 \rightarrow w_1 x_1 + b$

Classifying Points – 1D

$w^T x + b >= 0 \rightarrow w_1 x_1 + b >= 0$

$x_1 >= -b / w_1$



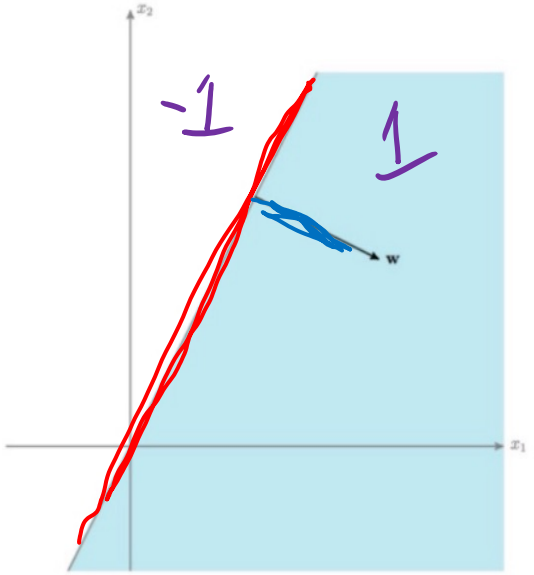
Classifying Points – 2D

$w^T x + b >= 0 \rightarrow w_1 x_1 + w_2 x_2 + b >= 0$

Classifying Points – 2D

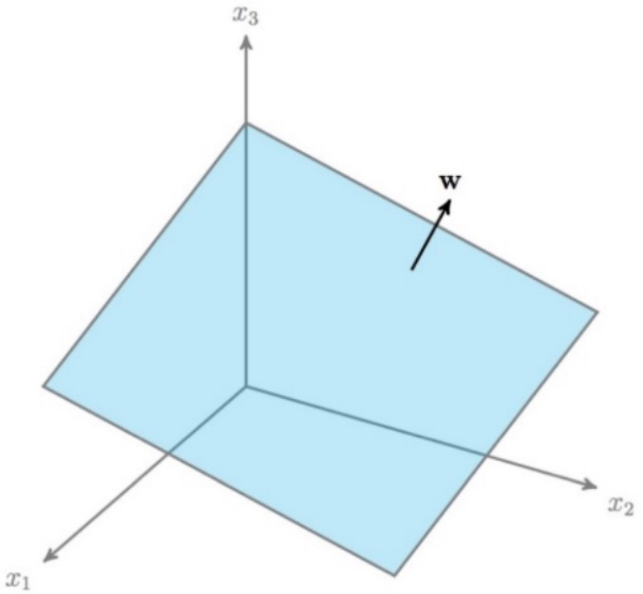
$w^T x + b >= 0 \rightarrow w_1 x_1 + w_2 x_2 + b >= 0$

