Machine Learning and Data Mining

Lecture 5.2: Midterm Review

CS 434

© Stefan Lee

Midterm Logistics

- **Midterm is in class and on paper.**
  - Will be open note. No limits on amount of material.
    - Recommend 2-3 pages or else you'll spend too much time searching.
    - Act of making sheet may help you focus your study.
  - Raise your hand if you have a question during the exam.

© Stefan Lee

Midterm Structure

Points	Section
_____ / 2	Bonus Questions On This Page
_____ / 40	Multiple Choice Concept Questions
_____ / 17	Support Vector Machines
_____ / 17	kNN and LOO Cross-Validation
_____ / 17	Fitting Binary Naïve Bayes Models
_____ / 9	Logistic Regression and Regularization
_____ / 100	Total

© Stefan Lee

RECAP

From ~~Last~~ Lectures

All (?)

© Stefan Lee

General Concepts

© Stefan Lee

No Assumptions ➔ No Learning

It's clear we are interested in mappings from input to output. Let's get concrete.

Millow (legally distinct from Zillow)

Goal: Predict whether a house will sell within a month

Your records contain d attributes for each house…

- Lets assume binary: HasPool? >300sqft?, IsHaunted?...
- Implies $x \in \{0,1\}^d$

… and whether it sold within a month.

- Implies $y \in \{0,1\}$

How many possible mappings from x to y are there?

© Stefan Lee

No Assumptions ➔ No Learning

How many possible mappings $g: \{0,1\}^d \rightarrow \{0,1\}$ from x to y are there?

What is a mapping? For each unique x, provide an output y.

x	y
0 0 … 0	0
0 0 … 1	1
.	.
.	.
.	.
1 1 … 1	0

x	y
0 0 … 0	1
0 0 … 1	1
.	.
.	.
.	.
1 1 … 1	0

x	y
0 0 … 0	1
0 0 … 1	0
.	.
.	.
.	.
1 1 … 1	0

…

Making no assumptions about the function -- 2^{2^d} possible!

© Stefan Lee

Will data save us?

Each unique data point fills in one of these rows for us. Say we have n.

x	y
0 0 … 0	1
0 0 … 1	
.	
.	
.	
1 1 … 1	0

Still $2^{2^d - n}$ possible mappings.
Far, far too many.

We need to make some sort of assumption about the relationship between input and output.

© Stefan Lee

Question Break!

What assumption is made for k-nearest neighbors?

A That the relationship between input and output can be derived from a linear function.

B The label changes smoothly as the features change in local regions.

C That each feature affects the output independently.

D kNN makes no assumptions.

© Stefan Lee

Question Break!

What assumption is made for logistic regression?

A That the relationship between input and output can be derived from a linear function.

B The label changes smoothly as the features change in local regions.

C That each feature affects the output independently.

D

© Stefan Lee

Question Break!

What other algorithms share a similar assumption to logistic regression?

A Naïve Bayes

B Support Vector Machines

C Perceptron

D Linear Regression

© Stefan Lee

Error Decomposition for Function Approximation

Find function $g: X \rightarrow Y$ from hypothesis space $\mathcal{H}$ that best approximates $f: X \rightarrow Y$ given a **dataset** $\mathcal{D}$ of observations of f

© Stefan Lee

Error Decomposition for Function Approximation

Find function $g: X \rightarrow Y$ from hypothesis space $\mathcal{H}$ that best approximates $f: X \rightarrow Y$ given a **dataset** $\mathcal{D}$ of observations of f

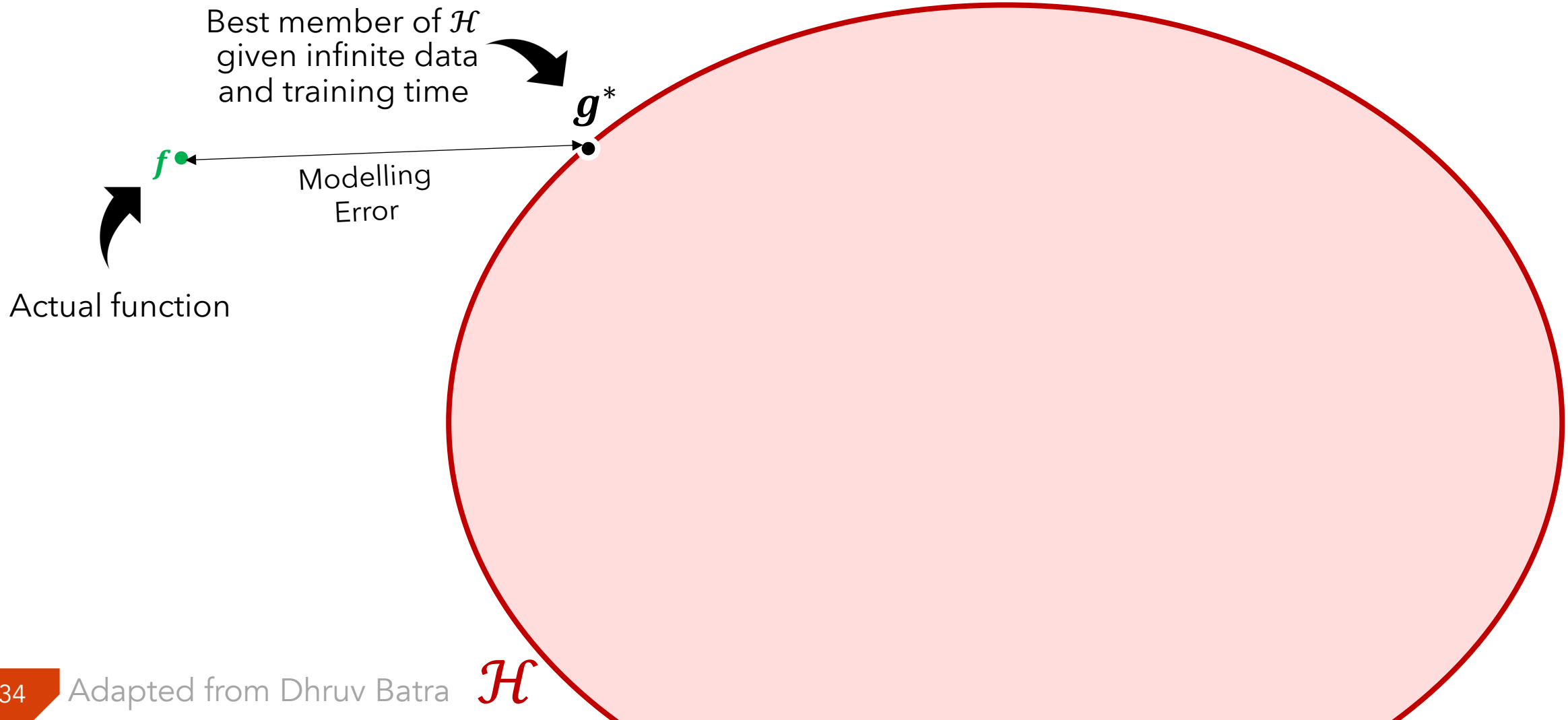
f

Actual function

$\mathcal{H}$

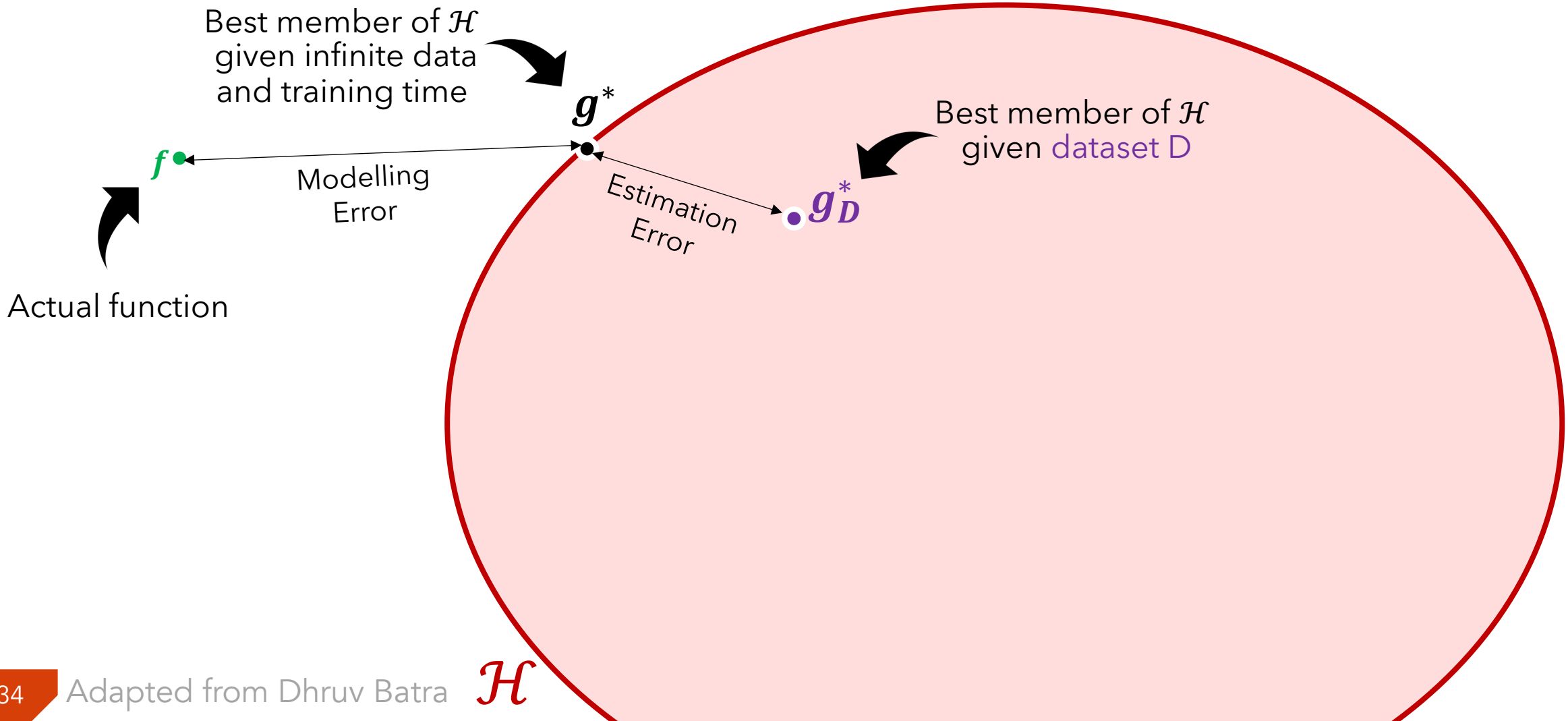
Error Decomposition for Function Approximation

Find function $g: X \rightarrow Y$ from hypothesis space $\mathcal{H}$ that best approximates $f: X \rightarrow Y$ given a **dataset** $\mathcal{D}$ of observations of f



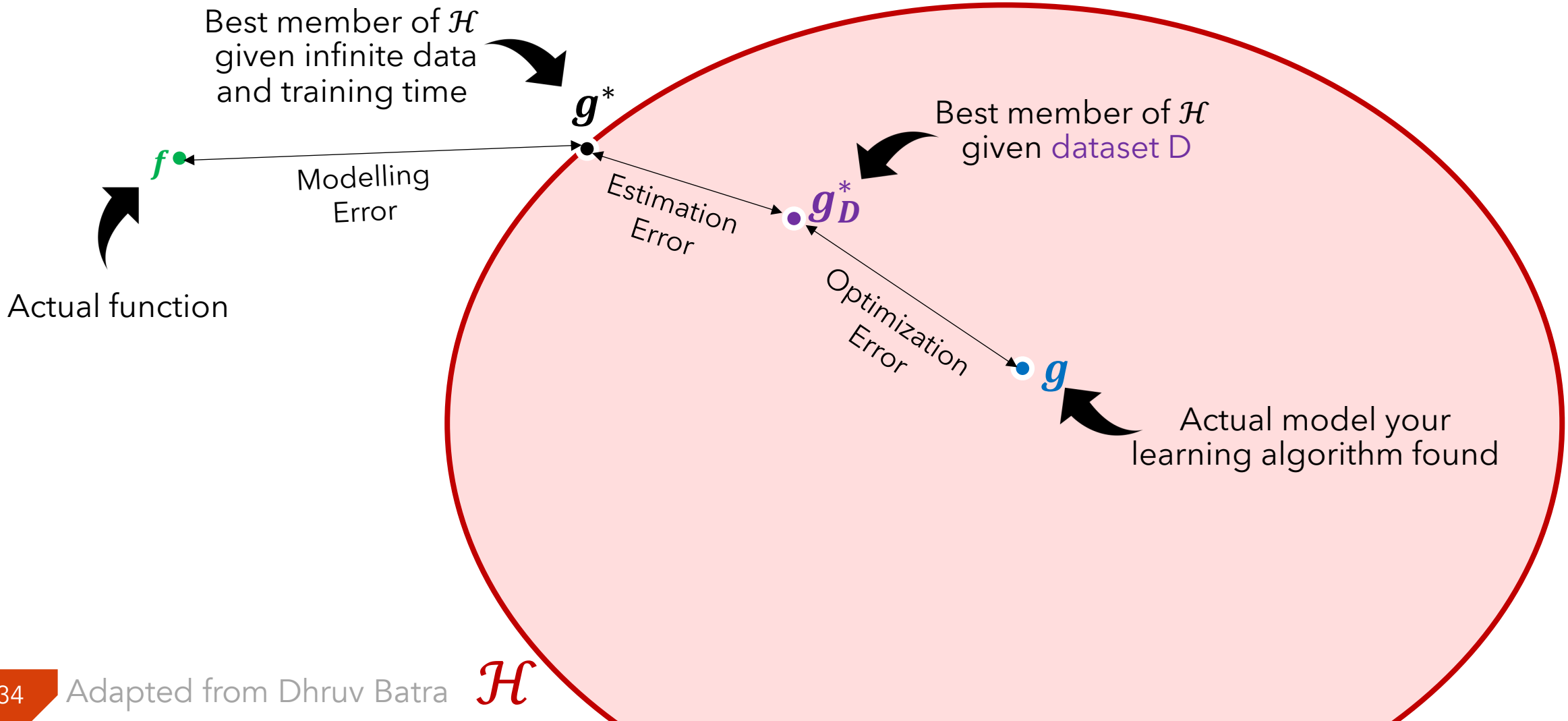
Error Decomposition for Function Approximation

Find function $g: X \rightarrow Y$ from hypothesis space $\mathcal{H}$ that best approximates $f: X \rightarrow Y$ given a **dataset** $\mathcal{D}$ of observations of f



Error Decomposition for Function Approximation

Find function $g: X \rightarrow Y$ from hypothesis space $\mathcal{H}$ that best approximates $f: X \rightarrow Y$ given a **dataset** $\mathcal{D}$ of observations of f



Question Break!

Which source(s) of error tend(s) towards zero with infinite data?

A Optimization Error

B Estimation Error

C Modelling Error

D Bayes Error

© Stefan Lee

Question Break!

Which source(s) of error tend(s) towards zero as computation goes to infinity?

A Optimization Error

B Estimation Error

C Modelling Error

D Bayes Error

© Stefan Lee

Question Break!

Consider a polynomial regression model of order m.

Suppose I train this model with m=2 and it performs poorly on both training and test data, but it performs well with m=5 . The initial model's poor performance is an example of _____.

A Optimization Error

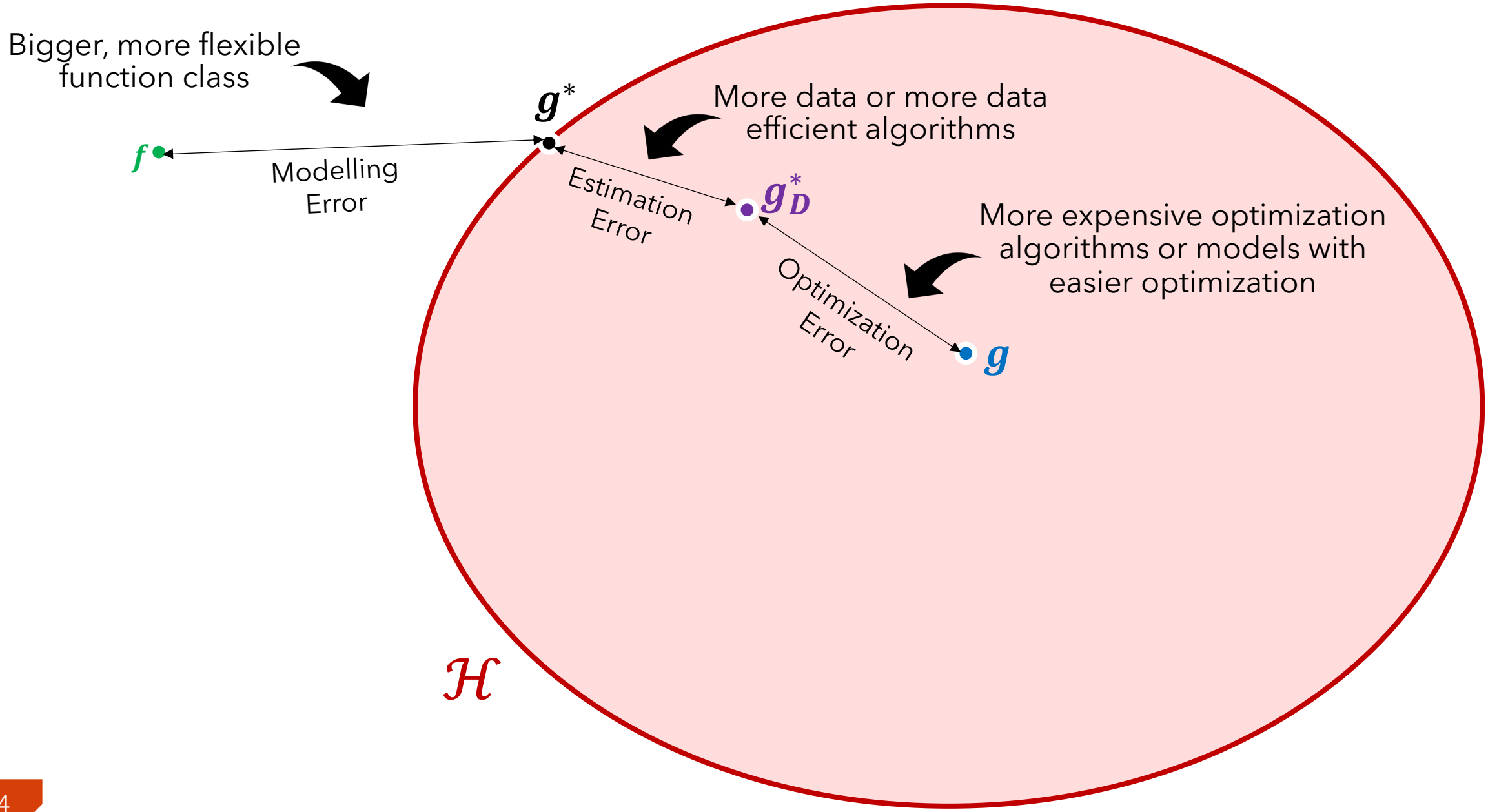
B Estimation Error

C Modelling Error

D Bayes Error

© Stefan Lee

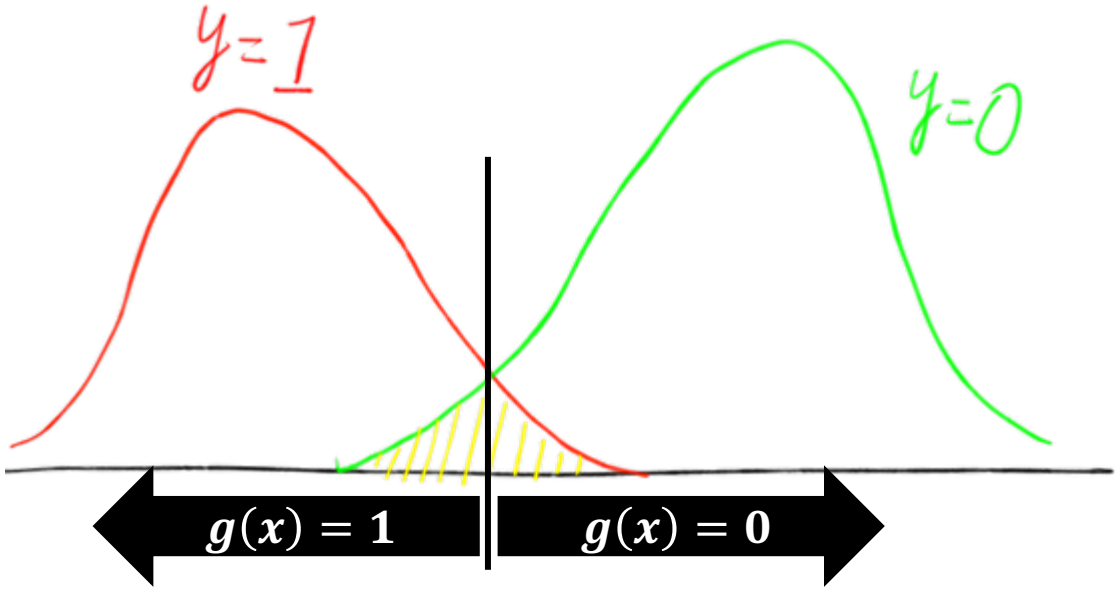
Reducing Sources of Error



© Stefan Lee

What is Bayes Error?

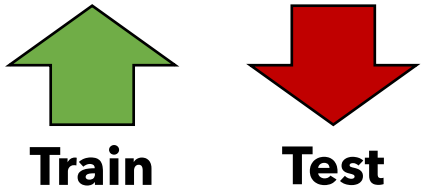
Bayes Error
Irreducible error inherit in the function being approximated – nothing we can fix.

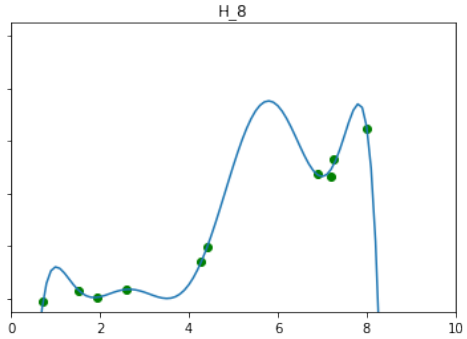


© Stefan Lee

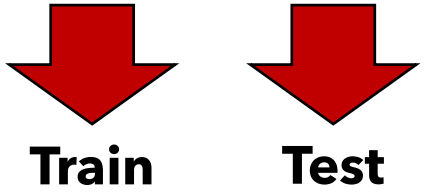
Overfitting, Underfitting, and Model Selection

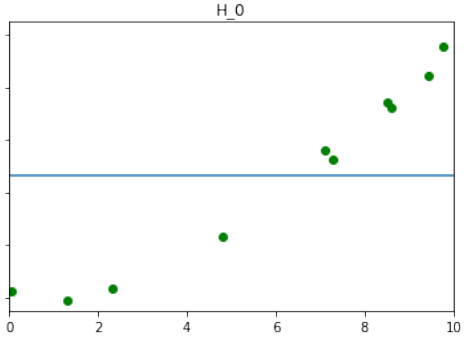
Overfitting: Model performs well on training data, but poorly on held-out testing data.





Underfitting: Model performs poorly on training data. Likely also performs poorly on test data.