$$X_2 >= \frac{-w_1 x_1 - b}{w_2}$$



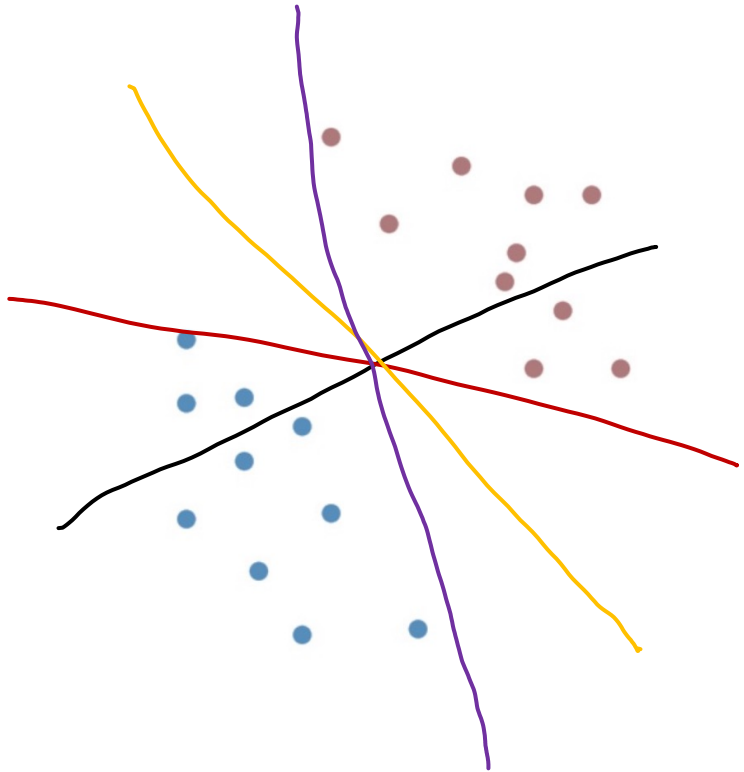
Classifying Points – 3D

Vector w is normal to the decision boundary



Decision Boundary

What is our best decision boundary?

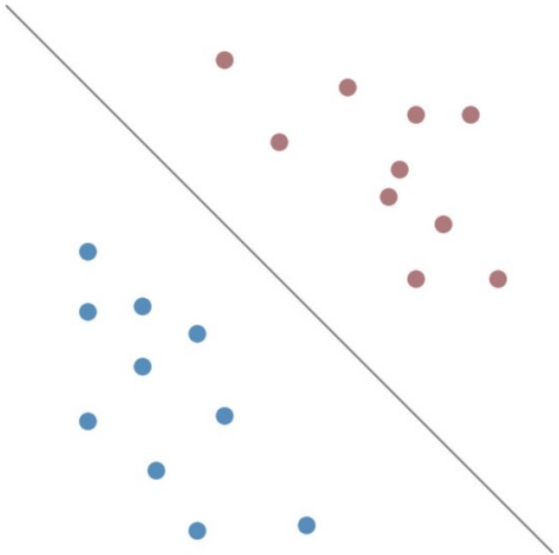


Best Decision Boundary

Why did we like this one the best?

Best gap boundary → examples

*Biggest **Margins***



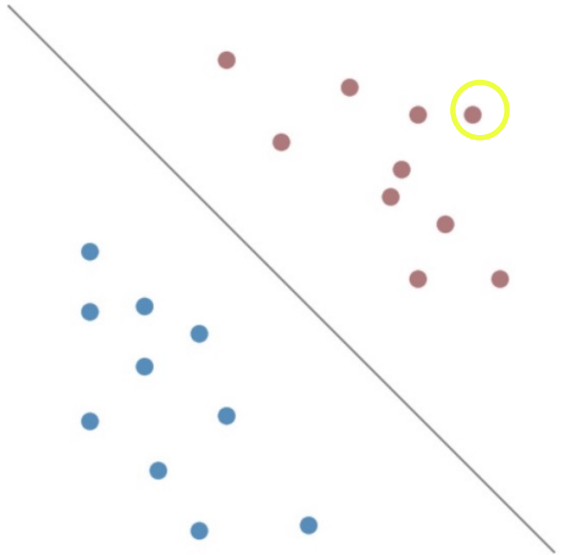
Best Decision Boundary

Why did we like this one the best?

Best gap boundary → examples

*Biggest **Margins***

$w^T x + b >> 0$, very confident $y_i = 1$



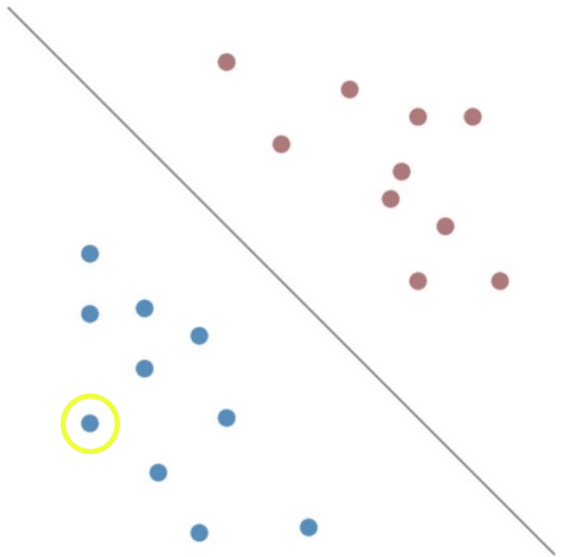
Best Decision Boundary

Why did we like this one the best?

Best gap boundary → examples

*Biggest **Margins***

$w^T x + b << 0$, very confident $y_i = -1$



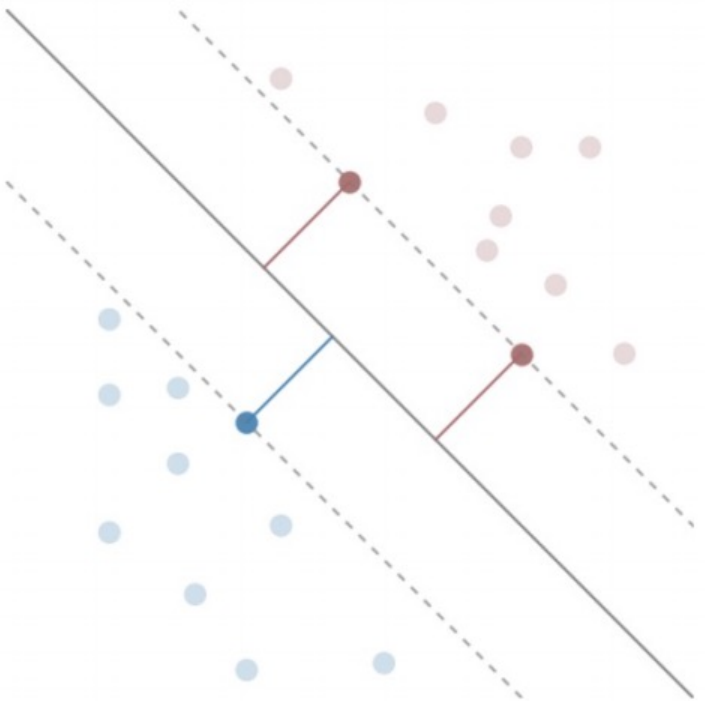
Best Decision Boundary

The ***Margin*** *(M)*

Distance DB → Closest Points

Support Vectors

closest Points to DB

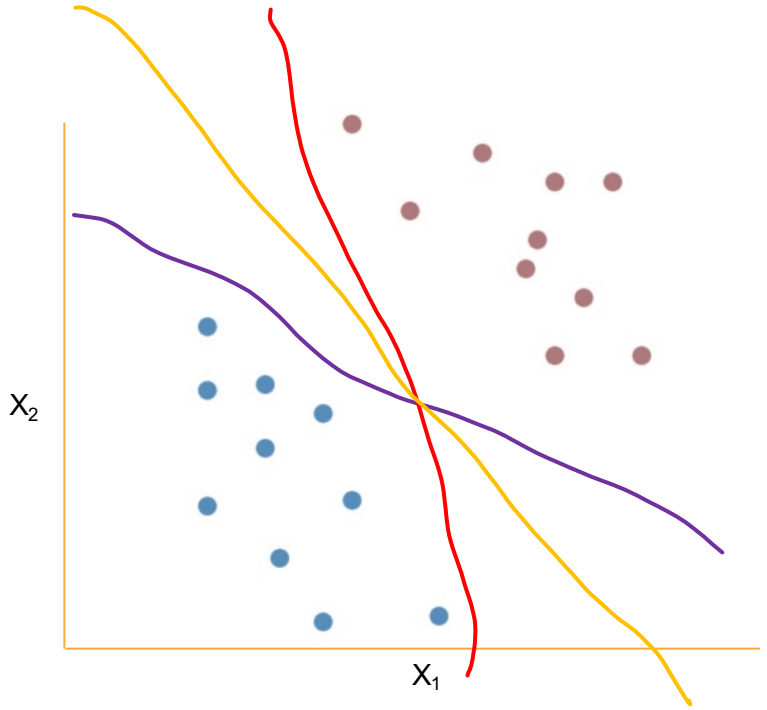


Problem Space

X (examples x features)

What is our best decision boundary?