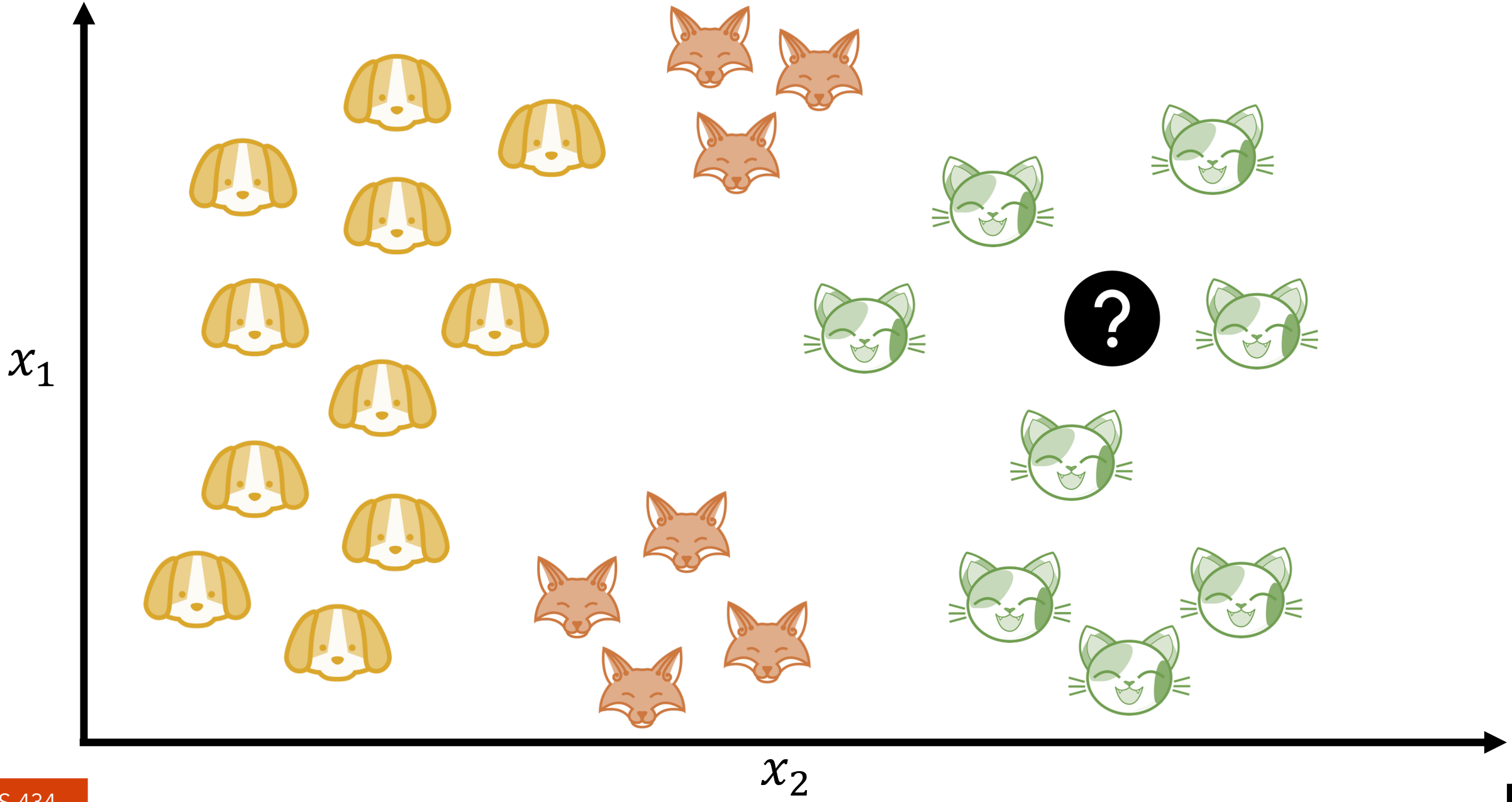
Model Selection: The exercise of finding a hypothesis space / model class that neither underfits or overfits. ← One of the reasons doing ML well is hard.

© Stefan Lee

kNN

© Stefan Lee

Intuition for your First Machine Learning Algorithm



© Stefan Lee

Formalizing k-Nearest Neighbors

Classification Rule: Given a dataset of $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, let $f_k(\boldsymbol{x})$ be the prediction of a k-nearest neighbor classifier on a point $\boldsymbol{x}$ such that:

k-nearest neighbor: $$f_k(\boldsymbol{x}) = \operatorname*{argmax}_{y \in Y} v_y$$

Predict whichever label most neighbors have

$$v_y = \sum_{i \in N_k(\boldsymbol{x})} \mathbb{I}[y_i = y]$$

Tally votes for each label from each neighbor

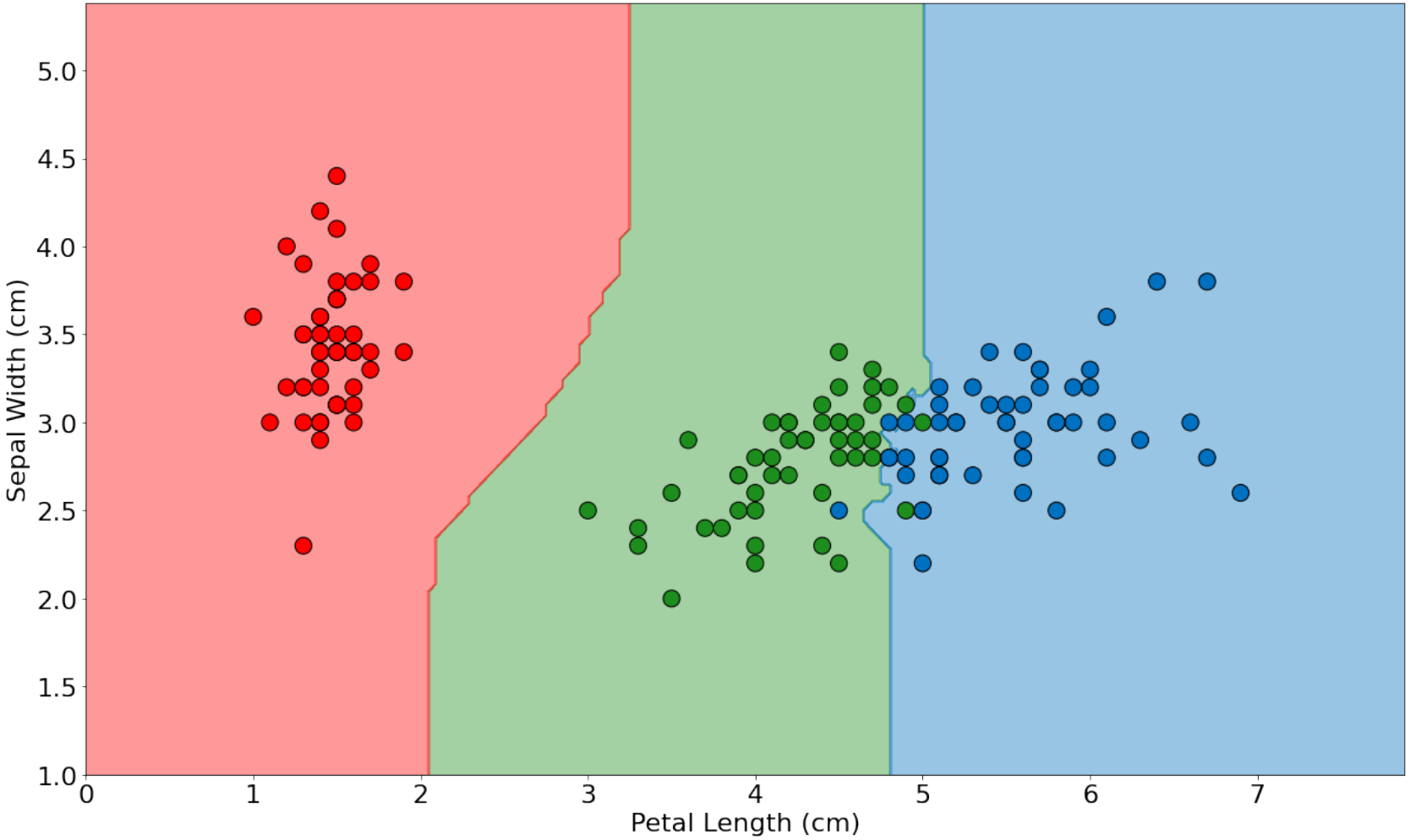
Note: The **indicator function** $\mathbb{I}[condition]$ is 1 if *condition* is true, and 0 otherwise.

© Stefan Lee

An Example: Iris Species Classification

Decisions made from a 5-nearest neighbor algorithm

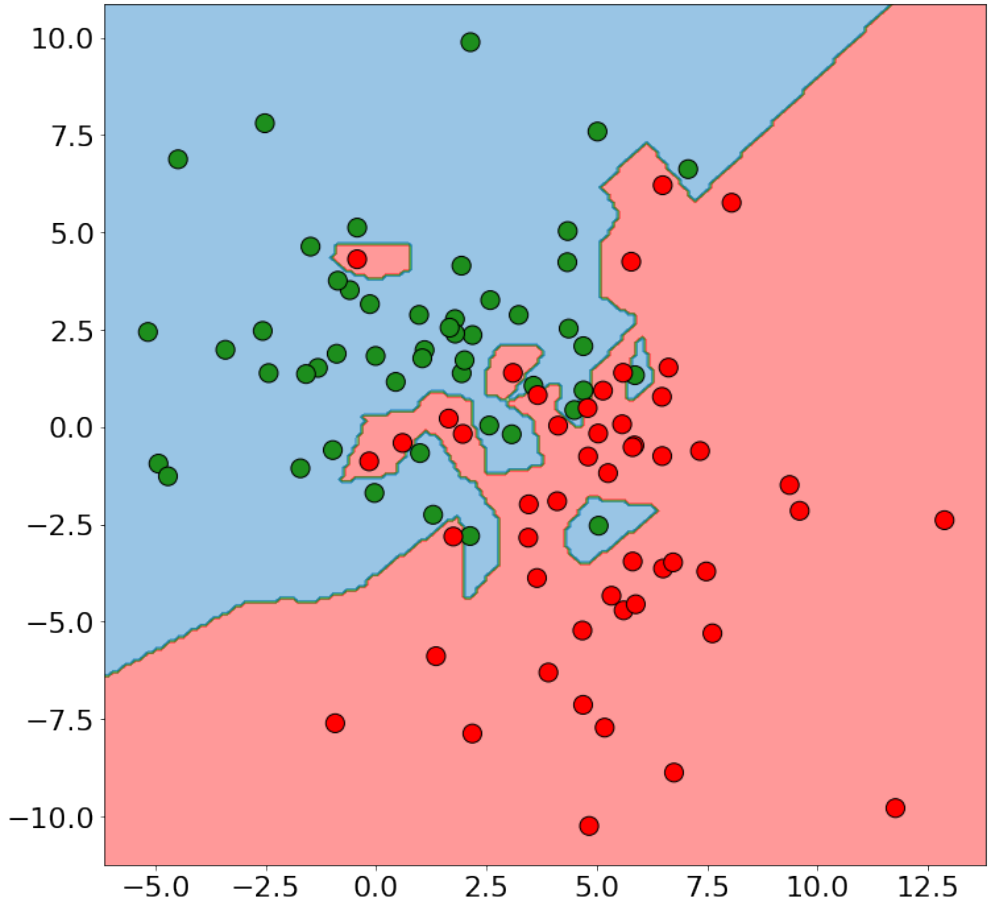
setosa, versicolor, and virginica



© Stefan Lee

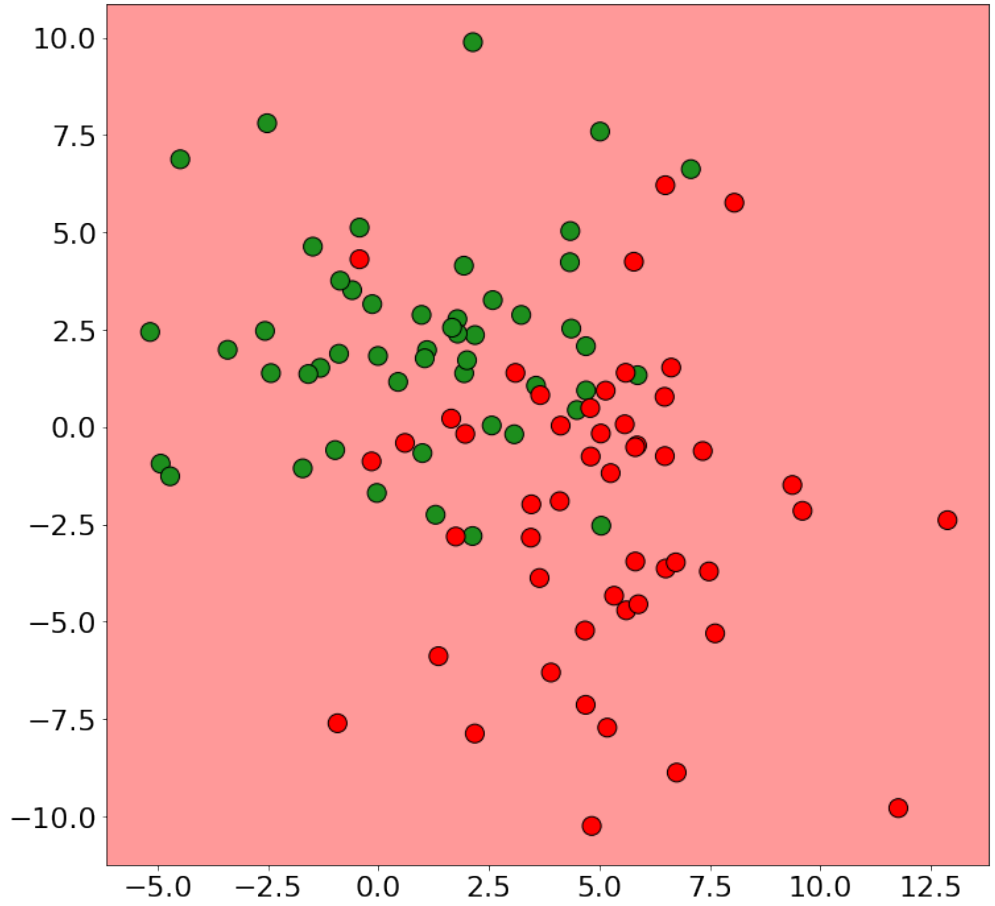
The Effect of Extreme k-values

k=1



At k=1, training error tends towards zero because each point is its own nearest neighbor!

k=n



At k=n, every point is a neighbor so the majority class in the dataset is predicted everywhere.

© Stefan Lee

Formalizing k-Nearest Neighbors for Regression

Regression Rule: Given a dataset of $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, let $f_k(\boldsymbol{x})$ be the prediction of a k-nearest neighbor regressor on a point $\boldsymbol{x}$ such that:

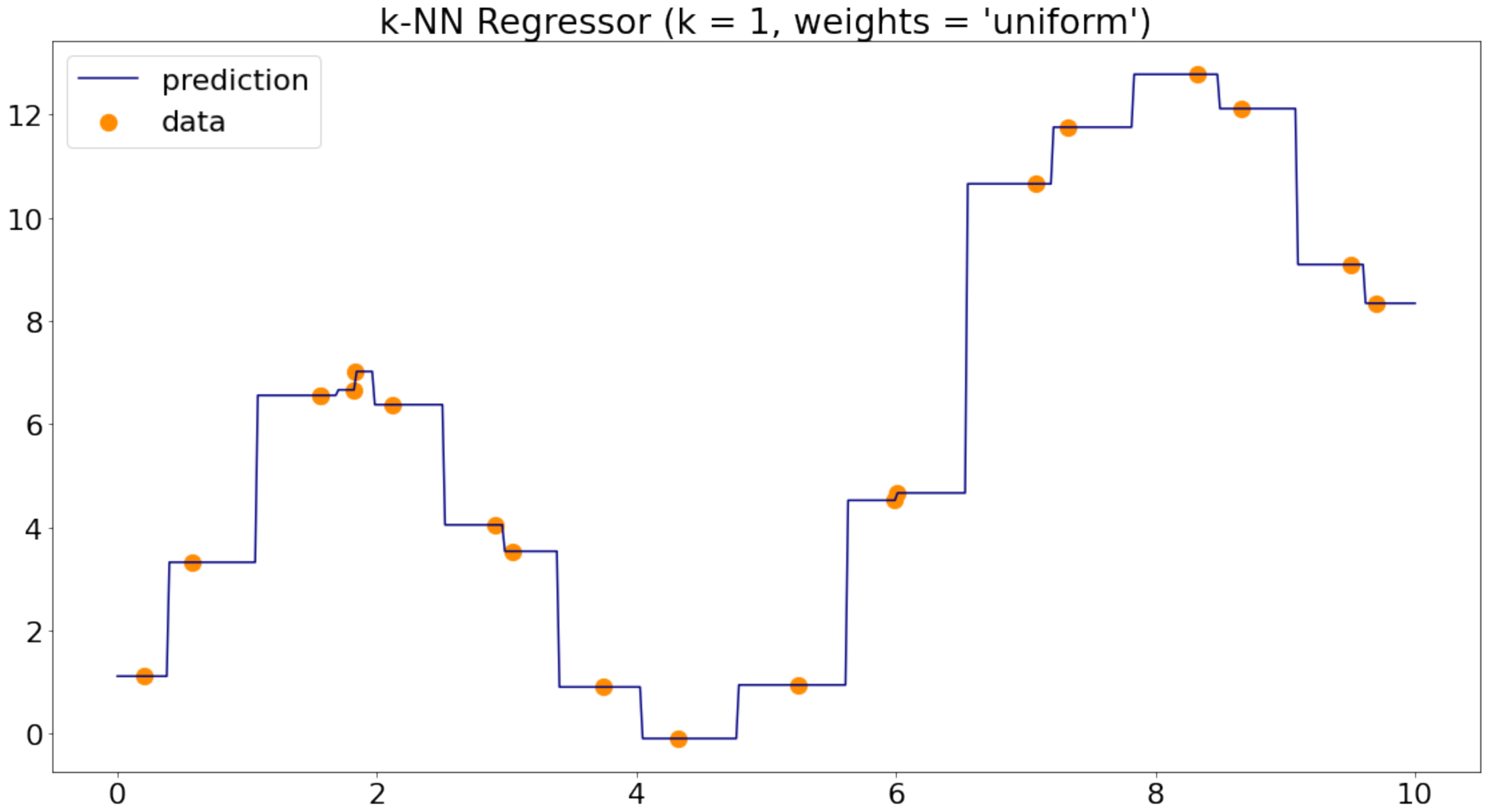
1-nearest neighbor: $$f_1(\boldsymbol{x}) = y_{N_1(\boldsymbol{x})}$$

k-nearest neighbor: $$f_k(\boldsymbol{x}) = \frac{1}{k}\sum_{i \in N_k(\boldsymbol{x})} y_i$$

Predict average value of neighbors

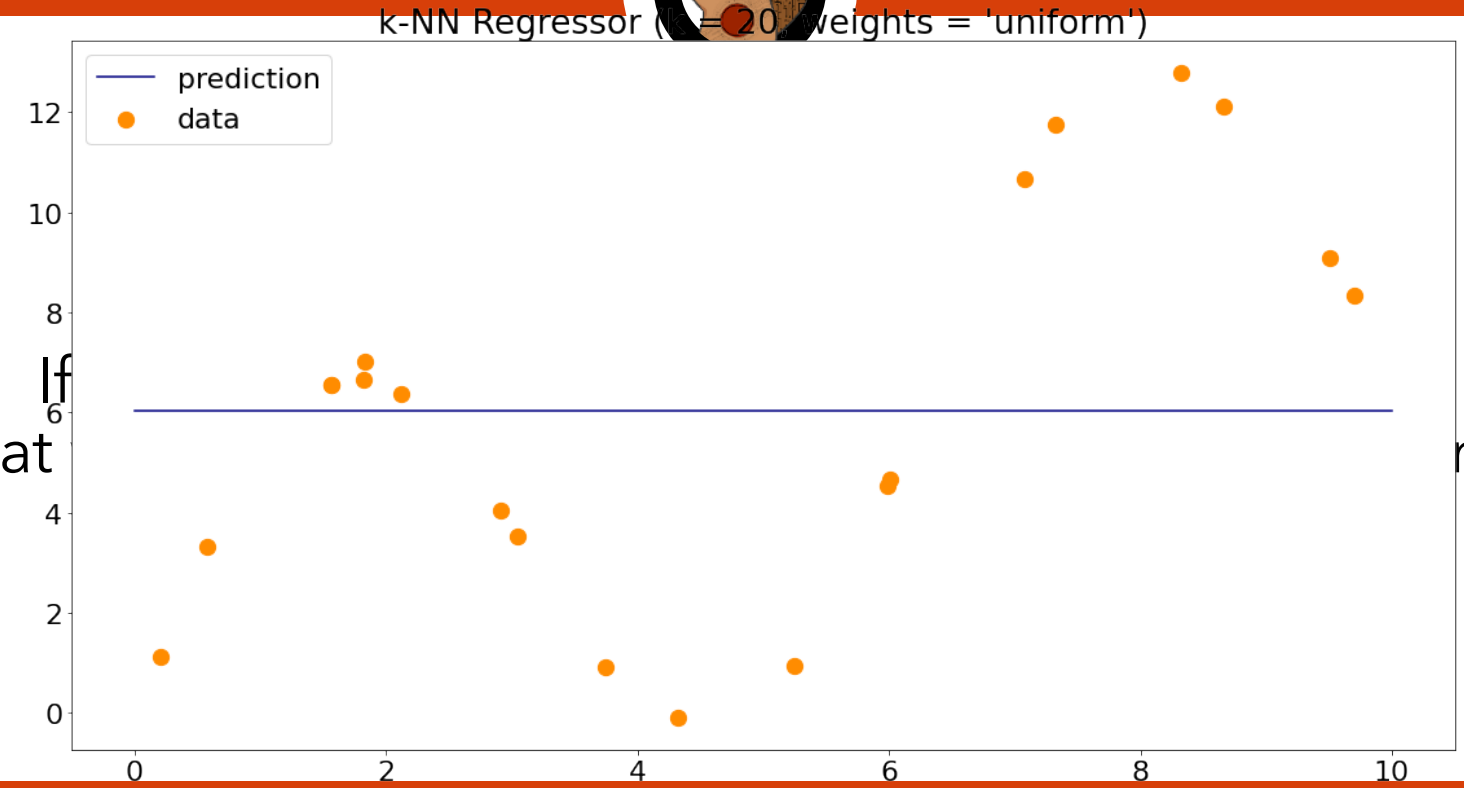
© Stefan Lee

Formalizing k-Nearest Neighbors for Regression



© Stefan Lee

Question Break!



If

what nts?

A The average of the test set

B An average of the nearest two points

C The average of the training set

D No idea.

© Stefan Lee

Practical Issues with k-NN

Finding Nearest Neighbors Can Be Expensive Computationally

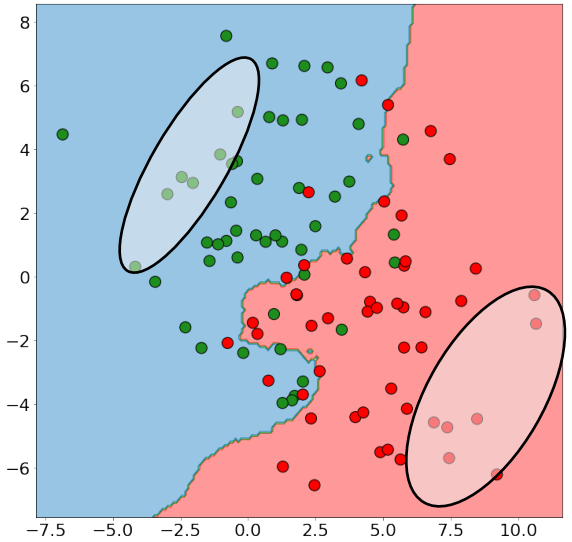
Suppose our training set contains n, d-dimensional points. For a new point, finding the nearest neighbors requires computing distance to all n points.

- **O(nd) for every test point!** (Most distances require O(d) operations)
- **Plenty of work to speed this up with smart data structures to quickly find approximate nearest neighbors – most focus on partitioning space**
  - Kd-trees, Ball-trees, Locality Sensitive Hash, ….

Have to Store All the Datapoints

For massive datasets, requires lots of memory

- Remove "unimportant examples"



© Stefan Lee

Relative Scale of Features Matters for Most Distance Measures

Supposer we want to build a model to predict a person's shoe size using the person's height (feet) and weight (lbs) to make the prediction.