$b + w_1x_1 + w_2x_2 + \ldots$



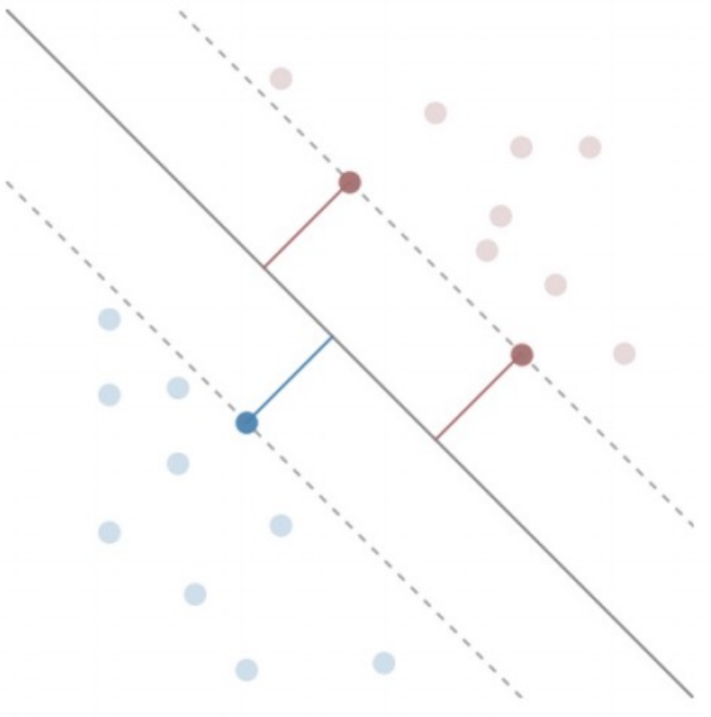
Best Decision Boundary

$\text{argmax}_{w,\,b}\,(M)$ such that points are classified correctly

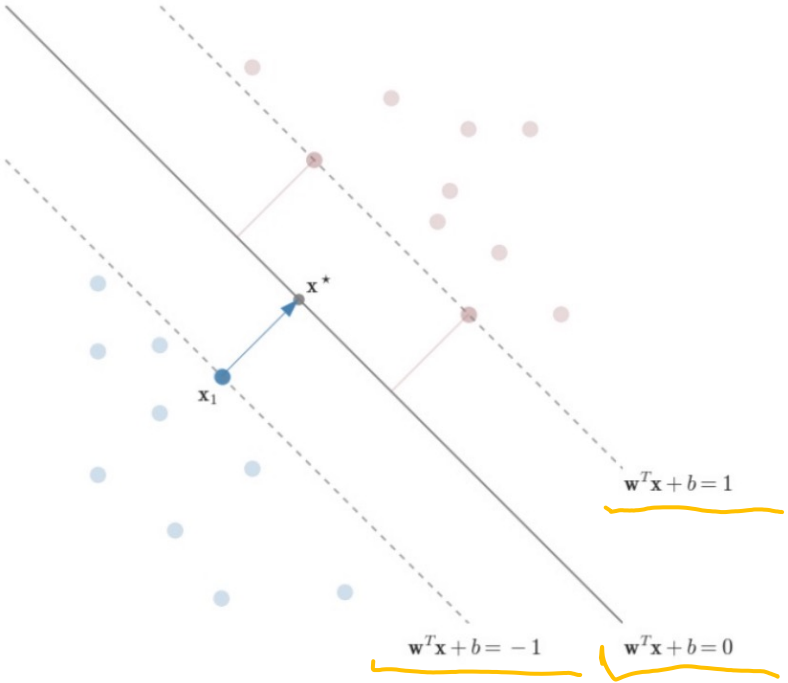
Best Decision Boundary

Support Vectors

$$w^T x + b = \pm 1$$



Maximizing our Margin

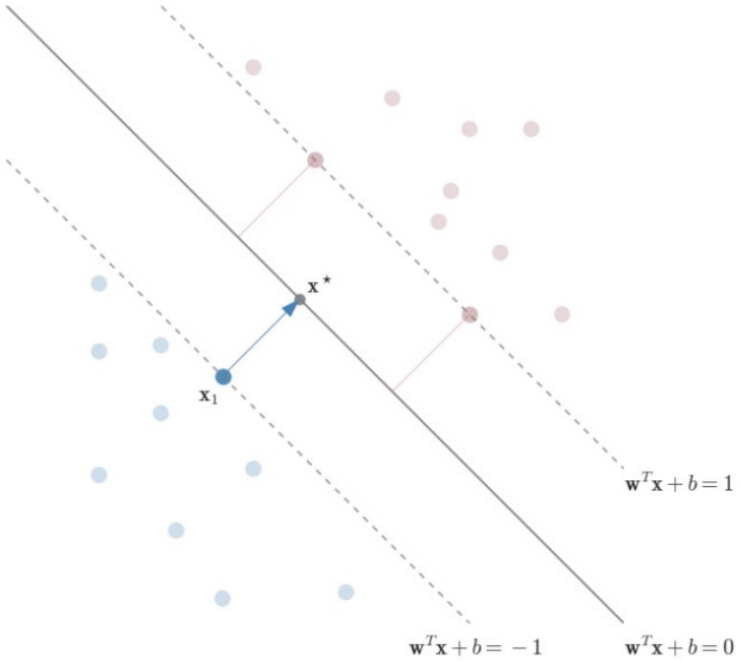


Maximizing our Margin

Choose support vector x_1
And closest point on DB x^*