P1: (6.1, 175) **P2:** (5.7, 168) **New Point:** (6.1', 170)

$$\boldsymbol{d(NP, P1) = \sqrt{(6.1 - 6.1)^2 + (175 - 170)^2} = 5}$$

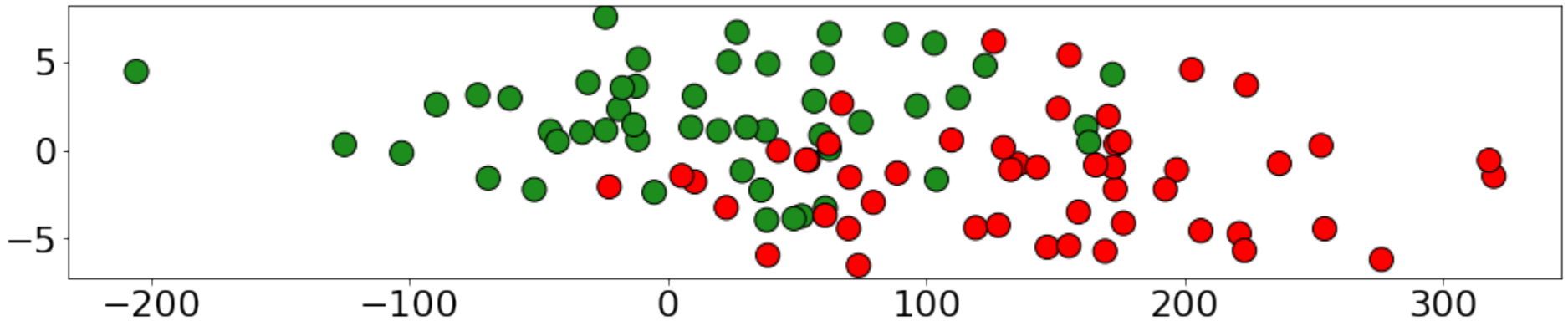
$$\boldsymbol{d(NP, P2) = \sqrt{(6.1 - 5.7)^2 + (170 - 168)^2} \approx 2.04}$$

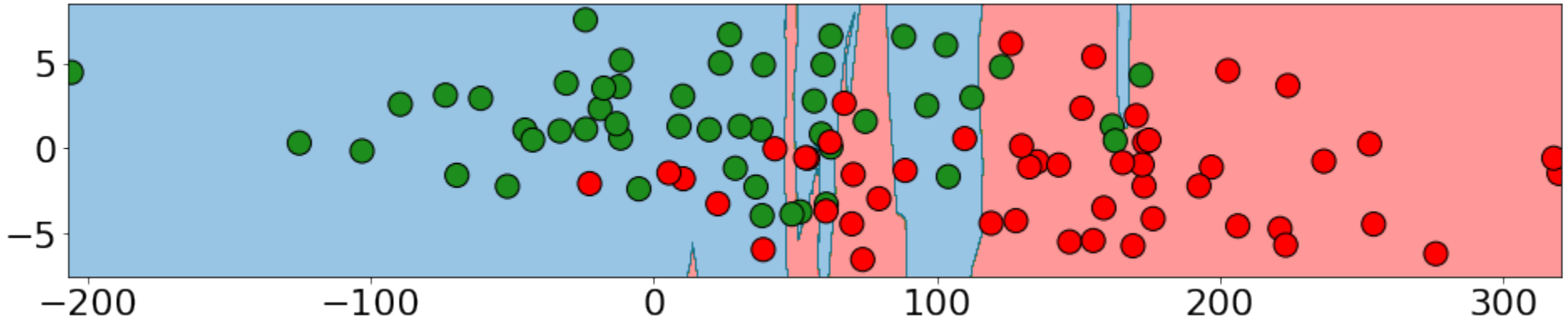
Because weight has a larger range of values, its difference dominates the distance calculation.

Features should be normalized to have the same range of values (e.g. [0,1]), otherwise features with larger ranges will have bigger impact on the distance.

© Stefan Lee

Relative Scale of Features Matters for Most Distance Measures

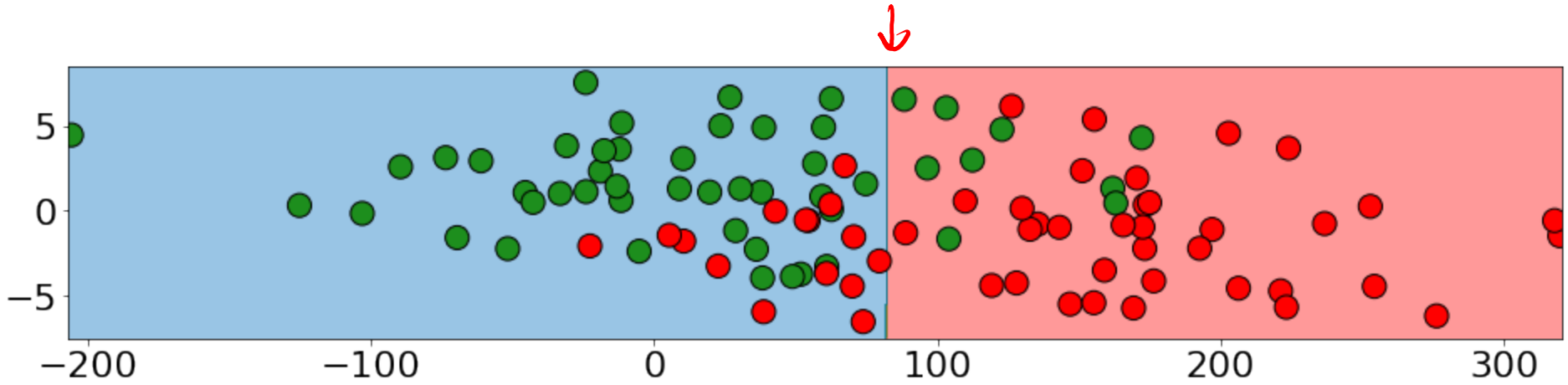




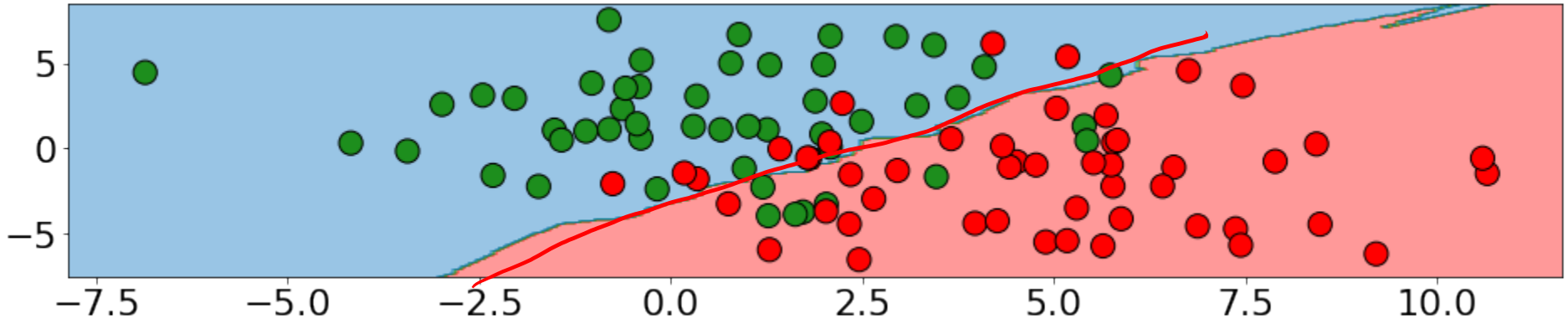
k=5, unwieghted

© Stefan Lee

Relative Scale of Features Matters for Most Distance Measures



k=50, unwieghted



k=50, same feature scale

© Stefan Lee

Hyperparameters of k-Nearest Neighbors

Seems like we've acquired a bunch of knobs we can turn and choices we can make when defining our "simple" model…. how do we choose them?

1) A distance metric (How do we tell which points are "close"?)

Euclidean / Manhattan / Some exotic Minkowski distance?

2) k (How many neighboring points do we look at?)

1? 5? 10? n/4?

3) Output Function (Given a set of neighbors, how do we compute the output?)

To weight or not to weight?

4) Weighting Function (Do all k neighbors contribute equally?)

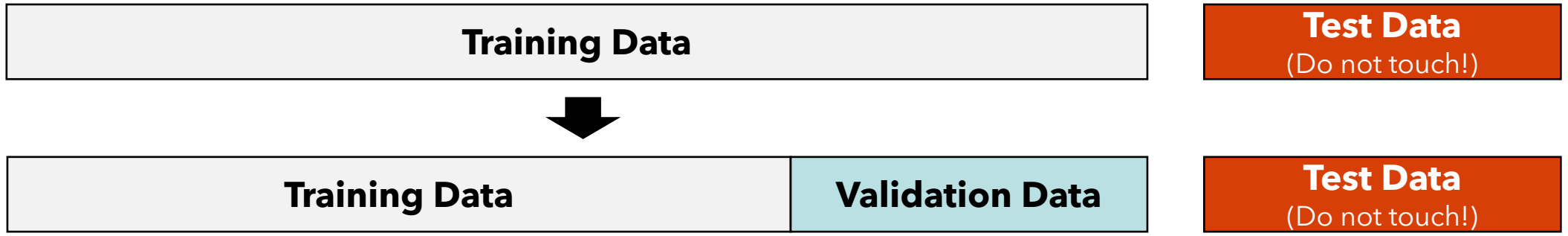
If weighting, what function? $\frac{1}{d(x,x_i)}$**?** $e^{-\frac{d(x,x_i)^2}{\sigma}}$**? If so, what bandwidth/variance?**

© Stefan Lee

Model Selection – The Validation Set

The training set won't help us avoid overfitting and the test set is off-limits.

Get a third set of datapoints and call it the **validation set**. May "create" this set of datapoints by randomly splitting off some fraction of the training set.



Model Selection Procedure:

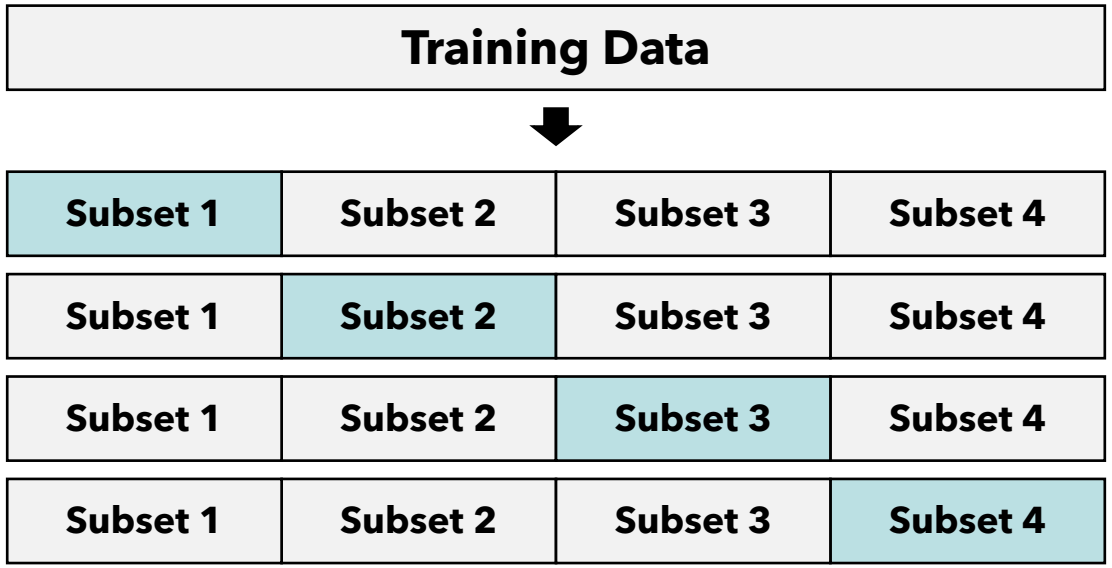
1. Pick a set of hyperparameters
2. Train your model on the **training set**
3. Evaluate on the **validation set**
4. If happy, stop. Otherwise loop back to 1.

At **very end** when you've picked the best hyperparameters you can, then you evaluate on test and report results. No more tuning at this point.

© Stefan Lee

Model Selection – K-fold Cross Validation

Idea: Make many validation sets and evaluate model choices by average performance over them.



Example: 4-fold cross validation

K-Fold Cross Validation Procedure

1. Randomly split the training set into K equally-sized subsets
   - Subsets should have similar distributions

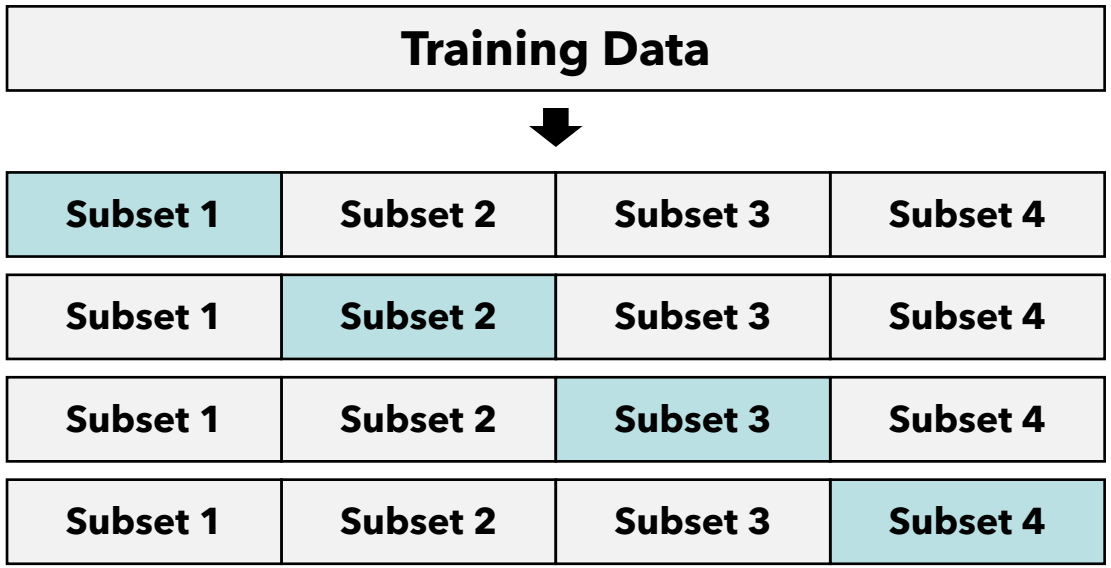
2. For each subset ***i***:
   i. Combine all subsets **except *i*** and train the model on this.
   ii. Evaluate the model on subset ***i***.

3. Compute average performance over these K train/evaluation runs.

© Stefan Lee

Model Selection – K-fold Cross Validation

Idea: Make many validation sets and evaluate model choices by average performance over them.



Example: 4-fold cross validation

Train on subsets 2,3,4. Test on subset 1 → $perf_1$

Train on subsets 1,3,4. Test on subset 2 → $perf_2$

Train on subsets 1,2,4. Test on subset 3 → $perf_3$

Train on subsets 1,2,3. Test on subset 4 → $perf_4$

Estimated validation performance is $\frac{1}{4}(\sum_i perf_i)$

© Stefan Lee

Model Selection – K-fold Cross Validation

Model Selection Procedure:

1. Pick a set of hyperparameters
2. Train your model on the **training set**
3. Evaluate on the **validation set**
4. If happy, stop. Otherwise loop back to 1.

At **very end** when you've picked the best hyperparameters you can, then you evaluate on test and report results. No more tuning at this point.

Model Selection Procedure with K-Fold Cross Validation:

1. Pick a set of hyperparameters
2. Perform **K-Fold Cross Validation** to estimate performance
3. If happy, stop. Otherwise loop back to 1.

At **very end** when you've picked the best hyperparameters you can, then you evaluate on test and report results. No more tuning at this point.

© Stefan Lee

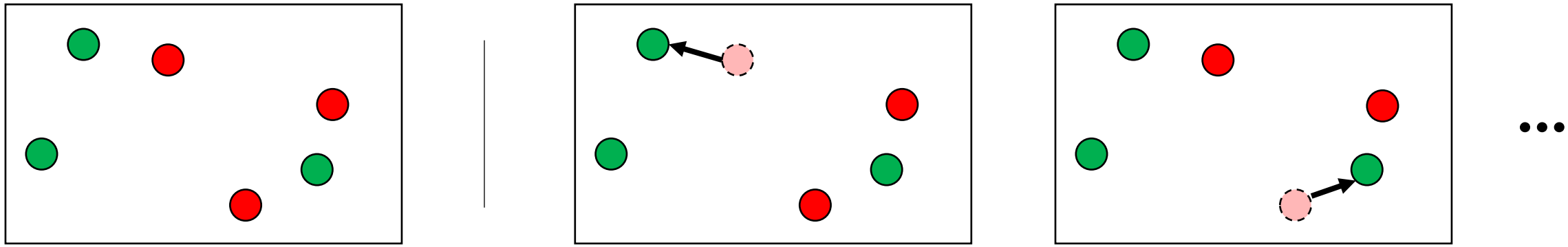
Model Selection – Leave-One-Out Cross Validation

What if K = n? Then we treat only one example as validation at a time. Call it Leave-One-Out.

Same approach as before, but we are going to have to train a LOT of models. May be prohibitively expensive for large, complicated, slow models.

k-NN: LOO cross validation is super easy in k-NN and other "lazy" learners

Simply classify each training point but **don't** let it select itself as a nearest neighbor.

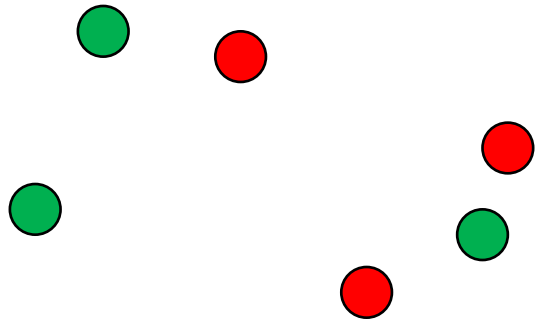


Example: Leave-One-Out Cross Validation with 1-nn

© Stefan Lee

Question Break!

What is the leave-one-out cross validation accuracy for a 1-nearest neighbor model on this dataset:



A $5/6^{th}$

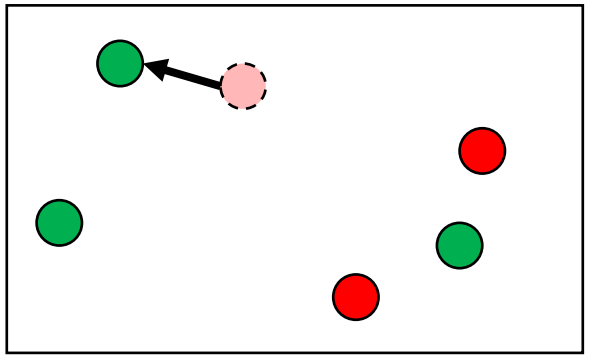
B $2/6^{th}$

C $3/6^{th}$

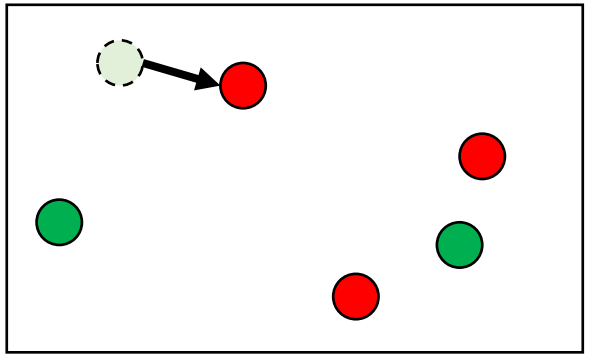
D $1/6^{th}$

© Stefan Lee

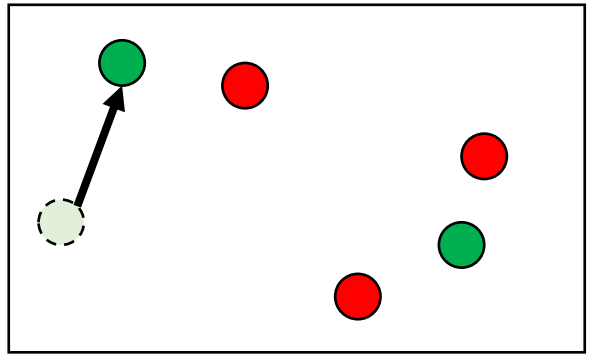
Example Leave-One-Out Cross Validation



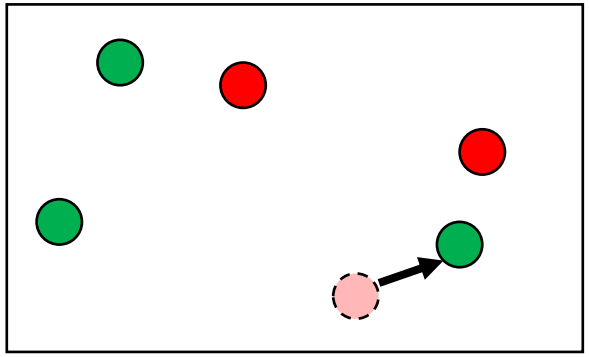
Accuracy: 0



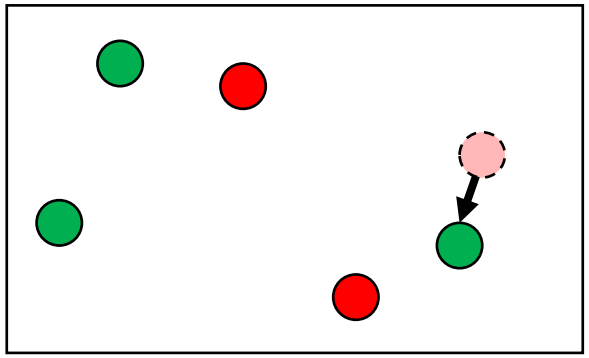
Accuracy: 0



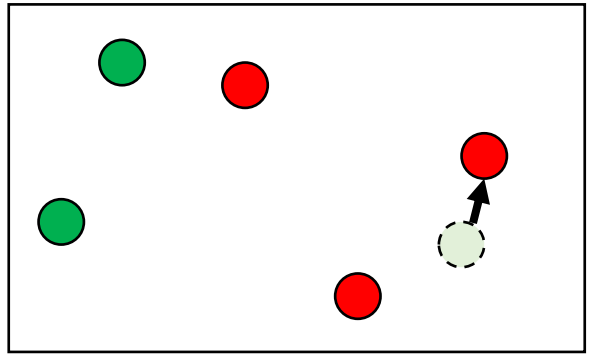
Accuracy: 1



Accuracy: 0



Accuracy: 0



Accuracy: 0

Leave-One-Out Cross Validation Accuracy Estimate: 1/6 or ≈16.6%

© Stefan Lee

Probability / MLE / MAP

© Stefan Lee

Your First Probabilistic Learning Algorithm: MLE

The intuitive answer is that 3/5 "fits the data" the best. We'll formalize that notion this lecture.

Maximum Likelihood Estimation – Find parameters that make the observed data most likely.

1. Assume a probabilistic model of how the data was generated $x \sim P(x; \theta)$ parameterized by some set of parameters θ

2. Find $\hat{\theta}_{MLE}$ that maximizes the probability (or likelihood) of generating the training data under the probabilistic model.

Why MLE?

- Often leads to "natural" or intuitive parameter estimates
- MLE is optimal if model class is correct (e.g. Normal model for normally distributed data)

© Stefan Lee

Baby Steps: Estimating the Bias of a Coin

Dataset: Suppose you observe a coin flipped **6** times such that $\boldsymbol{D} = \{\boldsymbol{H}, \boldsymbol{T}, \boldsymbol{H}, \boldsymbol{H}, \boldsymbol{H}, \boldsymbol{T}\}$

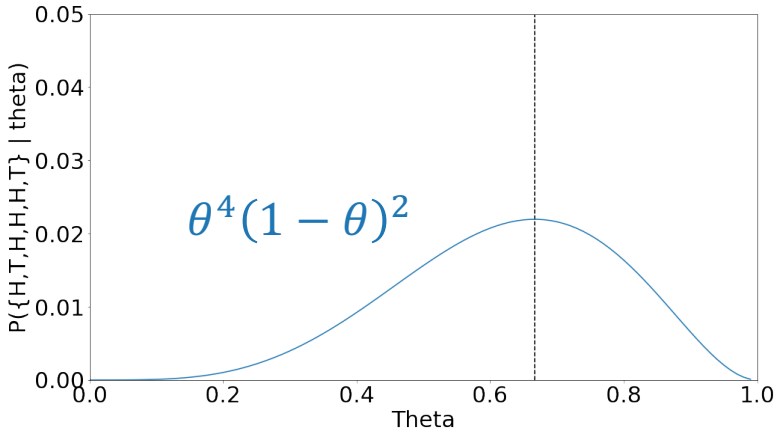
Model Assumption: Assume the probability of getting heads on the coin is some unknown θ.

Write out likelihood of the training data as a function of parameters θ:

$$\mathcal{L}(\theta) = P(D \mid \theta) = P(\{H,T,H,H,H,T\}|\theta) = P(H|\theta)P(T|\theta)P(H|\theta)P(H|\theta)P(H|\theta)P(T|\theta)$$

$$= \theta(1-\theta)\theta\theta\theta(1-\theta) \quad = \theta^4(1-\theta)^2$$

Find $\hat{\theta}_{MLE} = \underset{\theta}{\text{argmax}}\ \mathcal{L}(\theta)$**:**



Any guesses what the peak of the graph is?

$$\frac{4}{6} = \frac{\#Heads}{\#Heads + \#Tails}$$

Is this generally true?