move from x_1 to x^*

$x^* = x_1 + \lambda \mathbf{w}$



Maximizing our Margin

Margin is distance moved

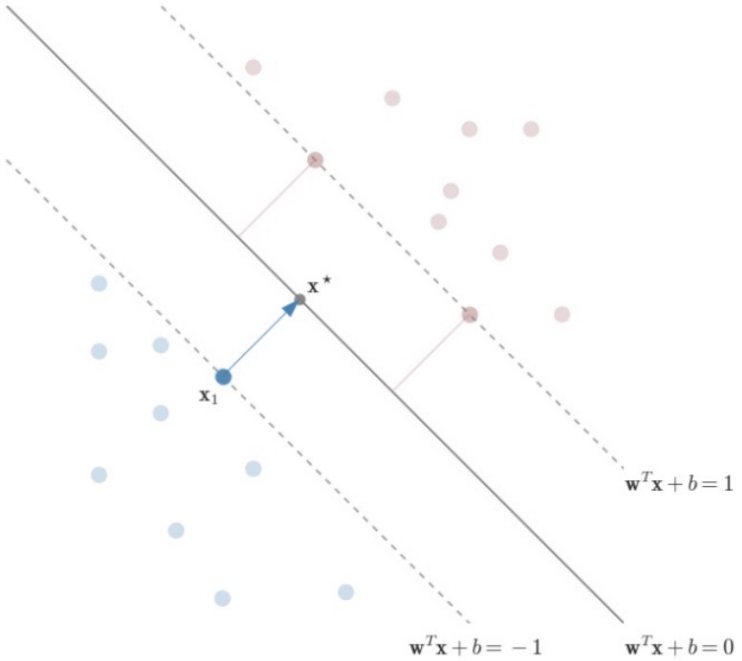
$\boldsymbol{M} = ||\lambda \mathbf{w}|| = \lambda ||\mathbf{w}||$

x^* is on the decision boundary

$$w^T x^* + b = 0$$

x_1 is on the -1 boundary

$$w^T x_1 + b = -1$$

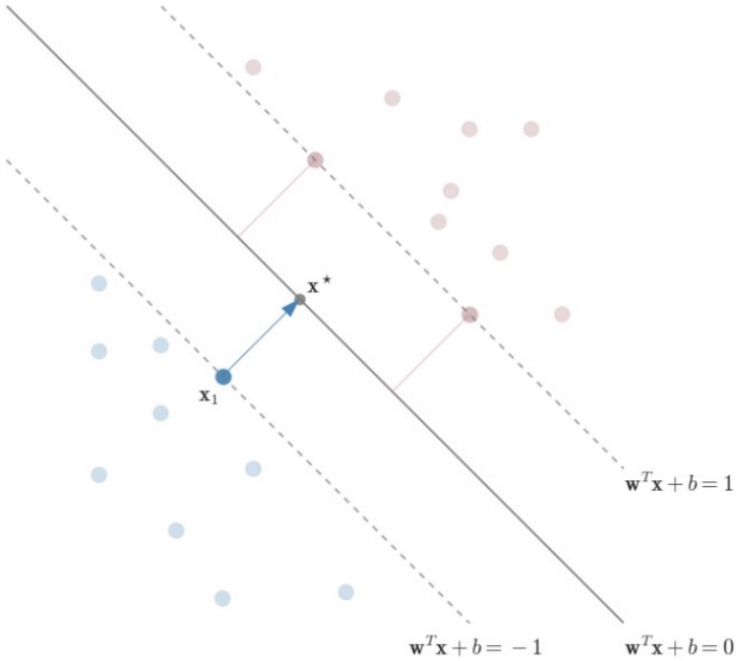


Maximizing our Margin

$x^* = x_1 + \lambda w$ (1)