© Stefan Lee

Question Break!

What assumption did we make when we set $P(\{H,T,H,H,H,T\}|\theta) = P(H|\theta)P(T|\theta)P(H|\theta)P(H|\theta)P(H|\theta)P(T|\theta)$?

A θ was greater than 0.

B θ was greater than 0.5

C Each coin flip was independent

D Each coin flip was dependent

© Stefan Lee

General Case: Estimating the Bias of a Coin with MLE

Dataset: Let X_i be a binary random variable (H→1, T→0). Given a dataset $D = \{x_i\}_{i=1}^{N}$ representing a coin being flipped N times. (e.g. {H,T,H,H,H,T} → {1,0,1,1,1,0})

Model Assumption: Assume that X is distributed according to a Bernoulli distribution with parameter θ such that: $P(x) = \theta^x(1-\theta)^{1-x}$

© Stefan Lee

General Case: Estimating the Bias of a Coin with MLE

Dataset: Let X_i be a binary random variable (H→1, T→0). Given a dataset $D = \{x_i\}_{i=1}^{N}$ representing a coin being flipped N times. (e.g. {H,T,H,H,H,T} → {1,0,1,1,1,0})

Model Assumption: Assume that X is distributed according to a Bernoulli distribution with parameter θ such that: $P(x) = \theta^x(1-\theta)^{1-x}$

Write out likelihood of the training data as a function of parameters θ:

$$\mathcal{L}(\theta) = P(D \mid \theta) = \prod_{i=1}^{N} P(x_i \mid \theta) = \prod_{i=1}^{N} \theta^{x_i}(1-\theta)^{1-x_i}$$

Collect all the like terms and sum their exponents

$$= \theta^{\Sigma x_i}(1-\theta)^{\Sigma(1-x_i)}$$

Now our goal is to find θ that maximizes this

$$= \theta^{\#Heads}(1-\theta)^{\#Tails}$$

Find $\hat{\theta}_{MLE} = \underset{\theta}{\text{argmax}}\ \mathcal{L}(\theta)$**:** How can we do this analytically?

© Stefan Lee

We will very often be most interested in the posterior.

Posterior
Belief about theta after evidence and prior belief combined

Likelihood
How likely theta makes the dataset

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

Prior
Belief about theta provided by human

Constant with respect to theta so typically will not address

Important to note – the posterior is a distribution over theta!

© Stefan Lee

Your Second Statistical Learning Algorithm: MAP

Maximum A Posteriori – Find parameters that make the observed data most likely but consider a prior over the parameters.

1. Assume a prior distribution over θ, $P(\theta)$

2. Assume a probabilistic model of how the data is generated:
   -- parameter $\theta \sim P(\theta)$ and then data $x \sim P(x|\theta)$

3. Find $\hat{\theta}_{MAP}$ that maximize the posterior $P(\theta|D) \propto P(D|\theta)P(\theta)$

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

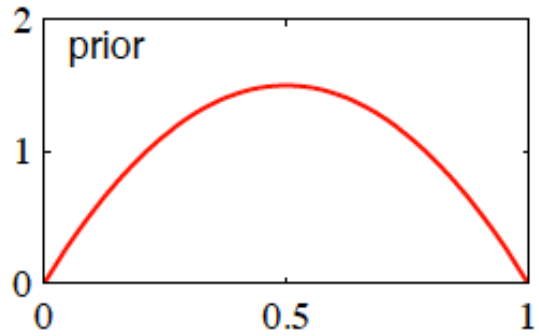
Why MAP?

- Rigorous framework to combine observations (likelihood) with beliefs (prior)

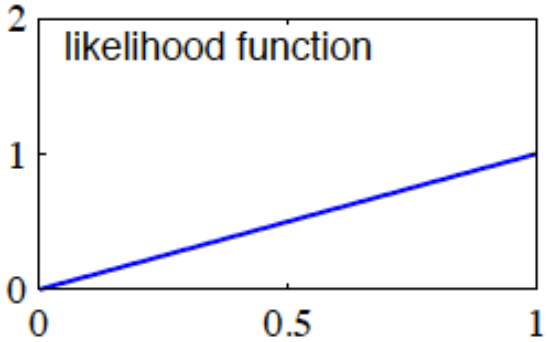
© Stefan Lee

Effect of Prior on Coin Flip

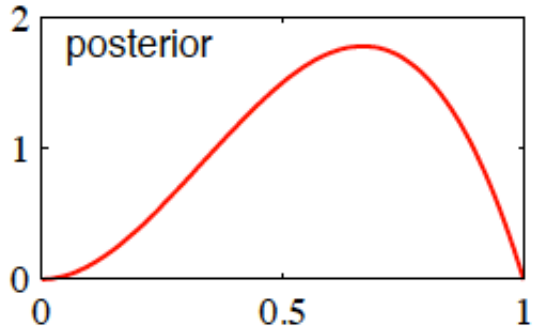
- **Prior = Beta(2,2)**
  - $\theta_{prior} = 0.5$



- **Dataset = {H}**
  - $L(\theta) = \theta$
  - $\theta_{MLE} = 1$



- **Posterior = Beta(3,2)**
  - $\theta_{MAP} = (3-1)/(3+2-2) = 2/3$



© Stefan Lee

Question Break!

What happens as the number of observed data points gets larger?

$$\widehat{\boldsymbol{\theta}}_{\boldsymbol{MAP}} = \frac{\#H + \beta_H - 1}{\#H + \beta_H + \#T + \beta_T - 2}$$

A The fake heads/tails from the prior matter more

B The fake heads/tails from the prior matter less

C You lost me with this Bayesian stuff

D

© Stefan Lee

Question Break!

What is the difference between a MLE and MAP estimate?

A MLE makes assumptions about the parameters before observing data

B One has more vowels

C There are no differences

D MAP maximizes the posterior (likelihood * prior) whereas MLE maximizes the likelihood

© Stefan Lee

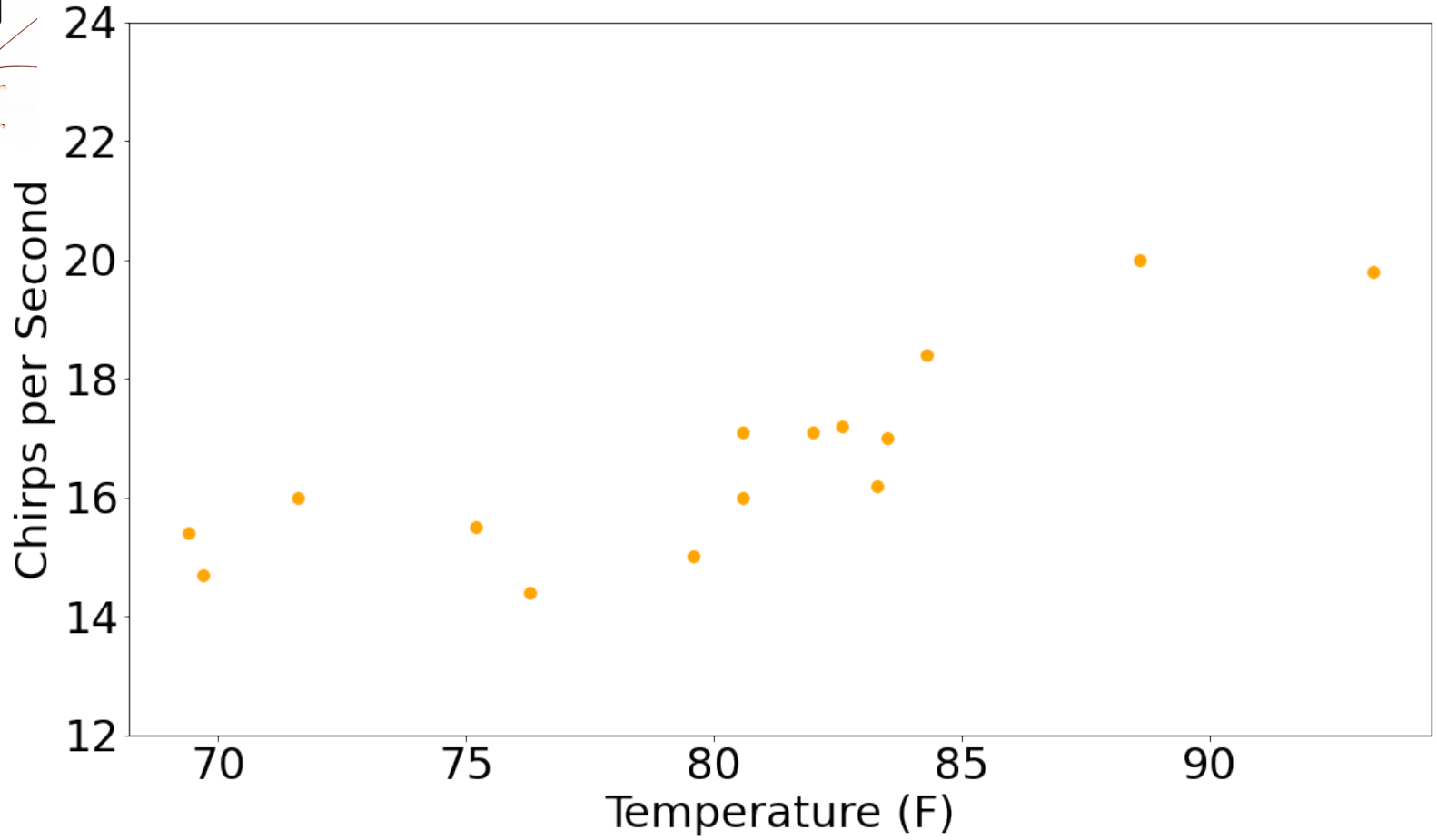
Linear Regression

© Stefan Lee

An Example Linear Regression Problem

Can we predict how annoying crickets will be based on the temperature?

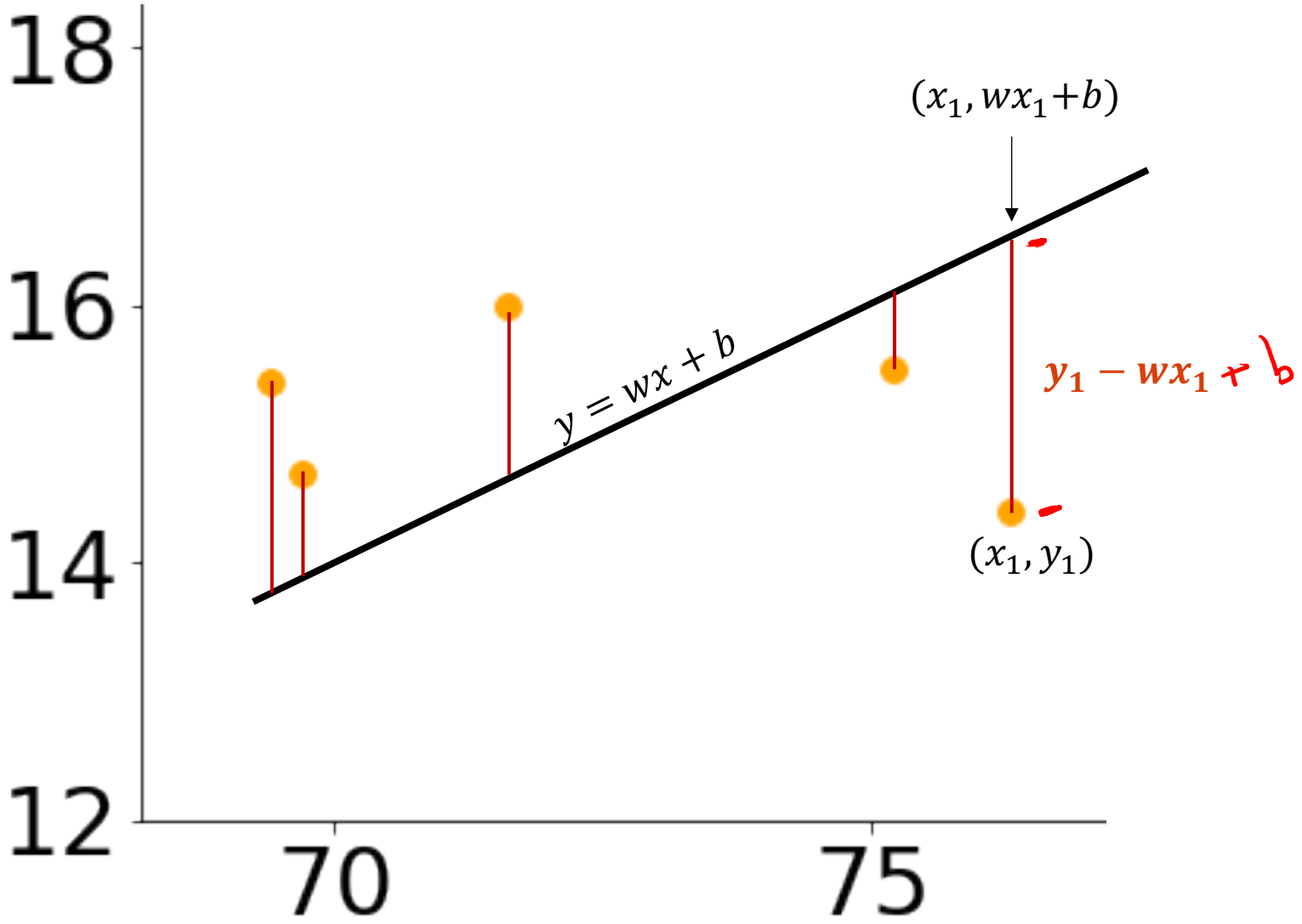
X	Y
F	CPS
88.6	20
71.6	16
93.3	19.8
84.3	18.4
80.6	17.1
75.2	15.5
69.7	14.7
82	17.1
69.4	15.4
83.3	16.2
79.6	15
82.6	17.2
80.6	16.
83.5	17
76.3	14.4



Data from: The Song of Insects by Dr.G.W. Pierce, Harvard College Press

© Stefan Lee

Visualizing Sum of Squared Error



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

Data from: The Song of Insects by Dr.G.W. Pierce, Harvard College Press

© Stefan Lee

Multidimensional Linear Regression

Can we express it entirely as matrix operations?!

X		Y
F	H	CPS
88.6	30	20
71.6	28	16
93.3	32	19.8
84.3	30	18.4
80.6	29	17.1
75.2	24	15.5
69.7	20	14.7
82	30	17.1
69.4	19	15.4
83.3	30	16.2
79.6	28	15
82.6	32	17.2
80.6	30	16.
83.5	30	17
76.3	24	14.4

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x_1^T} \\ \boldsymbol{x_2^T} \\ \vdots \\ \boldsymbol{x_n^T} \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{y_1} \\ \boldsymbol{y_2} \\ \vdots \\ \boldsymbol{y_n} \end{bmatrix}$$

$n \times d$ $\qquad$ $n \times 1$

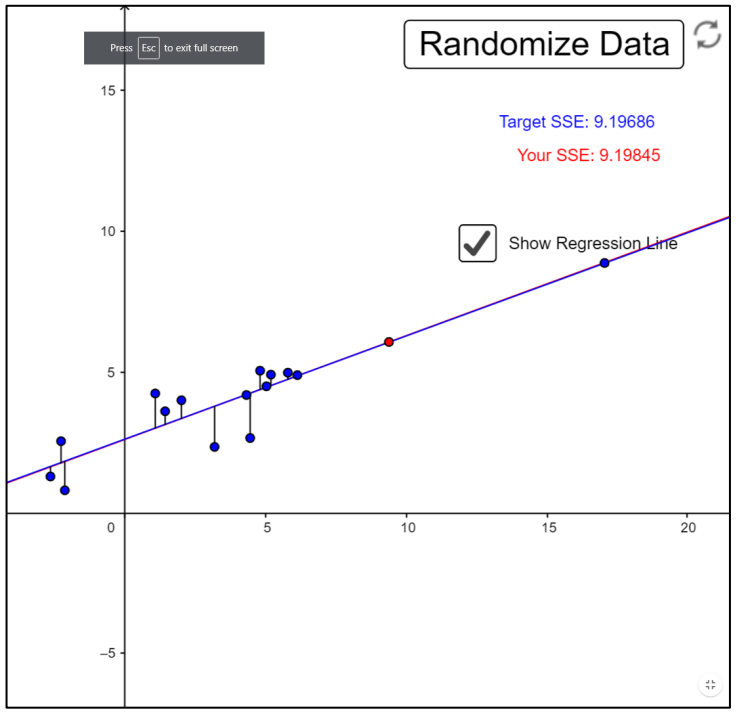
$$\mathbf{w} = [w_1, w_2, \ldots, w_d]^T \qquad d \times 1$$

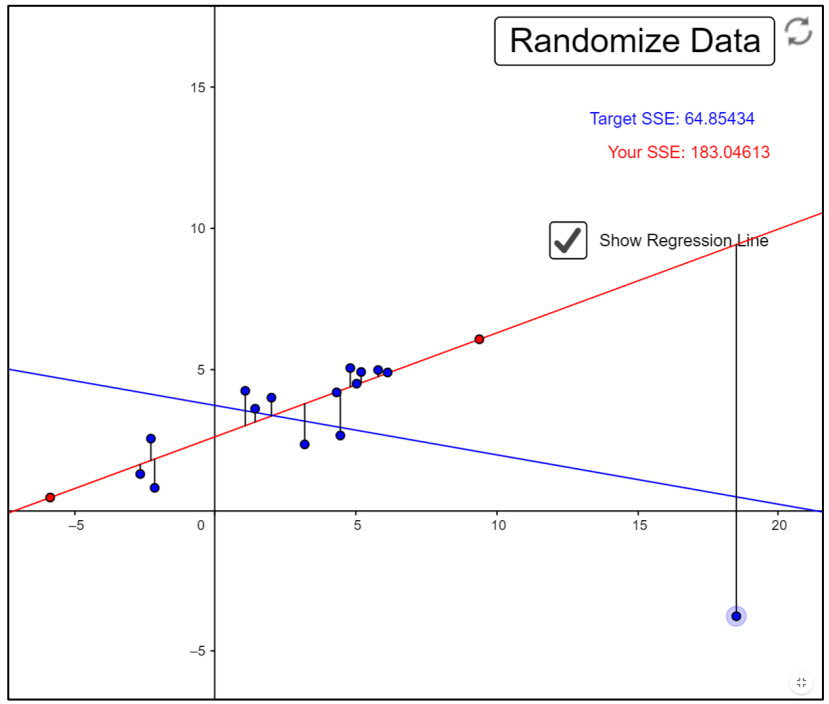
$$SSE(\boldsymbol{w}) = \sum_{i=1}^{n} \left(y_i - \boldsymbol{w^T x_i}\right)^2 = (\boldsymbol{y} - X\boldsymbol{w})^T (\boldsymbol{y} - X\boldsymbol{w})$$

© Stefan Lee

Robustness of Ordinary Least Squares (OLS)

Minimizing sum of squared error is *very* sensitive to outliers.





Probabilistic view: very little density in the tails of Gaussians.

https://www.geogebra.org/m/xC6zq7Zv

© Stefan Lee

$$SSE(\boldsymbol{w}) = (\boldsymbol{y} - X\boldsymbol{w})^T(\boldsymbol{y} - X\boldsymbol{w})$$

Take derivative and set equal to zero. Get closed-form solution:

$$2X^T(\boldsymbol{y} - X\boldsymbol{w}) = \vec{\mathbf{0}}$$

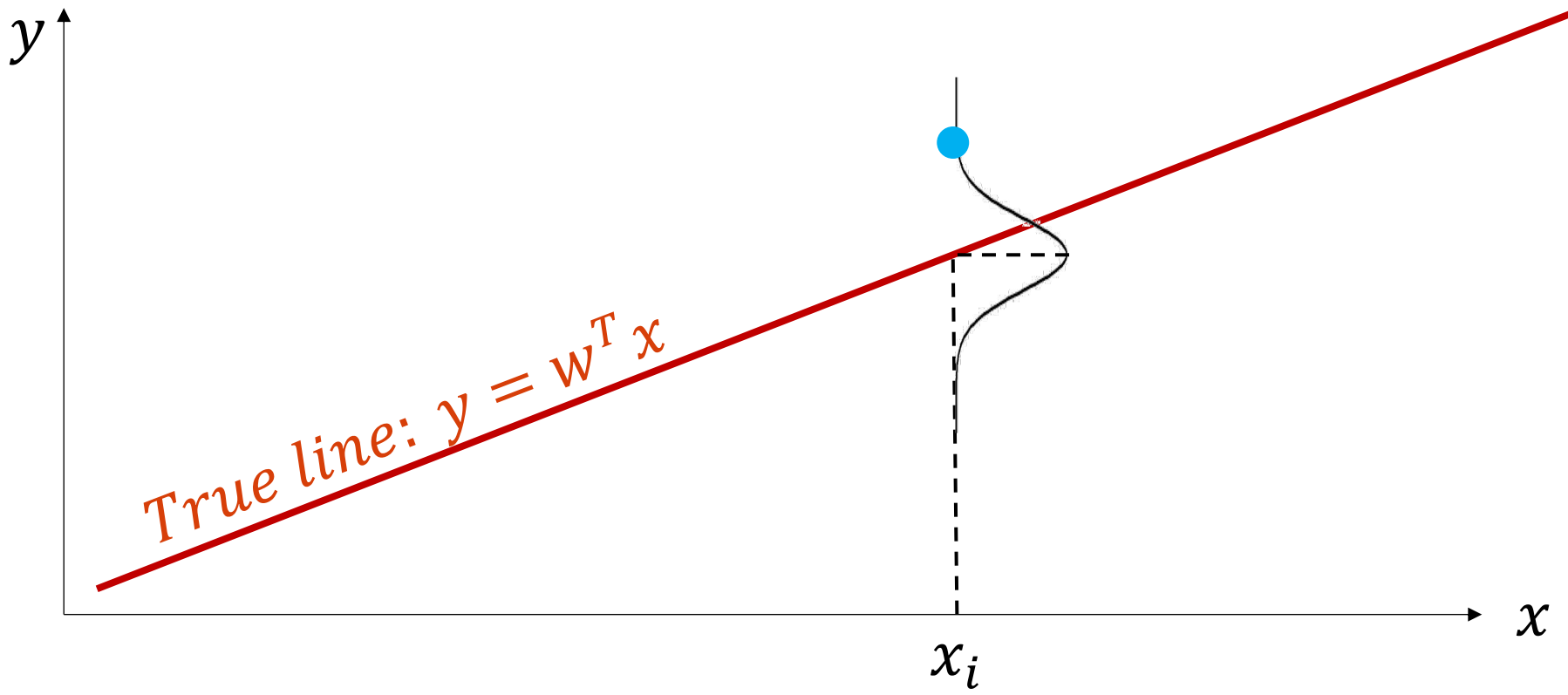
$$\Longrightarrow X^T X\boldsymbol{w} = X^T\boldsymbol{y}$$

$$\Longrightarrow \boldsymbol{w}^* = (X^T X)^{-1}X^T\boldsymbol{y}$$

© Stefan Lee

Our "generative story" for this data:

$$y_i = w^T x_i + \mathcal{N}(0, \sigma)$$



© Stefan Lee

Okay. Still not seeing why we care? Let's start doing MLE.

Dataset: Given a dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ assume the above conditional probability.

Model assumption: $$y_i = w^T x_i + \mathcal{N}(0, \sigma) \implies P(y_i | x_i, w) = \mathcal{N}(w^T x_i, \sigma)$$

Write out likelihood of the training data as a function of parameters θ:

$$\mathcal{L}(\theta) = P(D \mid \theta) = \prod_{i=1}^{N} P(y_i \mid w) = \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i - \boldsymbol{w^T x_i})^2}{2\sigma^2}}$$

Yikes. Let's apply a log and write the log-likelihood to clean this up:

$$\mathcal{LL}(\theta) = P(D \mid \theta) = N \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

© Stefan Lee

Let's view this from a different angle

Wait… this part is looking familiar…

$$\mathcal{LL}(\boldsymbol{\theta}) = N \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x_i})^2$$

How do we maximize this log-likelihood with respect to w?