$w^T x^* + b = 0$ (2)

$w^T x_1 + b = -1$ (3)



Maximizing our Margin

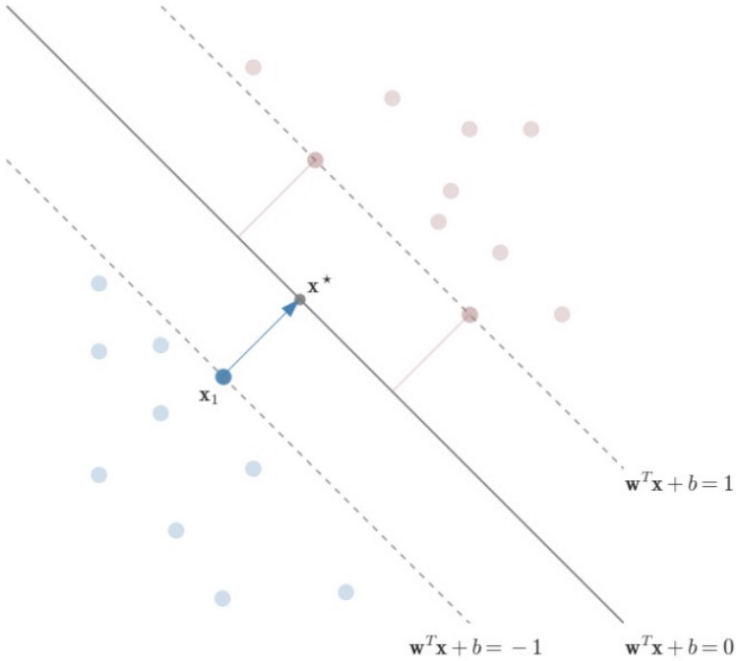
$x^* = x_1 + \lambda w$ (1)

$w^T x^* + b = 0$ (2)

$w^T x_1 + b = -1$ (3)

(1) Into (2)

$w^T (x_1 + \lambda \mathbf{w}) + b = 0$



Maximizing our Margin

$x^* = x_1 + \lambda w$ (1)

$w^T x^* + b = 0$ (2)

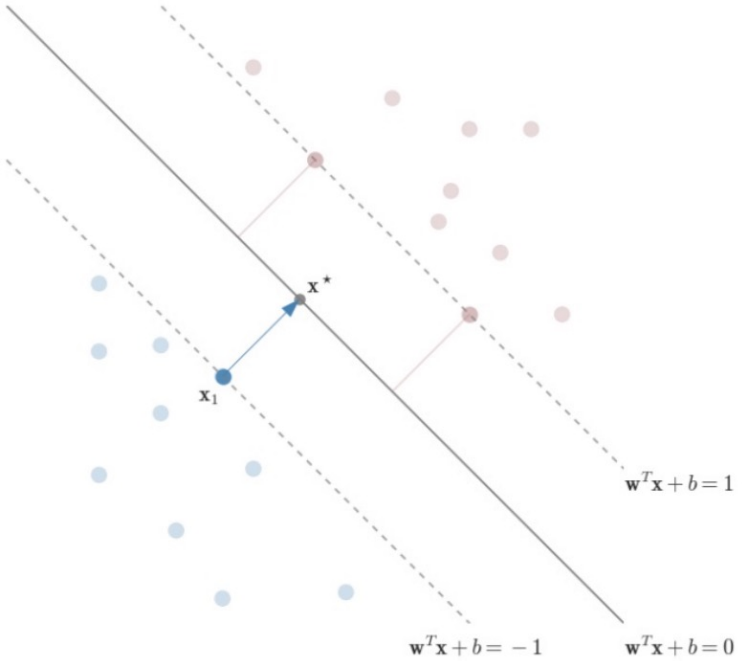
$w^T x_1 + b = -1$ (3)

(1) Into (2)

$w^T (x_1 + \lambda \mathbf{w}) + b = 0 \rightarrow$

$w^T x_1 + \lambda\, w^T \mathbf{w} + b = 0 \rightarrow$

$w^T x_1 + b + \lambda\, w^T \mathbf{w} = 0$ (4)