Find w^* such that $\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x_i})^2$ is as small as possible

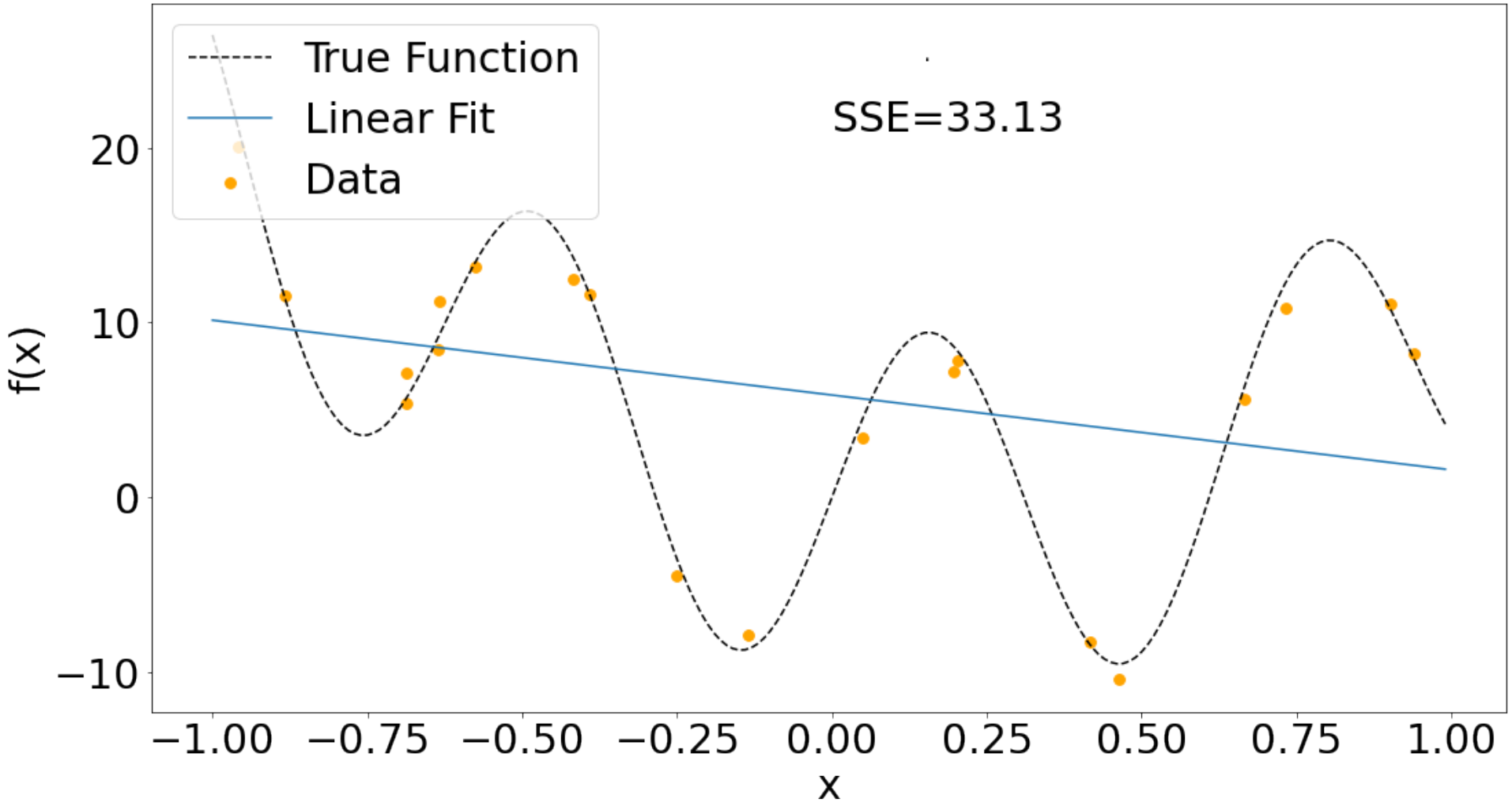
AKA find weights that minimize the sum of squared error!

Linear regression is just MLE of a linear model with Gaussian noise!

© Stefan Lee

Beyond Lines but Still Linear?

Fitting lines seems cool but many, many functions of interest aren't lines.

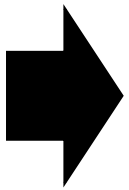


© Stefan Lee

Beyond Lines but Still Linear?

Idea: Solve a linear regression problem in a feature space that is non-linear in the original input! For example, could add a x^2 term.

X		Y
	x	Y
1	-0.25	-4.47
1	0.9	11.08
1	0.46	-10.44
1	0.2	7.19
1	-0.69	7.08
1	-0.69	5.38
1	-0.88	11.55
1	0.73	10.85
1	0.2	7.85
1	0.42	-8.31



X'			Y
	x	x^2	Y
1	-0.25	0.06	-4.47
1	0.9	0.81	11.08
1	0.46	0.21	-10.44
1	0.2	0.04	7.19
1	-0.69	0.48	7.08
1	-0.69	0.48	5.38
1	-0.88	0.77	11.55
1	0.73	0.53	10.85
1	0.2	0.04	7.85
1	0.42	0.18	-8.31

Solve for w such that:

$$y_i = w_0 1 + w_1 x_i + w_2 x_i^2$$

Linear function of non-linear transformations of x

$$y_i = \begin{bmatrix} 1, x_i, x_i^2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

© Stefan Lee

Let's generalize this further.

Each data point has d features. Let $\boldsymbol{\Phi}$: $\mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ be our basis function. Can make any arbitrary choice we want here based on how the data looks.

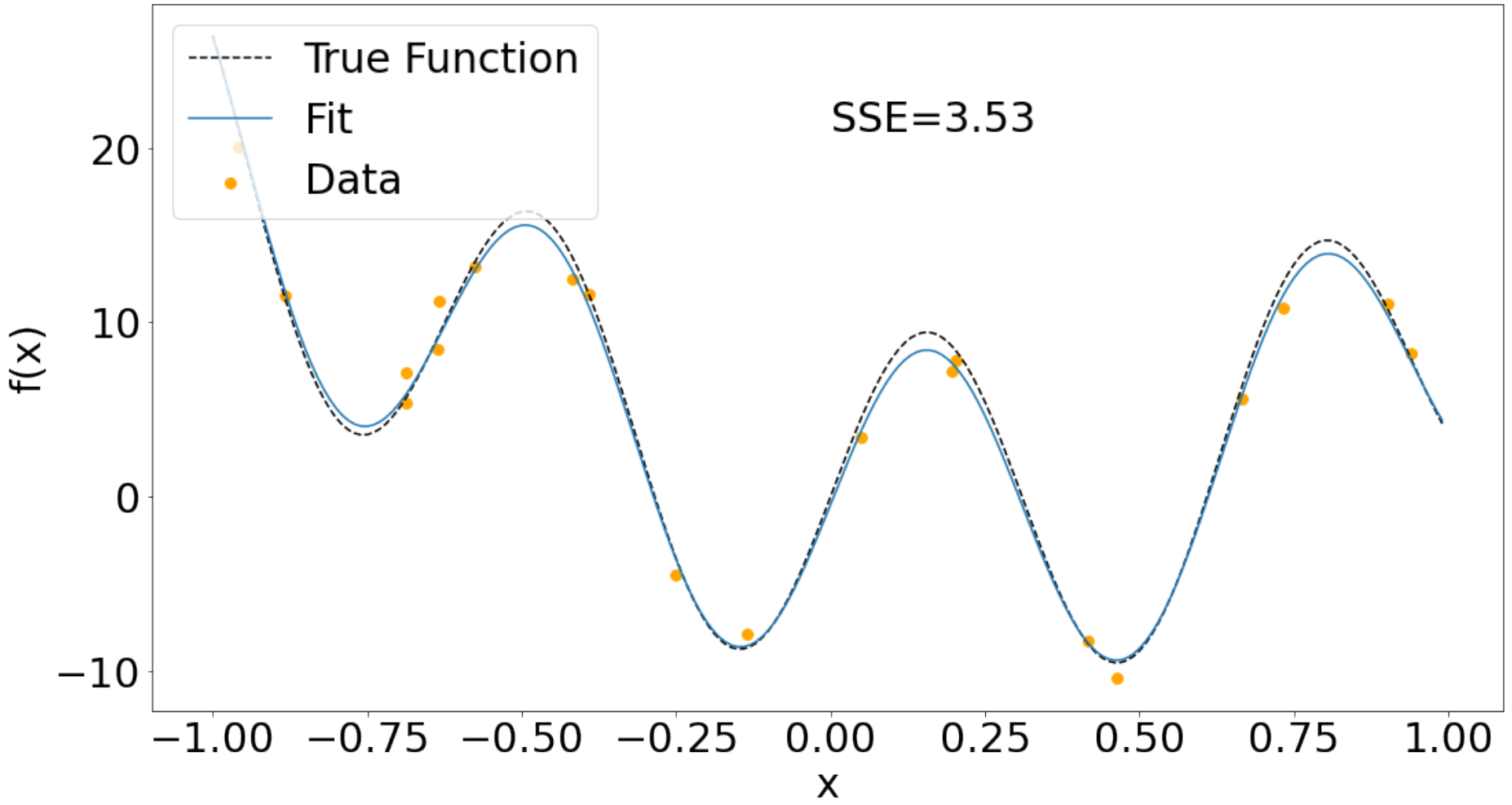
$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}^{\boldsymbol{T}}_{1\times d} \qquad \boldsymbol{\Phi}_{\mathrm{A}}(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ \sin(x_1) \\ \cos(x_2) \end{bmatrix}^{\boldsymbol{T}} \qquad \boldsymbol{\Phi}_{\mathrm{B}}(x) = \begin{bmatrix} 1 \\ x_1 x_2 \\ x_1 x_3 \\ x_1 x_4 \\ \vdots \\ x_n x_n \end{bmatrix}^{\boldsymbol{T}} \qquad \boldsymbol{\Phi}_{\mathrm{C}}(x) = \begin{bmatrix} 1 \\ \prod x_i \\ \sqrt{x_1} \end{bmatrix}^{\boldsymbol{T}}$$

No matter what we choose, the procedure is the same. Replace each $\boldsymbol{x}$ (row) in our data matrix with $\boldsymbol{\Phi}(\mathbf{x})$, then solve the linear regression problem.

© Stefan Lee

Linear Regression over Basis Functions

$$\boldsymbol{\Phi}(x) = [1 \quad x \quad x^2 \quad sin(10x)]$$



© Stefan Lee

Question Break!

Can I use linear regression over basis functions to find the weights w for the following function?

$$w_1 x_1 x_2 + w_2 x_2 \log x_2$$

A Yes

B No

C Maybe if you are tricky.

D

© Stefan Lee

Question Break!

Can I use linear regression over basis functions to find the weights w for the following function?

$$x_1^{w_1} x_2^{w_2}$$

A Yes

B No

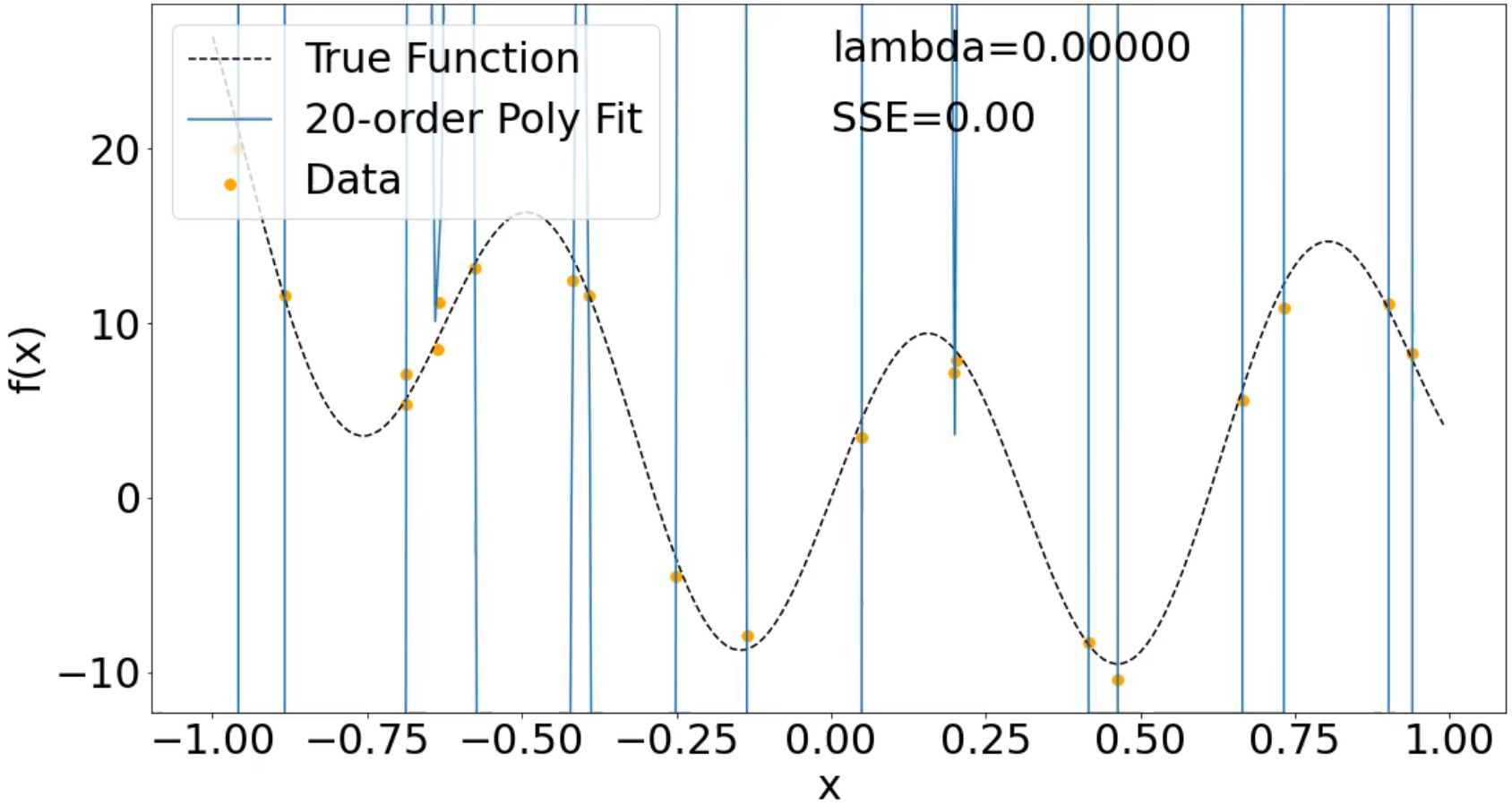
C Maybe if you are tricky.

D

© Stefan Lee

Effect of Regularization

$$w^* = \underset{w}{argmin} \quad (\boldsymbol{y} - X\boldsymbol{w})^T(\boldsymbol{y} - X\boldsymbol{w}) + \lambda \boldsymbol{w^T w} \quad \Rightarrow \quad \boldsymbol{w}^* = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y}$$



© Stefan Lee

Summary of Regularization

Regularization is a much more general concept in machine learning. Consider any learning task to minimize a loss $\mathcal{L}(w)$ on the training data – could add a regularizer term.

$$w^* = \underset{w}{argmin}\, \mathcal{L}(w) + \lambda * regularizer(w)$$

Generally speaking, larger $\lambda's$ lead to simpler models and reduce overfitting, smaller $\lambda's$ lead to more complex models but improve fit on training data.

Most commonly used regularizers are based on the norm of the weight vector.

© Stefan Lee

Logistic Regression

© Stefan Lee

Why not regress probabilities directly?

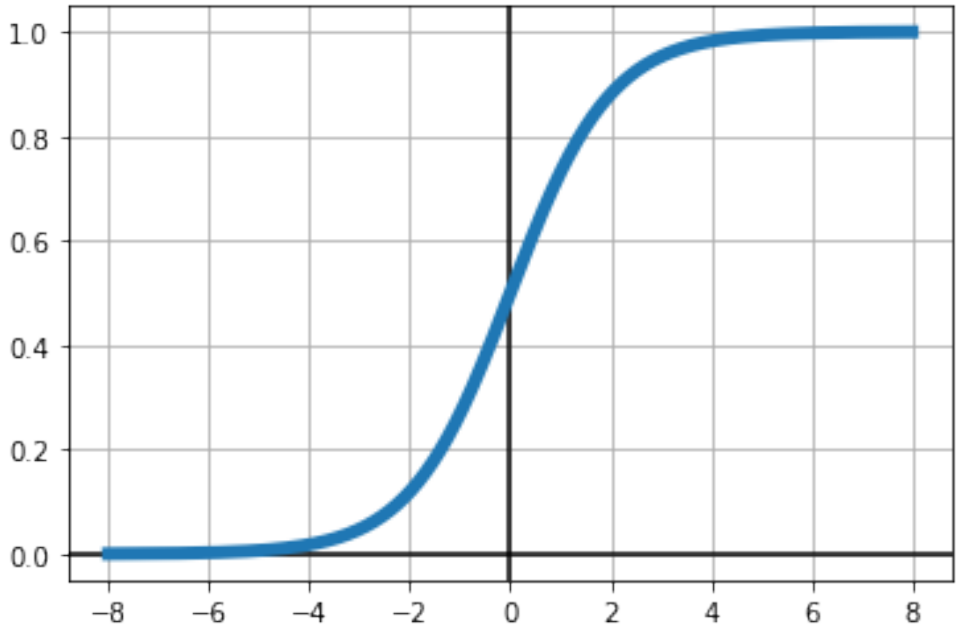
Introducing the logistic function:

- May also see it referred to as a sigmoid function or "logit" function

Logistic Function

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

Maps $(-\infty, \infty)$ to $(0,1)$



© Stefan Lee

Logistic Regression Model Assumption:

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w}) = \sigma(\boldsymbol{w^T x_i}) = \frac{1}{1 + e^{-\boldsymbol{w^T x_i}}}$$

$$P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w}) = 1 - \sigma(\boldsymbol{w^T x_i}) = \frac{e^{-\boldsymbol{w^T x_i}}}{1 + e^{-\boldsymbol{w^T x_i}}}$$

© Stefan Lee

Logistic Regression Model's Decision Boundary:

Predict 1 if

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w}) > P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w})$$

Divide both sides by $P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w})$

$$\Rightarrow \quad \frac{P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w})}{P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w})} > 1$$

Just some algebra

$$\Rightarrow \quad e^{\boldsymbol{w^T x}} > 1$$

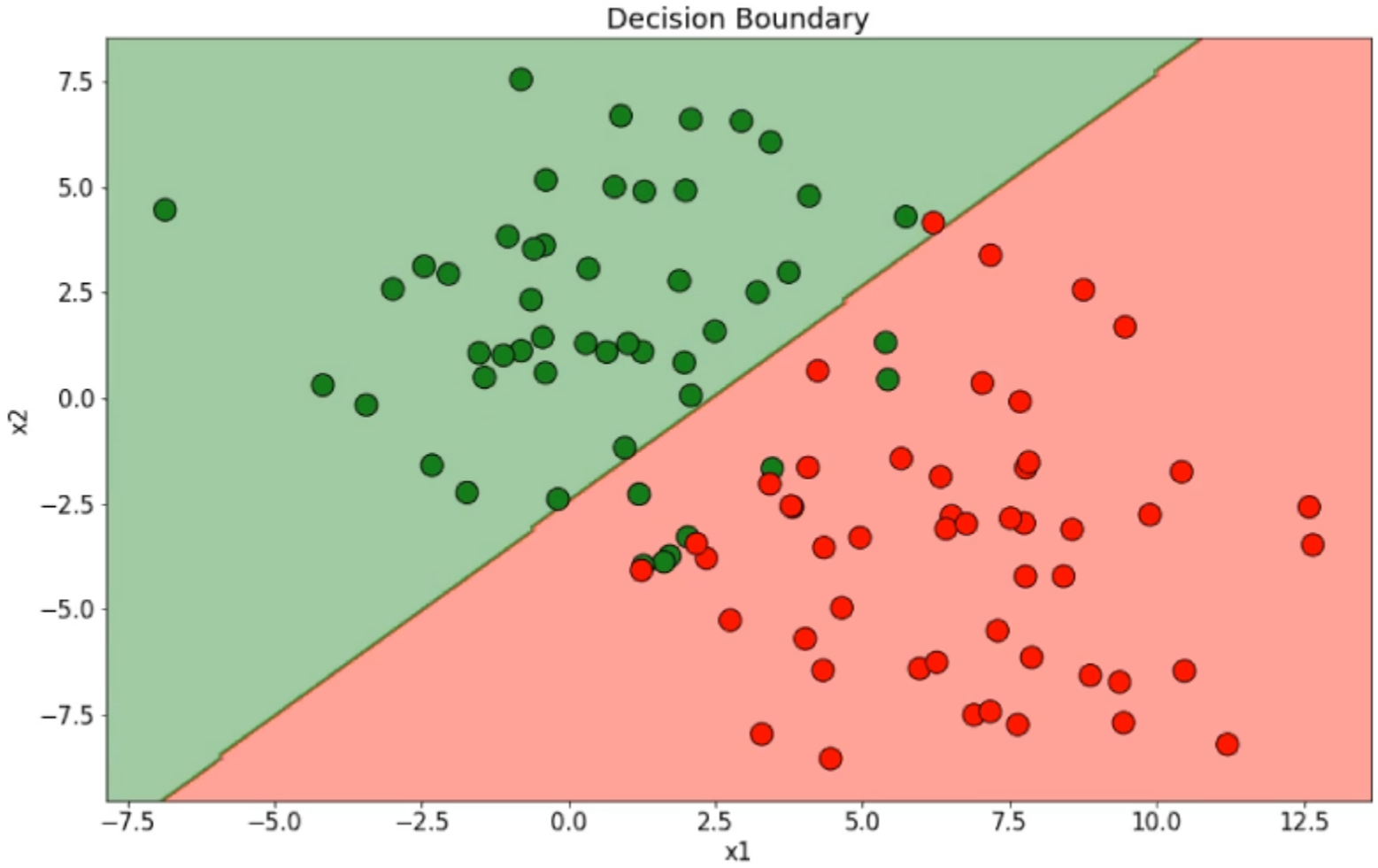
Log of both sides

$$\Rightarrow \quad \boldsymbol{w^T x} > 0$$

This tells us the decision boundary is the line $w^T x = 0$

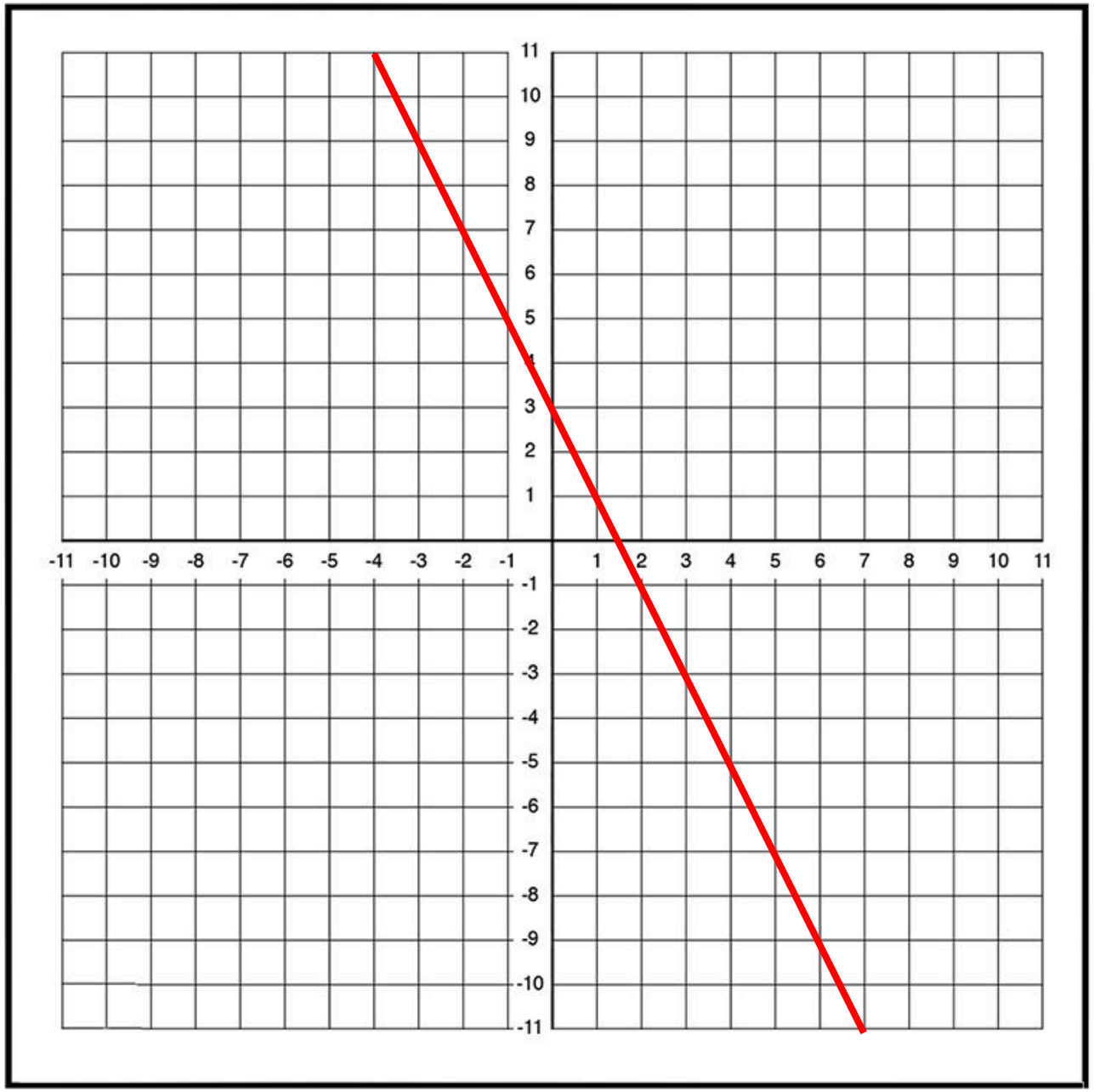
© Stefan Lee

Example logistic regression decision boundary – linear in the features.