Maximizing our Margin

$x^* = x_1 + \lambda w$ (1)

$w^T x^* + b = 0$ (2)

$w^T x_1 + b = -1$ (3)

(1) Into (2)

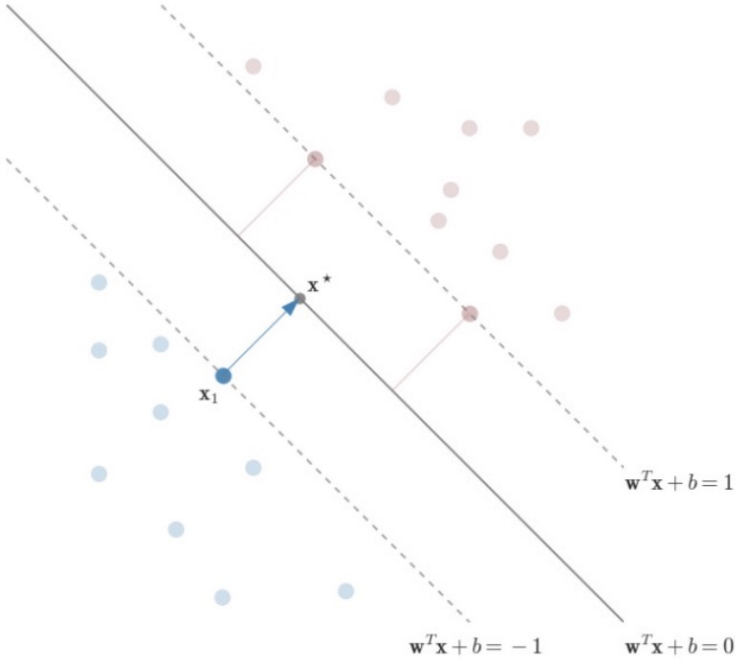
$w^T (x_1 + \lambda \mathbf{w}) + b = 0 \rightarrow$

$w^T x_1 + \lambda\, w^T \mathbf{w} + b = 0 \rightarrow$

$w^T x_1 + b + \lambda\, w^T \mathbf{w} = 0$ (4)

(3) Into (4)

$-1 + \lambda\, w^T \mathbf{w} = 0 \rightarrow \lambda\, w^T \mathbf{w} = 1$ (5)



Maximizing our Margin

Norm $\mathbf{w}$ ($||w||$) is the magnitude of $\mathbf{w}$

$$||\mathbf{w}|| = \sqrt{(w_1^2 + w_2^2 + w_3^2 + \ldots + w_n^2)}$$

Dot Product of a vector with itself (transposed) is the norm squared

$$||\mathbf{w}||^2 = (w_1^2 + w_2^2 + w_3^2 + \ldots + w_n^2) = \mathbf{w}^T\mathbf{w} \quad (6)$$

Maximizing our Margin

$x^* = x_1 + \lambda w$ (1)

$w^T x^* + b = 0$ (2)

$w^T x_1 + b = -1$ (3)

(1) Into (2)

$w^T (x_1 + \lambda \mathbf{w}) + b = 0 \rightarrow$

$w^T x_1 + \lambda\, w^T \mathbf{w} + b = 0 \rightarrow$

$w^T x_1 + b + \lambda\, w^T \mathbf{w} = 0$ (4)

(3) Into (4)

$-1 + \lambda\, w^T \mathbf{w} = 0 \rightarrow \lambda\, w^T \mathbf{w} = 1$ (5)

$\|\mathbf{w}\|^2 = \mathbf{w}^T \mathbf{w}$ (6)

(6) Into (5)

$\lambda\, \|\mathbf{w}\|^2 = 1$

Maximizing our Margin

$x^* = x_1 + \lambda w$ (1)

$w^T x^* + b = 0$ (2)

$w^T x_1 + b = -1$ (3)

(1) Into (2)

$w^T (x_1 + \lambda \mathbf{w}) + b = 0 \rightarrow$

$w^T x_1 + \lambda\, w^T \mathbf{w} + b = 0 \rightarrow$

$w^T x_1 + b + \lambda\, w^T \mathbf{w} = 0$ (4)

(3) Into (4)

$-1 + \lambda\, w^T \mathbf{w} = 0 \rightarrow \lambda\, w^T \mathbf{w} = 1$ (5)

$||\mathbf{w}||^2 = \mathbf{w}^T \mathbf{w}$ (6)

(6) Into (5)

$\lambda\, ||\mathbf{w}||^2 = 1$ (7)

$\mathbf{M} = ||\lambda \mathbf{w}|| = \lambda ||\mathbf{w}||$ (8)

(8) Into (7)

$\mathbf{M = 1 / ||w||}$

Best Decision Boundary

$\text{argmax}_{w,\,b}\,(M)$ such that points are classified correctly

Best Decision Boundary

$\text{argmax}_{w,\,b}\,(M)$ such that points are classified correctly

$\text{argmax}_{w,\,b}\,(1 / ||w||)$ such that points are classified correctly

Best Decision Boundary

$\text{argmax}_{w, b}$ (M) such that points are classified correctly

$\text{argmax}_{w, b}$ $(1 / ||w||)$ such that points are classified correctly

$w^T x_i + b <= -1$ if $y_i = -1$

$w^T x_i + b >= 1$ if $y_i = 1$

Best Decision Boundary

$\text{argmax}_{w,\,b}\,(M)$ such that points are classified correctly

$\text{argmax}_{w,\,b}\,(1 / ||w||)$ such that points are classified correctly

$w^T x_i + b <= -1$ if $y_i = -1$

$w^T x_i + b >= 1$ if $y_i = 1$

$y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Best Decision Boundary

$\text{argmax}_{w,\,b}\,(1 / ||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Optimize to find w, b

Best Decision Boundary

$\text{argmax}_{w,\,b}\,(1 / ||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Optimize to find w, b

But it's not differentiable! (i.e. we can't find a gradient vector)

Best Decision Boundary

$\text{argmax}_{w,\,b}\ (1 / ||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Not Differentiable

$\text{argmin}_{w,\,b}\ (||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Best Decision Boundary

$\text{argmax}_{w,\,b}\ (1 / ||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Not Differentiable

$\text{argmin}_{w,\,b}\ (||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Not Differentiable

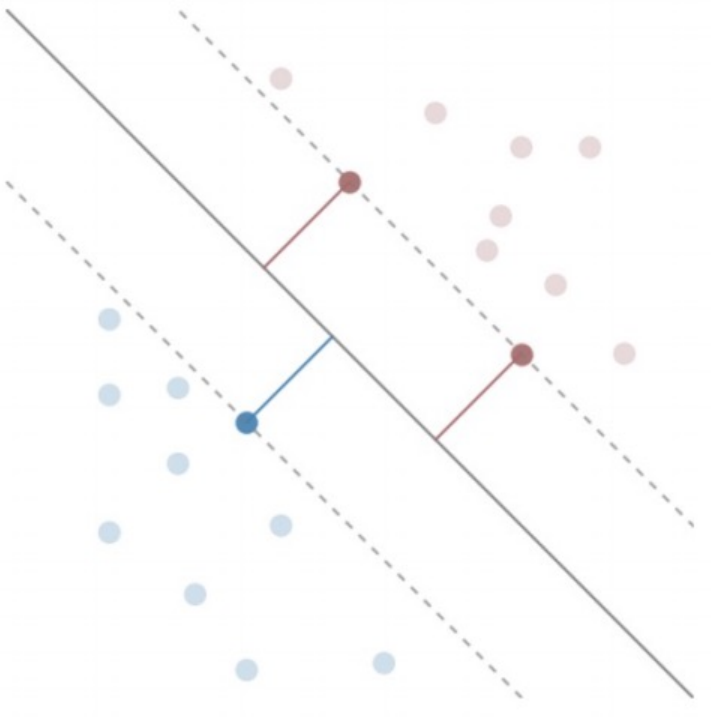
Best Decision Boundary

Support Vectors

$$w^T x + b = \pm 1$$

DB is unaffected by scale

$$(2w)^T x + 2b = 0$$



Best Decision Boundary

END

$\text{argmax}_{w,\,b}\ (1 / ||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Not Differentiable

$\text{argmin}_{w,\,b}\ (||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Not Differentiable

$\text{argmin}_{w,\,b}\ (||w||^2)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Convex + Differentiable

Requires a Convex quadratic program