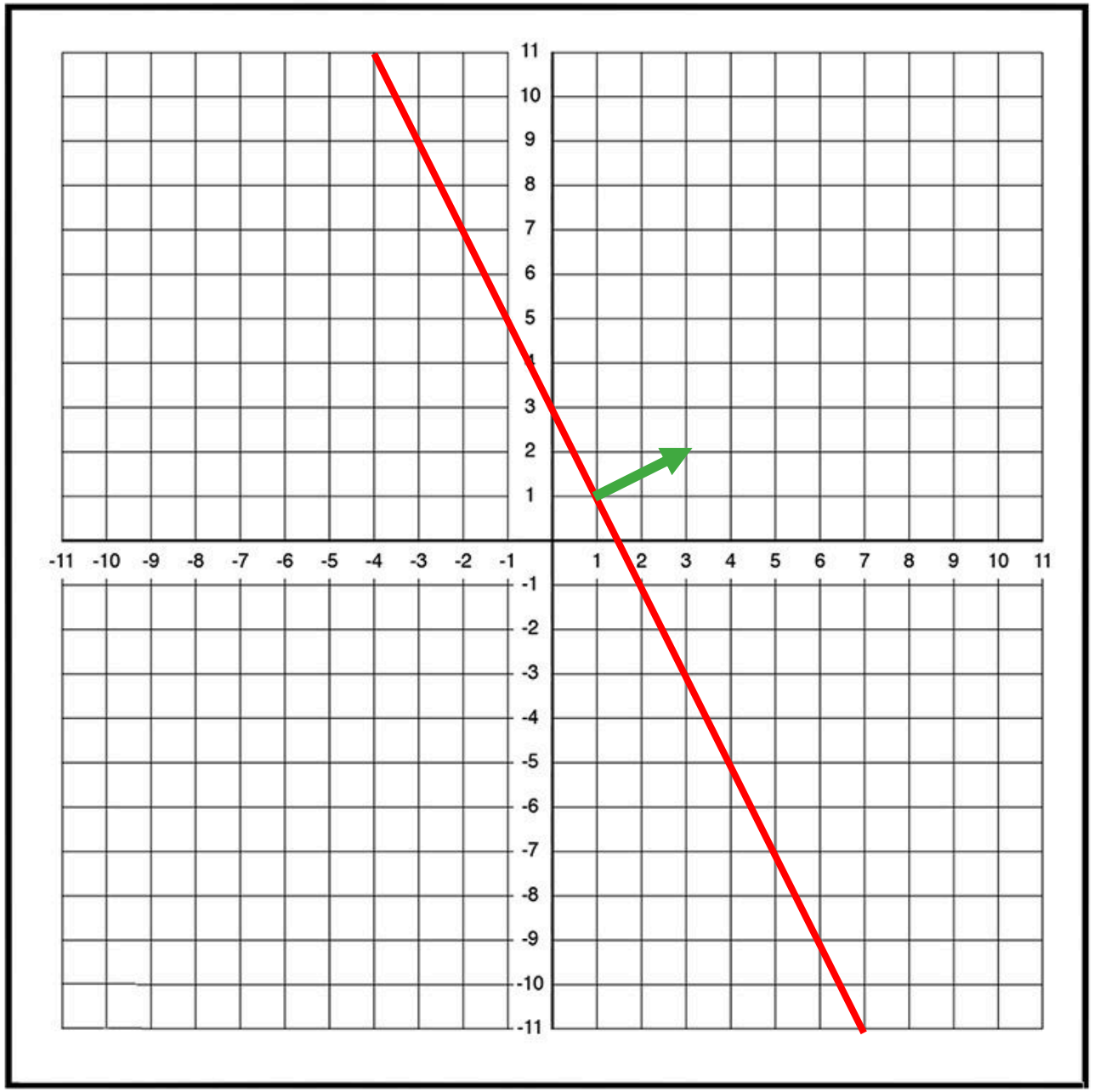
© Stefan Lee

Let's Be Sure We Understand Lines



© Stefan Lee

Let's Be Sure We Understand Lines



$$y = -2x + 3$$

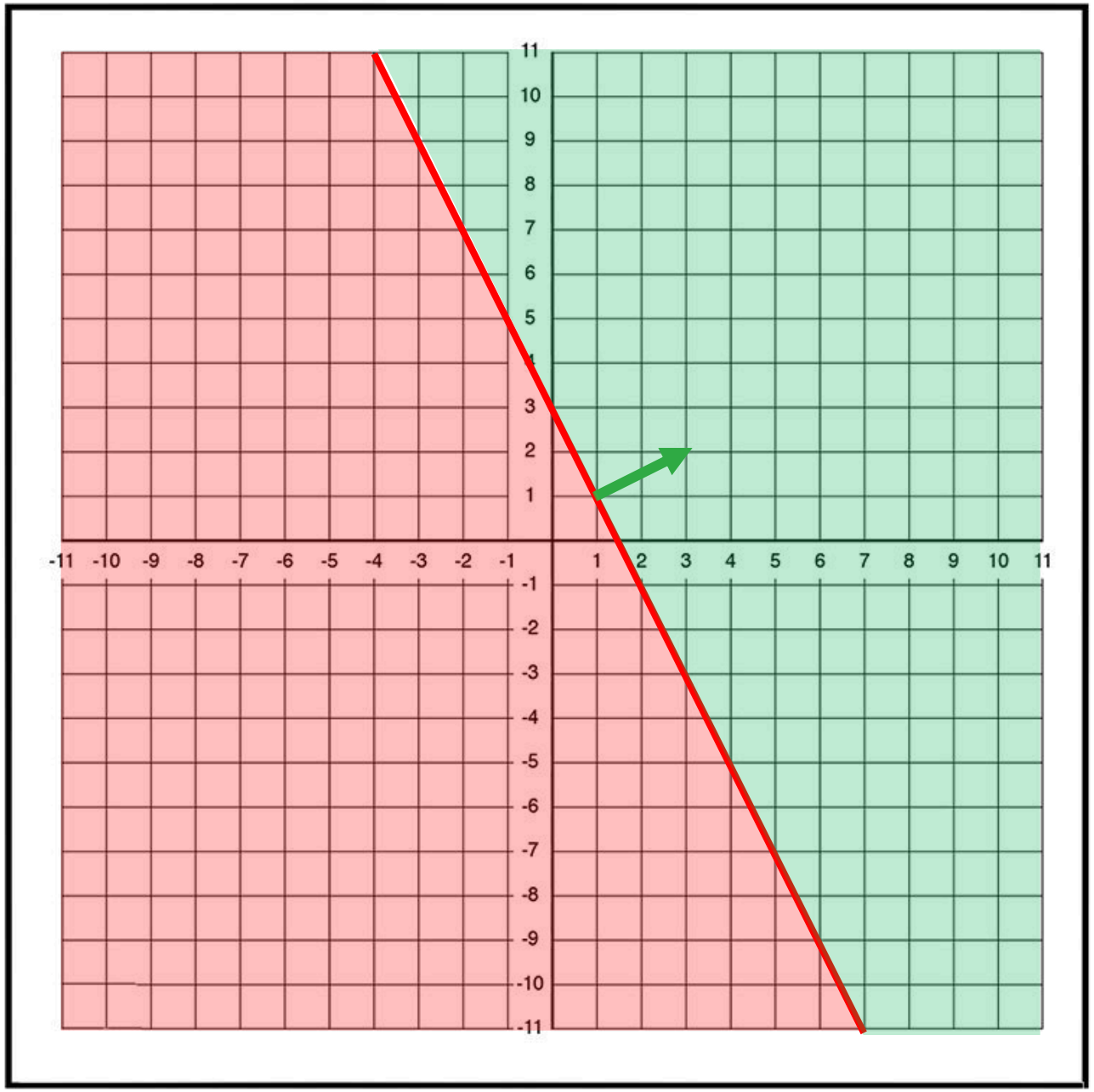
$$-3 + y + 2x = 0$$

$$-3 + [1 \;\; 2] \begin{bmatrix} y \\ x \end{bmatrix} = 0$$

$$w = [1 \;\; 2] \qquad b = -3$$

© Stefan Lee

Let's Be Sure We Understand Lines



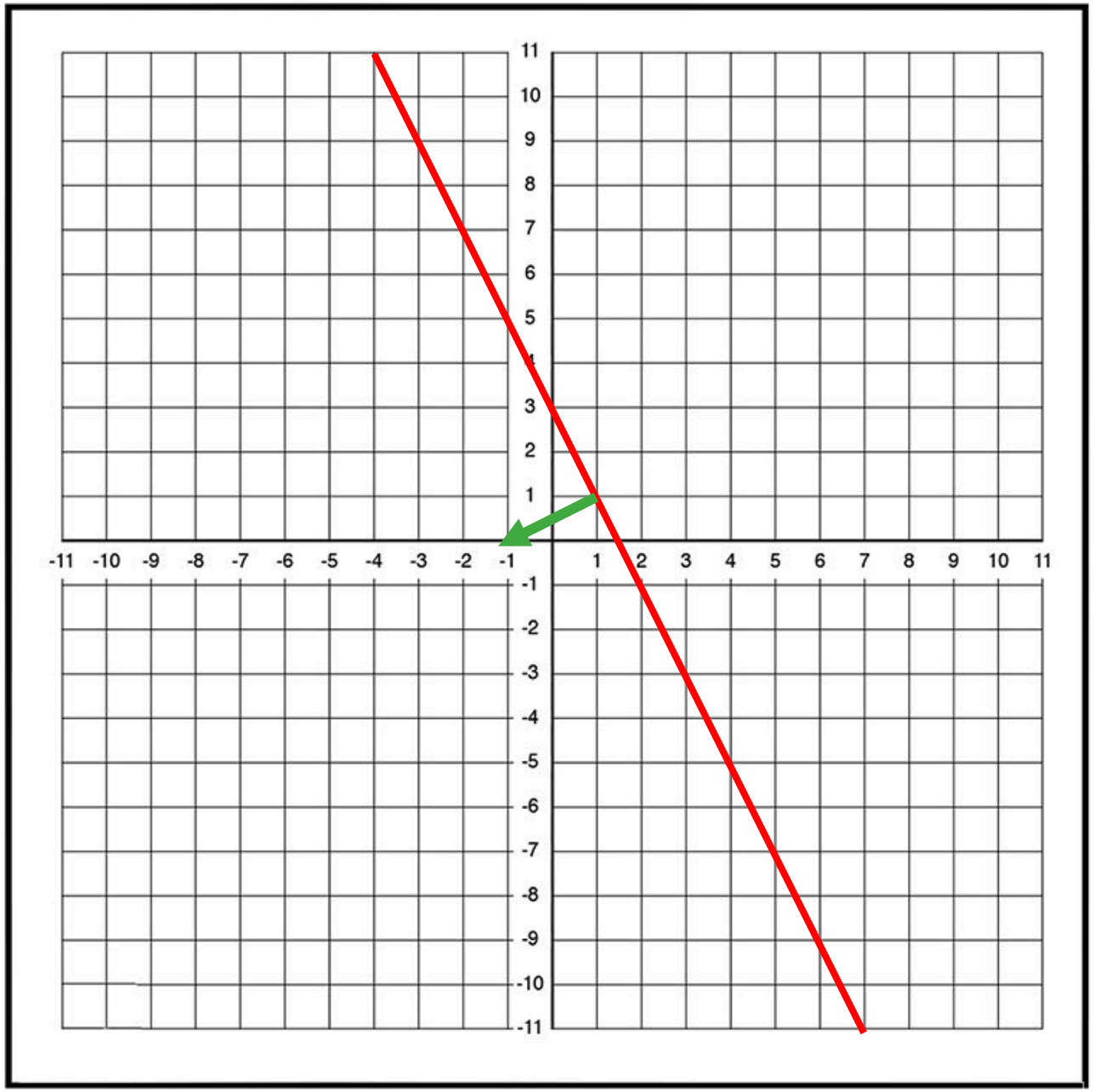
$$-3 + [1 \;\; 2] \begin{bmatrix} y \\ x \end{bmatrix} = 0$$

$$w = [1 \;\; 2] \qquad b = -3$$

$$-3 + [1 \;\; 2] \begin{bmatrix} y = 0 \\ x = 0 \end{bmatrix} = -3$$

© Stefan Lee

Let's Be Sure We Understand Lines



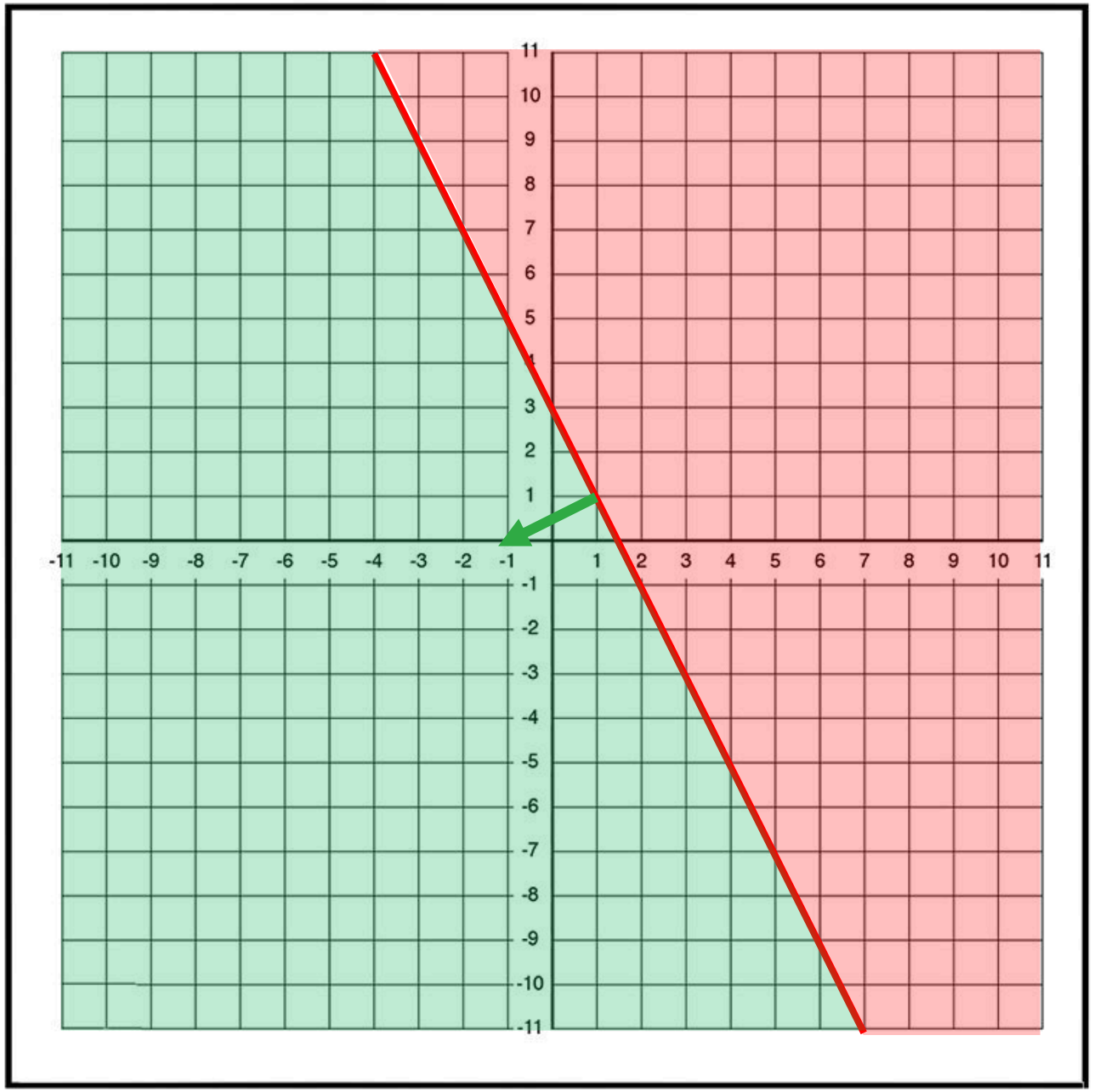
$$-3 + [1 \;\; 2] \begin{bmatrix} y \\ x \end{bmatrix} = 0$$

$$3 + [-1, -2] \begin{bmatrix} y \\ x \end{bmatrix} = 0$$

$$w = [-1, -2] \quad b = 3$$

© Stefan Lee

Let's Be Sure We Understand Lines



$$-3 + [1 \;\; 2] \begin{bmatrix} y \\ x \end{bmatrix} = 0$$

$$3 + [-1, -2] \begin{bmatrix} y \\ x \end{bmatrix} = 0$$

$$w = [-1, -2] \quad b = 3$$

$$3 + [-1, -2] \begin{bmatrix} y = 0 \\ x = 0 \end{bmatrix} = 3$$

© Stefan Lee

MLE for Logistic Regression

So we have an expression for the gradient but can't find the optimal analytically…

$$\nabla - logP(D \mid w) = \sum_{i=1}^{N} \nabla l_i = \sum_{i=1}^{N} \left(\sigma\left(\boldsymbol{w^T x_i}\right) - y_i\right)\mathbf{x_i}$$

Gradient Descent Algorithm for Logistic Regression

w = random(d) // randomly initialize weight vector

Repeat:

$\nabla = \vec{0}$

for each example $x_i, y_i \in D$: // compute gradient

$\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \frac{1}{1+e^{-\boldsymbol{w^T x_i}}}$ // compute the prediction for this point

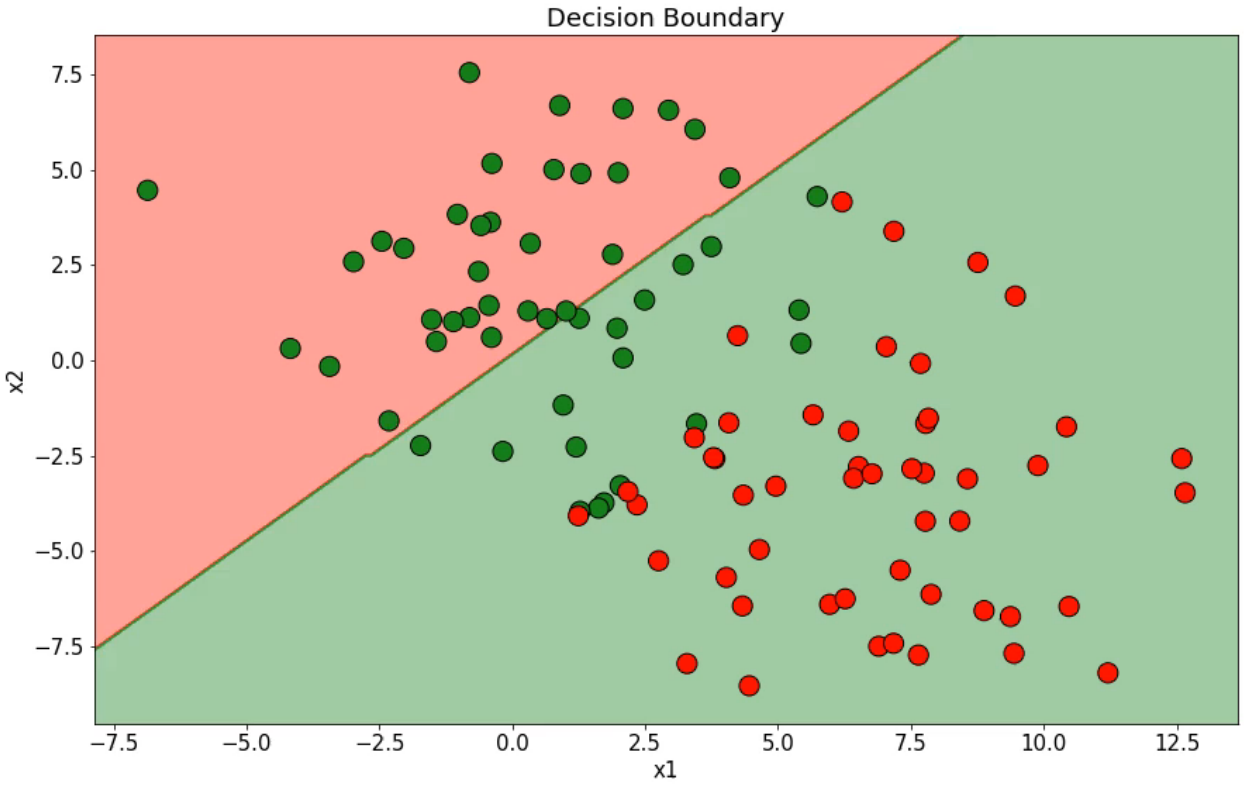
$\boldsymbol{\nabla} \mathrel{+}= (\hat{y}_i - y_i)\boldsymbol{x_i}$ // compute gradient of loss of point and sum

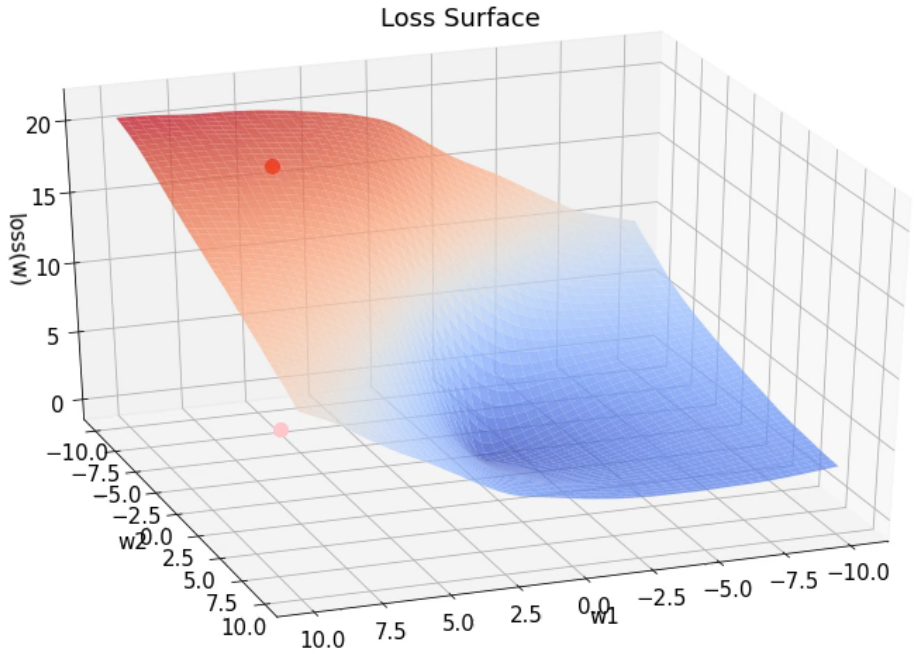
$\mathbf{w} = \mathbf{w} - \alpha\boldsymbol{\nabla}$ // take a step in the opposite direction of gradient

Until $|\nabla| \leq \epsilon$ // keep taking steps until convergence

© Stefan Lee

An example of Logistic Regression with Gradient Descent





© Stefan Lee

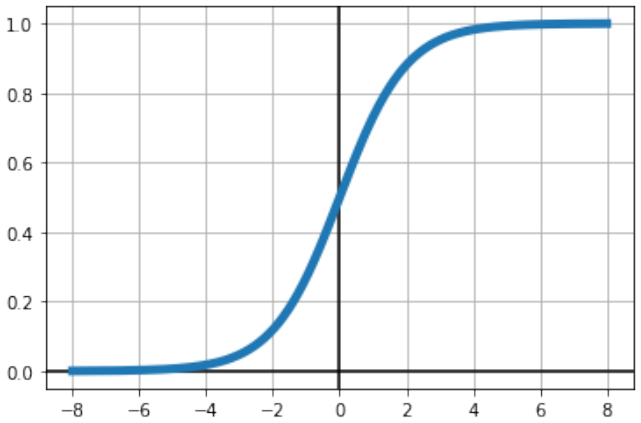
Perceptron

© Stefan Lee

Perceptron vs Logistic Regression

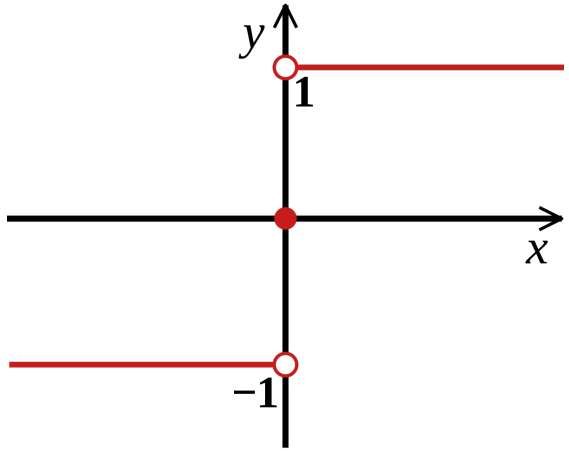
Logistic Function

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



Sign Function

$$sign(z) = \begin{cases} 1 & z \geq 0 \\ -1 & else \end{cases}$$



© Stefan Lee

Perceptron vs Logistic Regression

Logistic Regression

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w}) = \sigma(\boldsymbol{w^T x_i}) = \frac{1}{1 + e^{-\boldsymbol{w^T x_i}}}$$

$$y_i = argmax_y \, P(y|\boldsymbol{x_i}; \boldsymbol{w})$$

Linear Classifier

Probabilistic Interpretation

Trained with gradient descent or other optimization methods

Perceptron

$$y_i = sign(\boldsymbol{w^T x_i})$$

Linear Classifier

No useful probabilistic interpretation (mostly just geometry)

Trained with the "perceptron" algorithm

© Stefan Lee

The Perceptron: Learning Algorithm

Trained with a simple iterative algorithm

Perceptron Learning Algorithm

w = random(d)

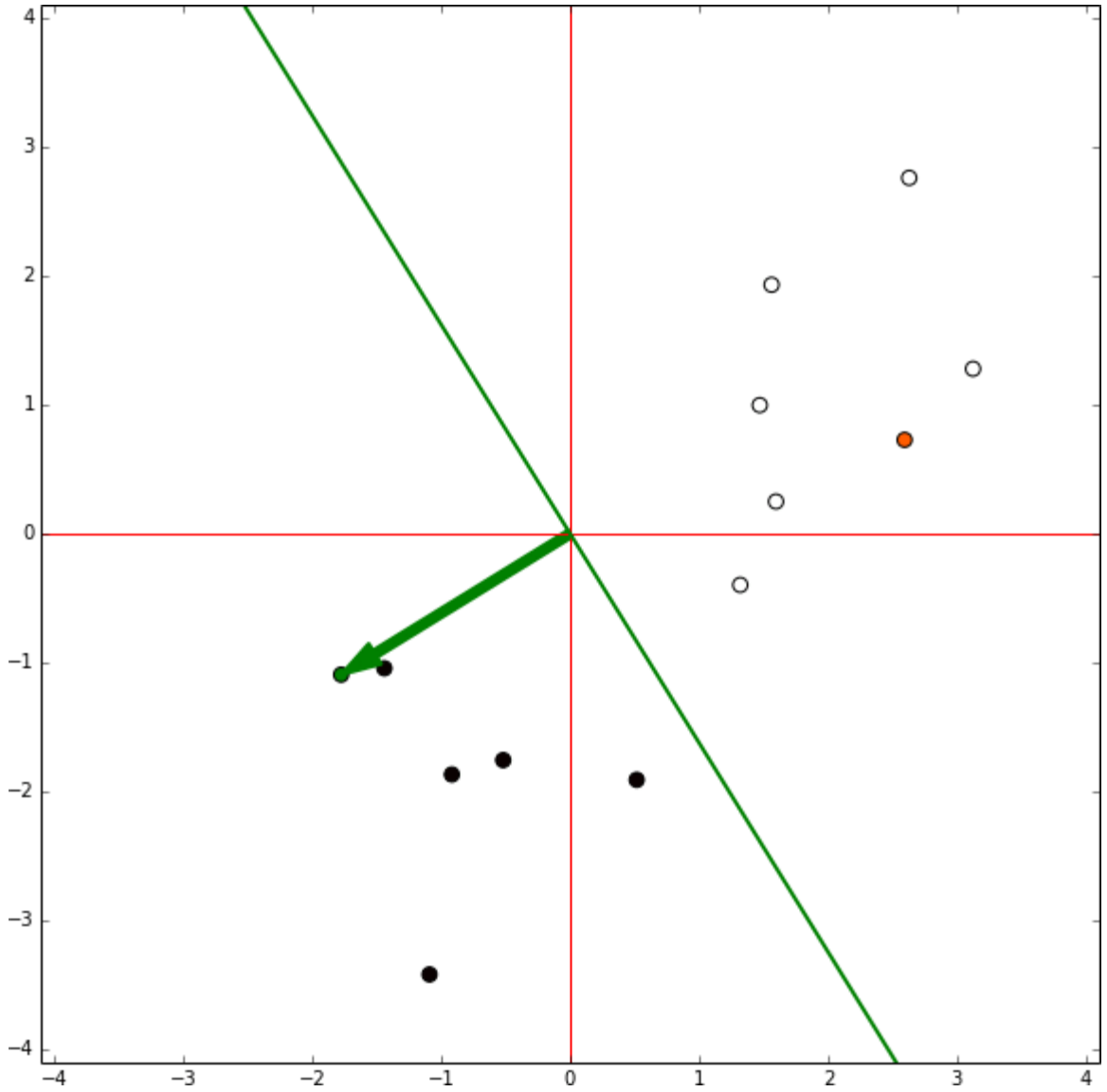
Repeat:

for each example $\boldsymbol{x_i}, y_i \in D$:

if $y_i \boldsymbol{w}^T \boldsymbol{x_i} < 0$: // if misclassification

$$\boldsymbol{w} = \boldsymbol{w} + \alpha y_i \boldsymbol{x_i}$$

Until no errors or max iterations



© Stefan Lee

The Perceptron: Limitations

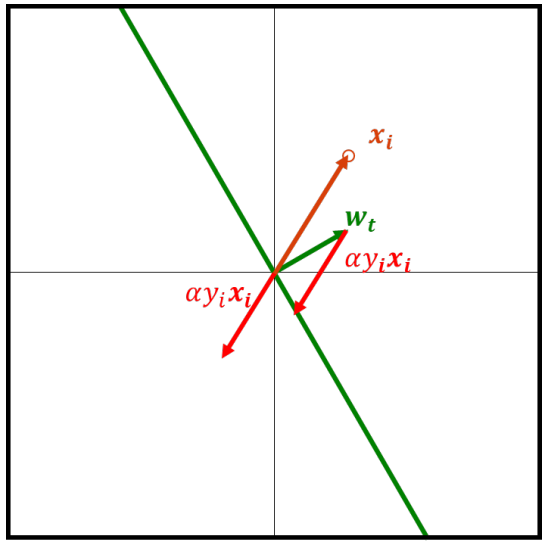
Perceptron Learning Algorithm

w = random(d)

While still errors or max iterations:

for each example $\boldsymbol{x_i}, y_i \in D$:

if $y\boldsymbol{w}^T\boldsymbol{x_i} < 0$: $\boldsymbol{w} = \boldsymbol{w} + \alpha y_i \boldsymbol{x_i}$



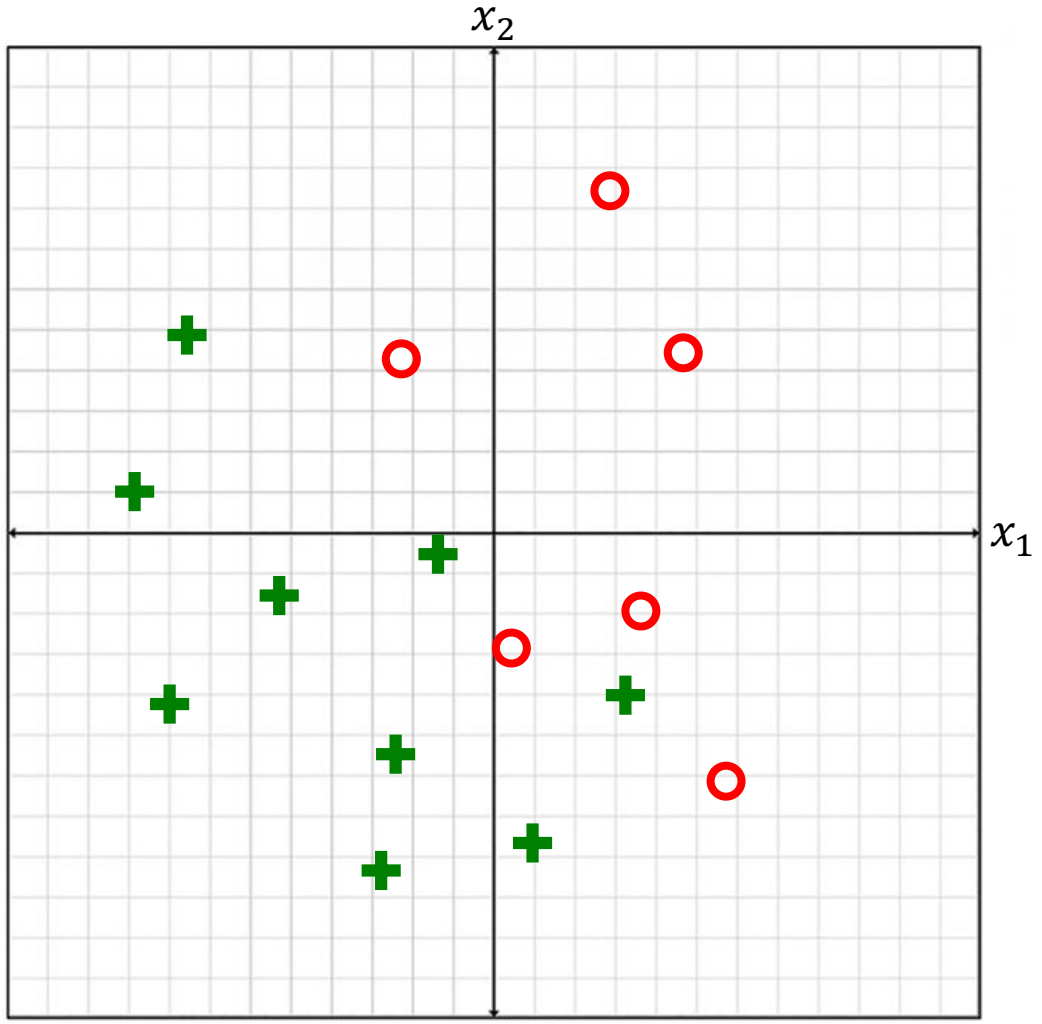
- Different solutions depending on initialization and order of visiting examples.
- Correcting for a misclassification could move the decision boundary so much that previously correct examples are now misclassified.
  - As such it must go over the training examples multiple times. Each time it goes through the whole training set, it is called an epoch.
- It will terminate if no update is made to $\boldsymbol{w}$ during one epoch

© Stefan Lee

The Perceptron: Limitations

- This algorithm is guaranteed to converge if data is linearly separable – i.e., it reaches a solution in finitely many steps.
  - Why and how many steps? (proof)

- For non-linearly separable cases (like on right), algorithm fails to converge and may be arbitrarily bad if terminated at some maximum number of updates.



© Stefan Lee

Naïve Bayes

© Stefan Lee

Reminders about (conditional) independence

Independence: If two random variables X and Y are independent, then:

$$P(X,Y) = P(X)P(Y), \qquad P(X|Y) = P(X), \qquad P(Y|X) = P(Y)$$

Conditional Independence: If two random variables X and Y are conditionally independent given Z, then:

$$P(X,Y|Z) = P(X|Z)P(Y|Z), \qquad P(X|Y,Z) = P(X|Z), \qquad P(Y|X,Z) = P(Y|Z)$$

© Stefan Lee

Reminders about (conditional) independence

Example: Define three random variables:

- **H:** A person's height
- **V:** How many words they know
- **A:** A person's age

def children:
Short people with small vocabularies.

Are height and vocabulary size independent?

No. Children tend to have lower vocabularies than adults (on average). So without knowing the person's age: $P(V, H) \neq P(V)P(H)$

However, if I tell you their age... its likely that these are independent. $\mathrm{P}(\mathrm{V}, \mathrm{H}|A) = P(V|A)P(H|A)$

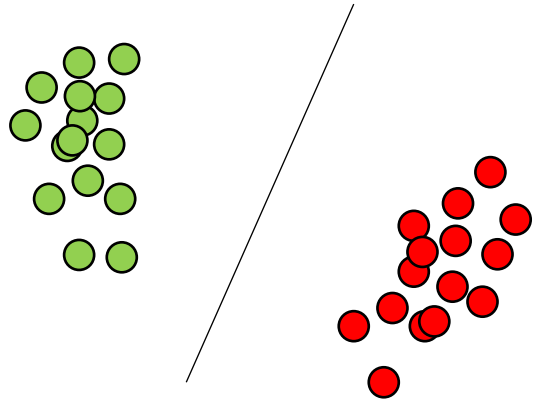
Events that are dependent in general, can be made independent given some other observation.

© Stefan Lee

Generative vs. Discriminative Models

Discriminative Classifiers:

- Learn $P(y|\boldsymbol{x})$ directly
- Logistic regression is one example
- *Nomenclature note -- people will also refer to algorithms that model no distribution as discriminative (such as kNN).*

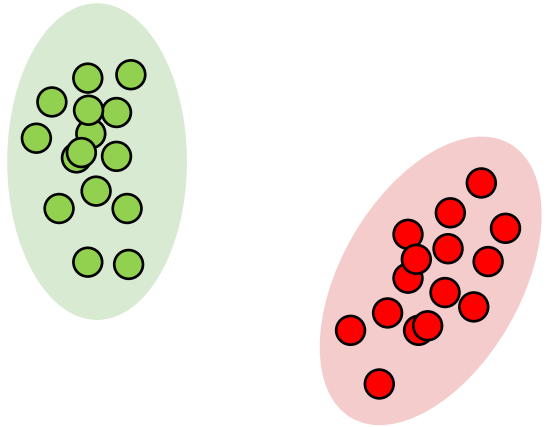


Generative Classifiers:

- Learn $P(\boldsymbol{x}|y)$ and $P(y)$
- Compute $P(y|\boldsymbol{x})$ using Bayes Rule

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})} = \frac{P(\boldsymbol{x}|y)P(y)}{\sum_y P(\boldsymbol{x}|y)P(y)}$$

- Naïve Bayes is one example (today)



Both classify according to argmax $P(y|\boldsymbol{x})$. Just learn and represent it differently.

© Stefan Lee

The Naïve Bayes Assumption:

Each feature is **conditionally independent** given the class label.

$$P(\boldsymbol{x}|y) = \prod_{i=1}^{d} P(x_i|y)$$

© Stefan Lee

How does this help?

$$P(y|\boldsymbol{x}) \propto P(\boldsymbol{x}|y)P(y)$$

Bayes Theorem

$$= P(y)\prod_{i=1}^{d} P(x_i|y)$$

Naïve Bayes Assumption

Can make predictions this way:

$$\underset{c=1,2,3,\ldots,k}{\operatorname{argmax}}\ P(y=c)\prod_{i=1}^{d} P(x_i|y=c)$$

© Stefan Lee

Parameter Cost of Learning P(x|y):

In general, if I measure **d** things in **x** with each having **m** options, P(**x**|y) will have $\mathbf{O}(\boldsymbol{cm^d})$ free parameters to learn for a **c** class problem. **Yikes.**

With the Naïve Bayes assumption, we only need $\mathbf{O}(\boldsymbol{cmd})$ for the $P(x_i|y)$ distributions and $\boldsymbol{c-1}$ for the class prior $P(y)$ for this setting.

© Stefan Lee

The Naïve Bayes Model Steps:

1. **Learn the conditional $P(x_i|y = c)$ for each feature x_i and class c** (training)
2. **Estimate $P(y = c)$ as a fraction of records with $y = c$ for each class c** (training)
3. **For a new example $x = [x_1, \dots, x_m]^T$, predict:** (testing)

$$\underset{c=1,2,3,\dots,k}{\operatorname{argmax}} P(y = c) \prod_{i=1}^{d} P(x_i|y = c)$$

© Stefan Lee

The zero-probability problem:

$$\underset{c=0,1}{\operatorname{argmax}} P(y = c) \prod_{i=1}^{d} P(w_i | y = c)$$

If any one of these terms is 0 for an instance, whole thing is 0.

Why might that happen?

What if a new email contains a word we never saw in the training emails?

$$P(w|spam) = 0 \; and \; P(w|not \; spam) = 0$$

© Stefan Lee

Laplace Smoothing for Binary Variables

For binary variable x_i, add a small prior to $p(x_i|y = c)$:

- Bernoulli

$$p(x_i|y = c) = \frac{(\#\ of\ times\ x_i\ is\ true\ and\ y = c) + 1}{(\#\ times\ y = c) + 2}$$

Taking our P(LowerUpper | CS434=easy) example:

$$p(lower|CS434 = easy) = \frac{(\#\ of fresh\&soph\ who\ think\ cs434\ is\ easy) + 1}{(\#fresh\&soph) + 2}$$

This is just adding a prior to the estimated conditional distributions.

Specifically, a Beta(1,1) prior.

© Stefan Lee

A Binary Example

X1	X2	Y
1	0	1
0	1	1
1	0	0

y	P(Y=y)
0	1/3
1	2/3

x	P(X1=x|Y=1)
0	½
1	½

x	P(X1=x|Y=0)
0	0
1	1

x	P(X2=x|Y=1)
0	½
1	½

x	P(X2=x|Y=0)
0	1
1	0

© Stefan Lee

Fit A Bernoulli Naïve Bayes Model To This Data

Train

x_1	x_2	x_3	x_4	y
1	0	1	1	1
1	0	0	1	1
0	0	0	1	1
1	1	1	1	2
0	1	1	0	0
1	1	0	1	1
0	1	0	1	1
1	1	1	1	2
0	1	1	0	0
1	1	1	0	0
1	1	1	0	2
0	1	1	1	2
0	1	1	0	2
1	0	1	1	2
0	1	1	0	2
0	0	1	1	1
1	1	1	0	0
1	1	1	0	0
0	0	1	0	2
1	0	1	1	2

Test

x_1	x_2	x_3	x_4	y
1	1	0	0	1
1	1	1	1	0
1	0	1	1	1
0	0	0	0	2
0	1	1	1	2
0	1	1	1	0

© Stefan Lee

Fit A Bernoulli Naïve Bayes Model To This Data

Train

x_1	x_2	x_3	x_4	y
1	0	1	1	1
1	0	0	1	1
0	0	0	1	1
1	1	1	1	2
0	1	1	0	0
1	1	0	1	1
0	1	0	1	1
1	1	1	1	2
0	1	1	0	0
1	1	1	0	0
1	1	1	0	2
0	1	1	1	2
0	1	1	0	2
1	0	1	1	2
0	1	1	0	2
0	0	1	1	1
1	1	1	0	0
1	1	1	0	0
0	0	1	0	2
1	0	1	1	2

Class Prior

y=0 Conditionals

y=1 Conditionals

y=2 Conditionals

© Stefan Lee

Fit A Bernoulli Naïve Bayes Model To This Data

Train

x_1	x_2	x_3	x_4	y
1	0	1	1	1
1	0	0	1	1
0	0	0	1	1
1	1	1	1	2
0	1	1	0	0
1	1	0	1	1
0	1	0	1	1
1	1	1	1	2
0	1	1	0	0
1	1	1	0	0
1	1	1	0	2
0	1	1	1	2
0	1	1	0	2
1	0	1	1	2
0	1	1	0	2
0	0	1	1	1
1	1	1	0	0
1	1	1	0	0
0	0	1	0	2
1	0	1	1	2

Class Prior

P(y=0) = 5/20 **P(y=1) = 6/20** **P(y=2) = 9/20**

y=0 Conditionals

y=1 Conditionals

y=2 Conditionals

© Stefan Lee

Fit A Bernoulli Naïve Bayes Model To This Data

Train

x_1	x_2	x_3	x_4	y
1	0	1	1	1
1	0	0	1	1
0	0	0	1	1
1	1	1	1	2
0	1	1	0	0
1	1	0	1	1
0	1	0	1	1
1	1	1	1	2
0	1	1	0	0
1	1	1	0	0
1	1	1	0	2
0	1	1	1	2
0	1	1	0	2
1	0	1	1	2
0	1	1	0	2
0	0	1	1	1
1	1	1	0	0
1	1	1	0	0
0	0	1	0	2
1	0	1	1	2

Class Prior

P(y=0) = 5/20 P(y=1) = 6/20 P(y=2) = 9/20

y=0 Conditionals

P(x1=1 | y=0) = 3/5

P(x2=1 | y=0) = 5/5

P(x3=1 | y=0) = 5/5

P(x4=1 | y=0) = 0/5

y=1 Conditionals

y=2 Conditionals

© Stefan Lee

Fit A Bernoulli Naïve Bayes Model To This Data

Train

x_1	x_2	x_3	x_4	y
1	0	1	1	1
1	0	0	1	1
0	0	0	1	1
1	1	1	1	2
0	1	1	0	0
1	1	0	1	1
0	1	0	1	1
1	1	1	1	2
0	1	1	0	0
1	1	1	0	0
1	1	1	0	2
0	1	1	1	2
0	1	1	0	2
1	0	1	1	2
0	1	1	0	2
0	0	1	1	1
1	1	1	0	0
1	1	1	0	0
0	0	1	0	2
1	0	1	1	2

Class Prior

P(y=0) = 5/20 P(y=1) = 6/20 P(y=2) = 9/20

y=0 Conditionals

P(x1=1 | y=0) = 3/5

P(x2=1 | y=0) = 5/5

P(x3=1 | y=0) = 5/5

P(x4=1 | y=0) = 0/5

y=1 Conditionals

P(x1=1 | y=1) = 3/6

P(x2=1 | y=1) = 2/6

P(x3=1 | y=1) = 2/6

P(x4=1 | y=1) = 6/6

y=2 Conditionals

© Stefan Lee

Fit A Bernoulli Naïve Bayes Model To This Data

Train

x_1	x_2	x_3	x_4	y
1	0	1	1	1
1	0	0	1	1
0	0	0	1	1
1	1	1	1	2
0	1	1	0	0
1	1	0	1	1
0	1	0	1	1
1	1	1	1	2
0	1	1	0	0
1	1	1	0	0
1	1	1	0	2
0	1	1	1	2
0	1	1	0	2
1	0	1	1	2
0	1	1	0	2
0	0	1	1	1
1	1	1	0	0
1	1	1	0	0
0	0	1	0	2
1	0	1	1	2

Class Prior

P(y=0) = 5/20 P(y=1) = 6/20 P(y=2) = 9/20

y=0 Conditionals

P(x1=1 | y=0) = 3/5

P(x2=1 | y=0) = 5/5

P(x3=1 | y=0) = 5/5

P(x4=1 | y=0) = 0/5

y=1 Conditionals

P(x1=1 | y=1) = 3/6

P(x2=1 | y=1) = 2/6

P(x3=1 | y=1) = 2/6

P(x4=1 | y=1) = 6/6

y=2 Conditionals

P(x1=1 | y=2) = 5/9

P(x2=1 | y=2) = 6/9

P(x3=1 | y=2) = 9/9

P(x4=1 | y=2) = 5/9

© Stefan Lee

Fit A Bernoulli Naïve Bayes Model To This Data

Class Prior

P(y=0) = 5/20	P(y=1) = 6/20	P(y=2) = 9/20

y=0 Conditionals	y=1 Conditionals	y=2 Conditionals
P(x1=1 | y=0) = 3/5	P(x1=1 | y=1) = 3/6	P(x1=1 | y=2) = 5/9
P(x2=1 | y=0) = 5/5	P(x2=1 | y=1) = 2/6	P(x2=1 | y=2) = 6/9
P(x3=1 | y=0) = 5/5	P(x3=1 | y=1) = 2/6	P(x3=1 | y=2) = 9/9
P(x4=1 | y=0) = 0/5	P(x4=1 | y=1) = 6/6	P(x4=1 | y=2) = 5/9

Test

x_1	x_2	x_3	x_4	y
1	1	0	0	1
1	1	1	1	0
1	0	1	1	1
0	0	0	0	2
0	1	1	1	2
0	1	1	1	0

$$P(y = 0 \mid [1\ 1\ 1\ 1]) \propto P(x_1 = 1|y = 0)P(x_2 = 1|y = 0)\ P(x_3 = 1|y = 0)\ P(x_4 = 1|y = 0)P(y = 0)$$
$$\propto \frac{3}{5} * \frac{5}{5} * \frac{5}{5} * \frac{0}{5} * \frac{5}{20} = 0$$

$$P(y = 1 \mid [1\ 1\ 1\ 1]) \propto P(x_1 = 1|y = 1)P(x_2 = 1|y = 1)\ P(x_3 = 1|y = 1)\ P(x_4 = 1|y = 1)P(y = 1)$$
$$\propto \frac{3}{6} * \frac{2}{6} * \frac{2}{6} * \frac{6}{6} * \frac{6}{20} = \frac{432}{25920} = 0.016$$

$$P(y = 2 \mid [1\ 1\ 1\ 1]) \propto P(x_1 = 1|y = 2)P(x_2 = 1|y = 2)\ P(x_3 = 1|y = 2)\ P(x_4 = 1|y = 2)P(y = 2)$$
$$\propto \frac{5}{9} * \frac{6}{9} * \frac{9}{9} * \frac{5}{9} * \frac{9}{20} = \frac{12150}{131220} = 0.092$$

© Stefan Lee

Fit A Bernoulli Naïve Bayes Model To This Data

Class Prior

P(y=0) = 5/20	P(y=1) = 6/20	P(y=2) = 9/20

y=0 Conditionals	y=1 Conditionals	y=2 Conditionals
P(x1=1 | y=0) = 3/5	P(x1=1 | y=1) = 3/6	P(x1=1 | y=2) = 5/9
P(x2=1 | y=0) = 5/5	P(x2=1 | y=1) = 2/6	P(x2=1 | y=2) = 6/9
P(x3=1 | y=0) = 5/5	P(x3=1 | y=1) = 2/6	P(x3=1 | y=2) = 9/9
P(x4=1 | y=0) = 0/5	P(x4=1 | y=1) = 6/6	P(x4=1 | y=2) = 5/9

Test

x_1	x_2	x_3	x_4	y
1	1	0	0	1
1	1	1	1	0
1	0	1	1	1
0	0	0	0	2
0	1	1	1	2
0	1	1	1	0

$$P(y=0 \mid [1\ 1\ 0\ 0]) \propto P(x_1=1|y=0)P(x_2=1|y=0)\ P(x_3=0|y=0)\ P(x_4=0|y=0)P(y=0)$$
$$\propto \frac{3}{5} * \frac{5}{5} * \frac{0}{5} * \frac{5}{5} * \frac{5}{20} = 0$$

$$P(y=1 \mid [1\ 1\ 0\ 0]) \propto P(x_1=1|y=1)P(x_2=1|y=1)\ P(x_3=0|y=1)\ P(x_4=0|y=1)P(y=1)$$
$$\propto \frac{3}{6} * \frac{2}{6} * \frac{4}{6} * \frac{0}{6} * \frac{6}{20} = 0$$

$$P(y=2 \mid [1\ 1\ 0\ 0]) \propto P(x_1=1|y=2)P(x_2=1|y=2)\ P(x_3=0|y=2)\ P(x_4=0|y=2)P(y=2)$$
$$\propto \frac{5}{9} * \frac{6}{9} * \frac{0}{9} * \frac{4}{9} * \frac{9}{20} = 0$$

© Stefan Lee

Support Vector Machines

© Stefan Lee

Comparing Hard and Soft Margin SVM

Hard-Margin SVM

Hard-Margin SVM Primal:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 \ \ \forall i$$

Hard-Margin SVM Dual:

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i}^{n}\sum_{j}^{n} \alpha_i\alpha_j y_i y_j \boldsymbol{x_i^T x_j}$$

$$s.t. \sum_{i=1}^{n} \alpha_i\, y_i = 0$$

$$0 \leq \alpha_i \ \ \forall i$$

Given optimal alphas – weight vector is:

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^*\, y_i \boldsymbol{x_i}$$

Soft-Margin SVM

Soft-Margin SVM Primal:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i} \xi_i$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 - \xi_i \ \ \forall i$$

$$\xi_i \geq 0$$

Soft-Margin SVM Dual:

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i}^{n}\sum_{j}^{n} \alpha_i\alpha_j y_i y_j \boldsymbol{x_i^T x_j}$$

$$s.t. \sum_{i=1}^{n} \alpha_i\, y_i = 0$$

$$0 \leq \alpha_i \leq C \ \ \forall i$$

Given optimal alphas – weight vector is:

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^*\, y_i \boldsymbol{x_i}$$

© Stefan Lee

Question Break!

What happens as if the **slack penalty** C goes to zero?

Maximize margin

Minimize violations of the margin

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \xi_i$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0$$

Allow some violations

A The soft-margin SVM reverts to the hard-margin SVM formulation.

B The margin gets infinitely big.

C Errors aren't penalized.

D Errors are very heavily penalized.

© Stefan Lee

Question Break!

What happens as if the **slack penalty** C goes to infinity?

Maximize margin

Minimize violations of the margin

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \xi_i$$

$$s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0$$

Allow some violations

A The soft-margin SVM reverts to the hard-margin SVM formulation.

B The margin gets infinitely big.

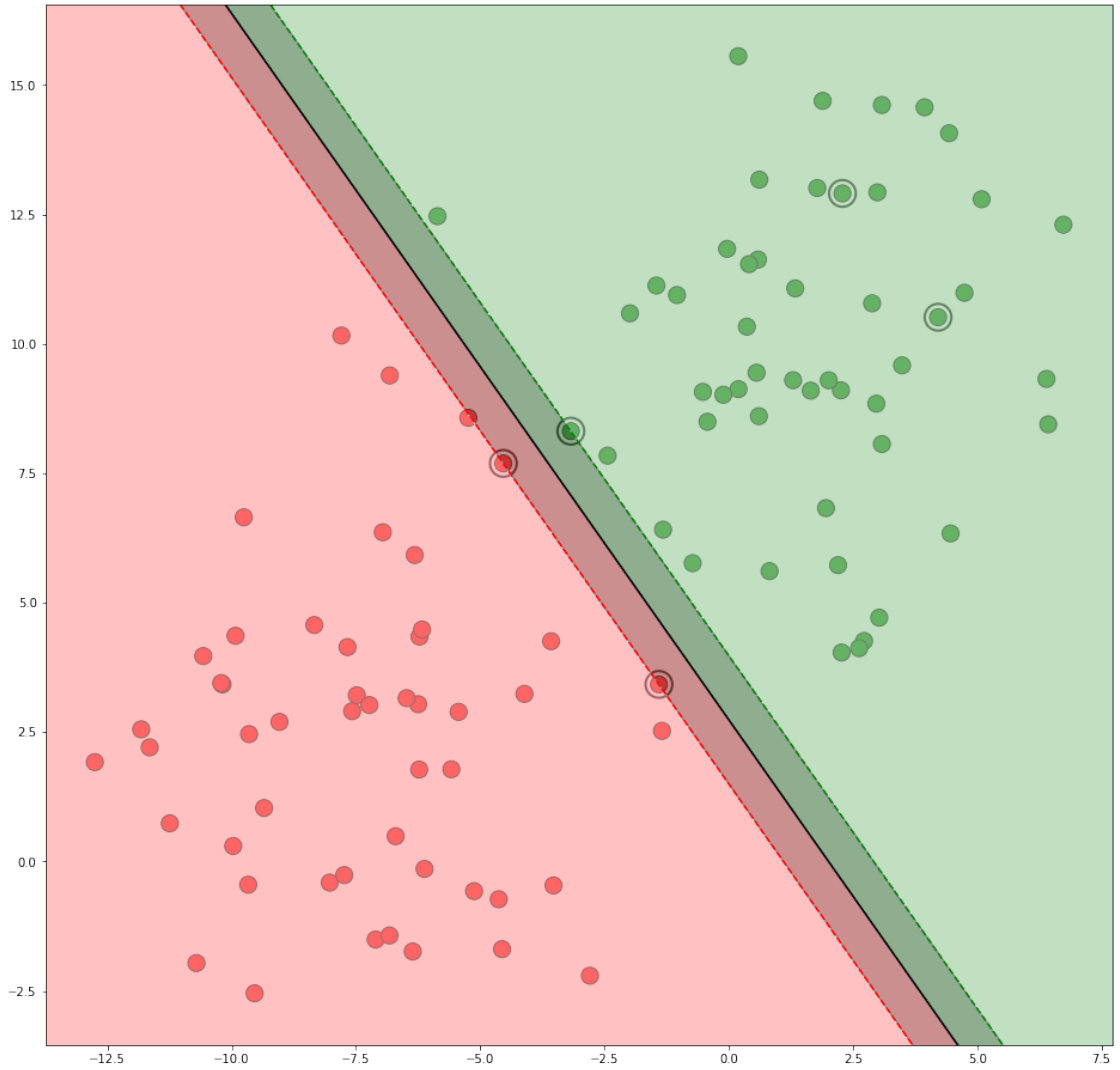
C Errors aren't penalized.

D Errors are very heavily penalized.

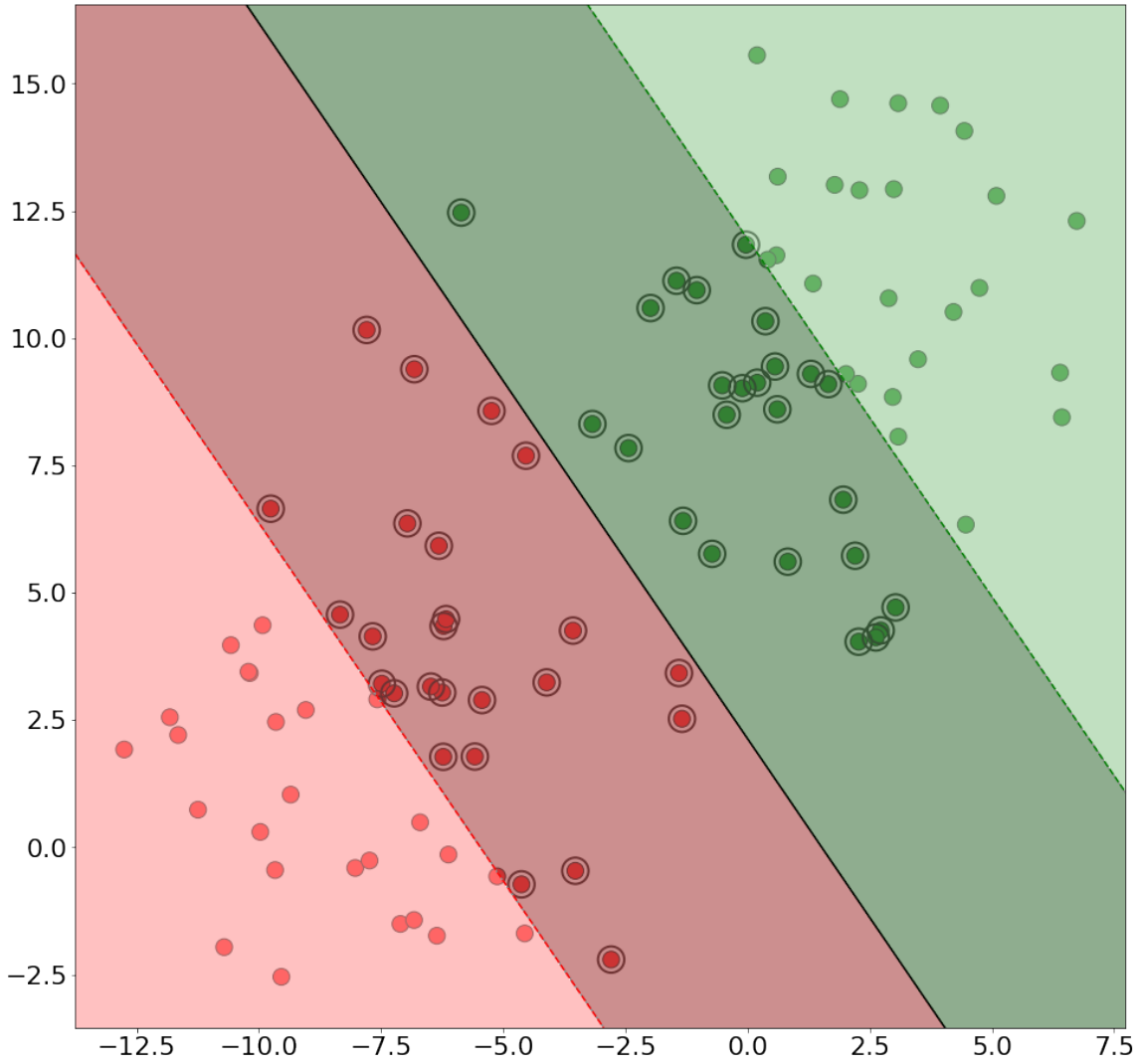
© Stefan Lee

SVM Hyperpameters

Hard Margin



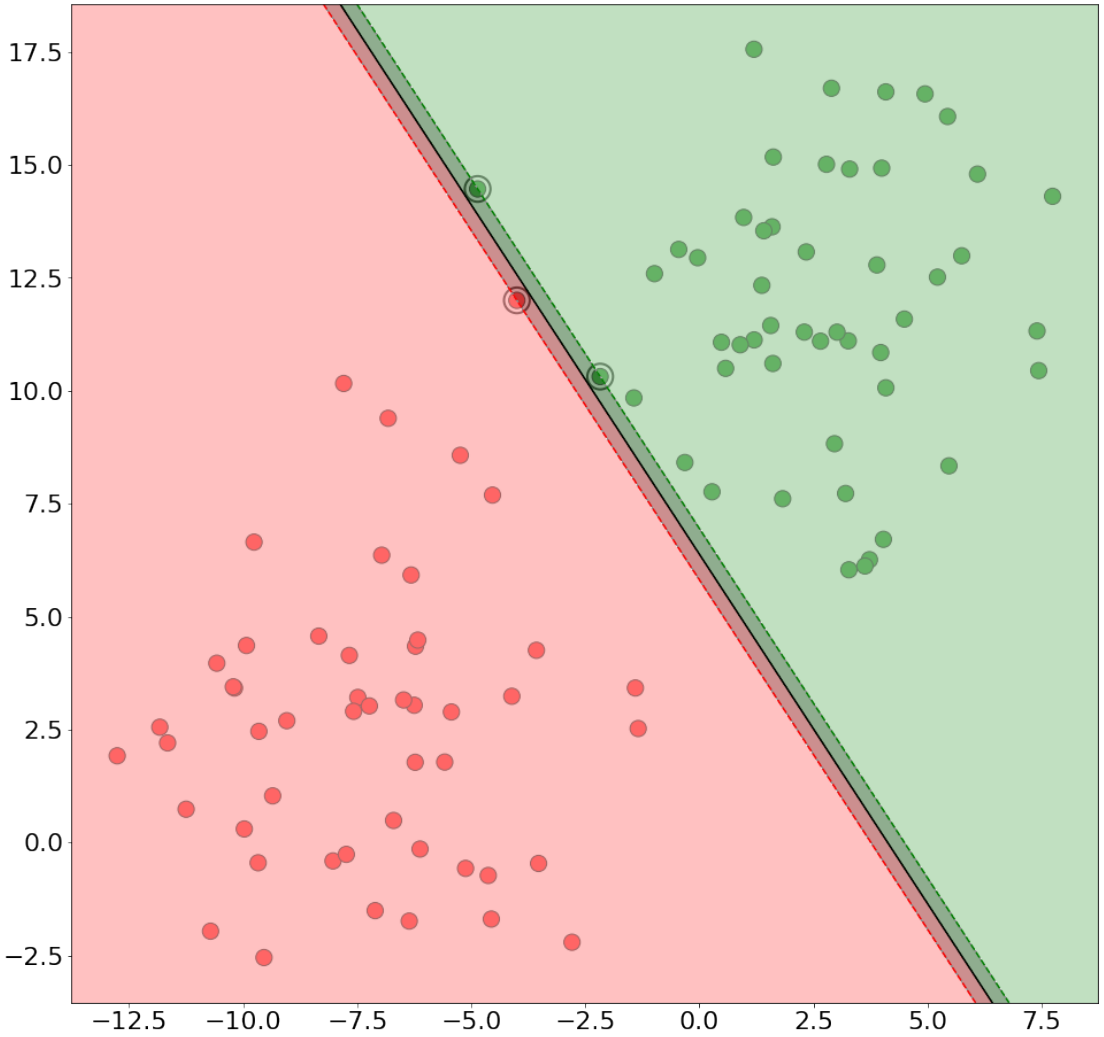
Soft Margin C=0.001



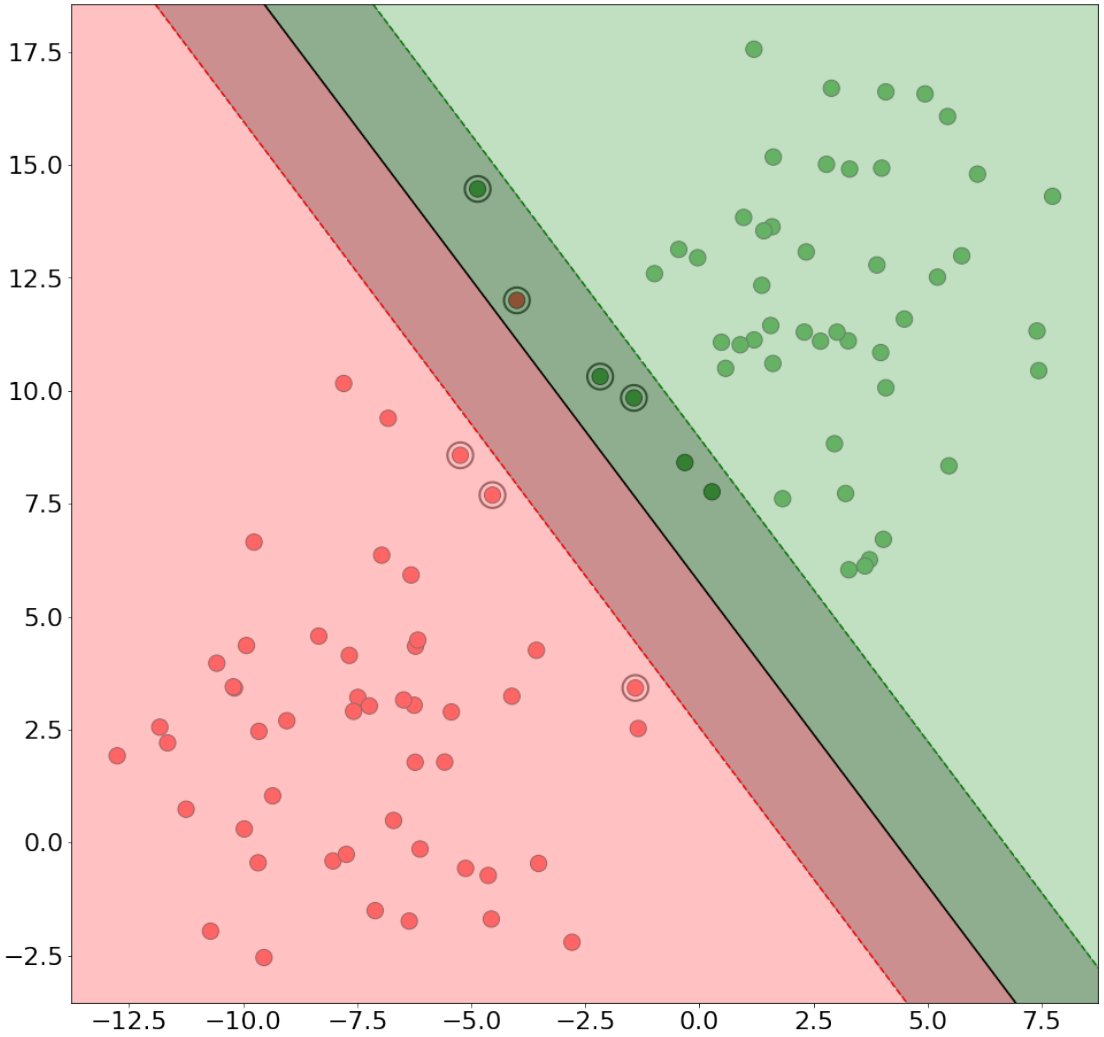
© Stefan Lee

SVM Hyperpameters

Hard Margin

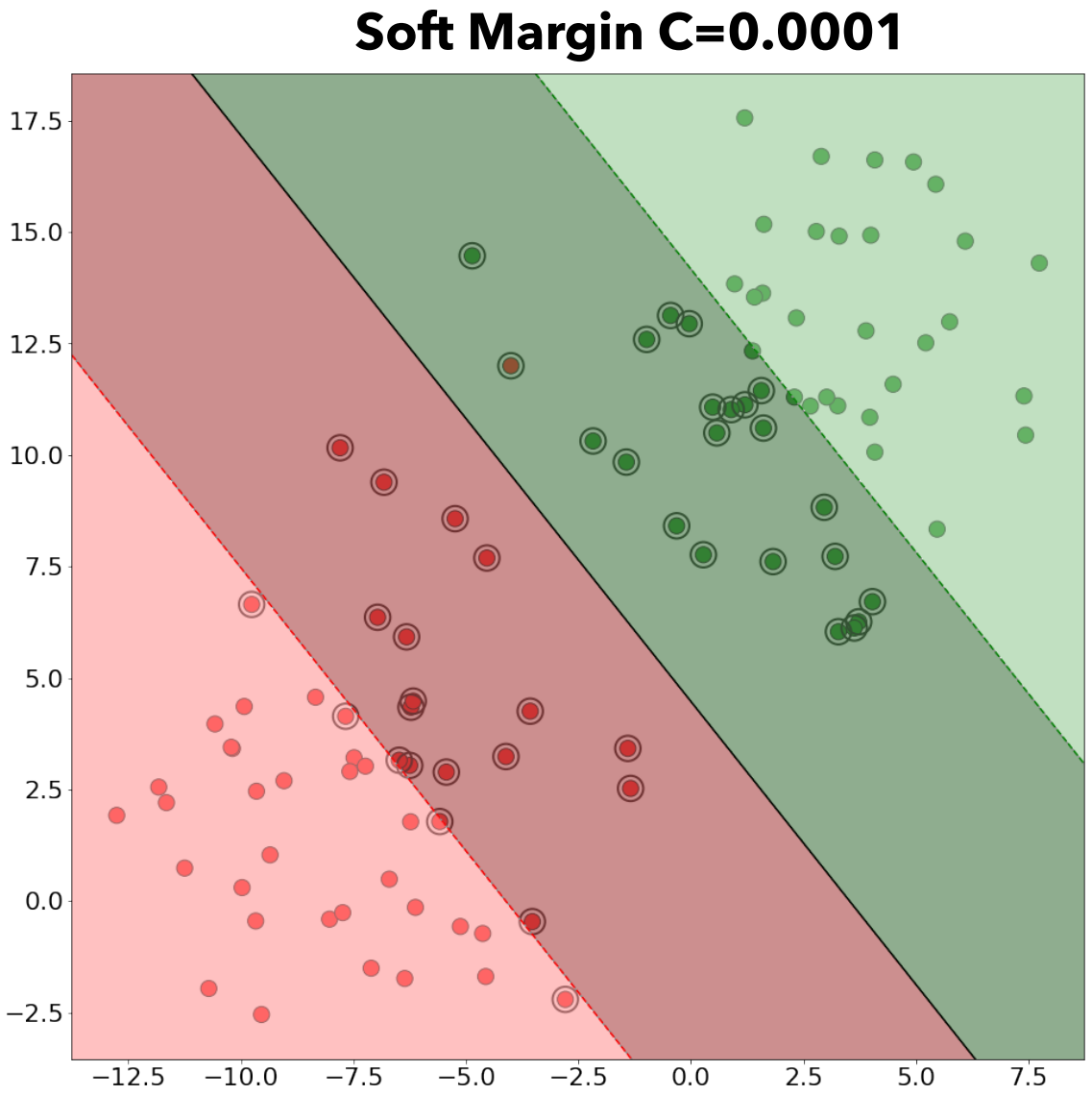


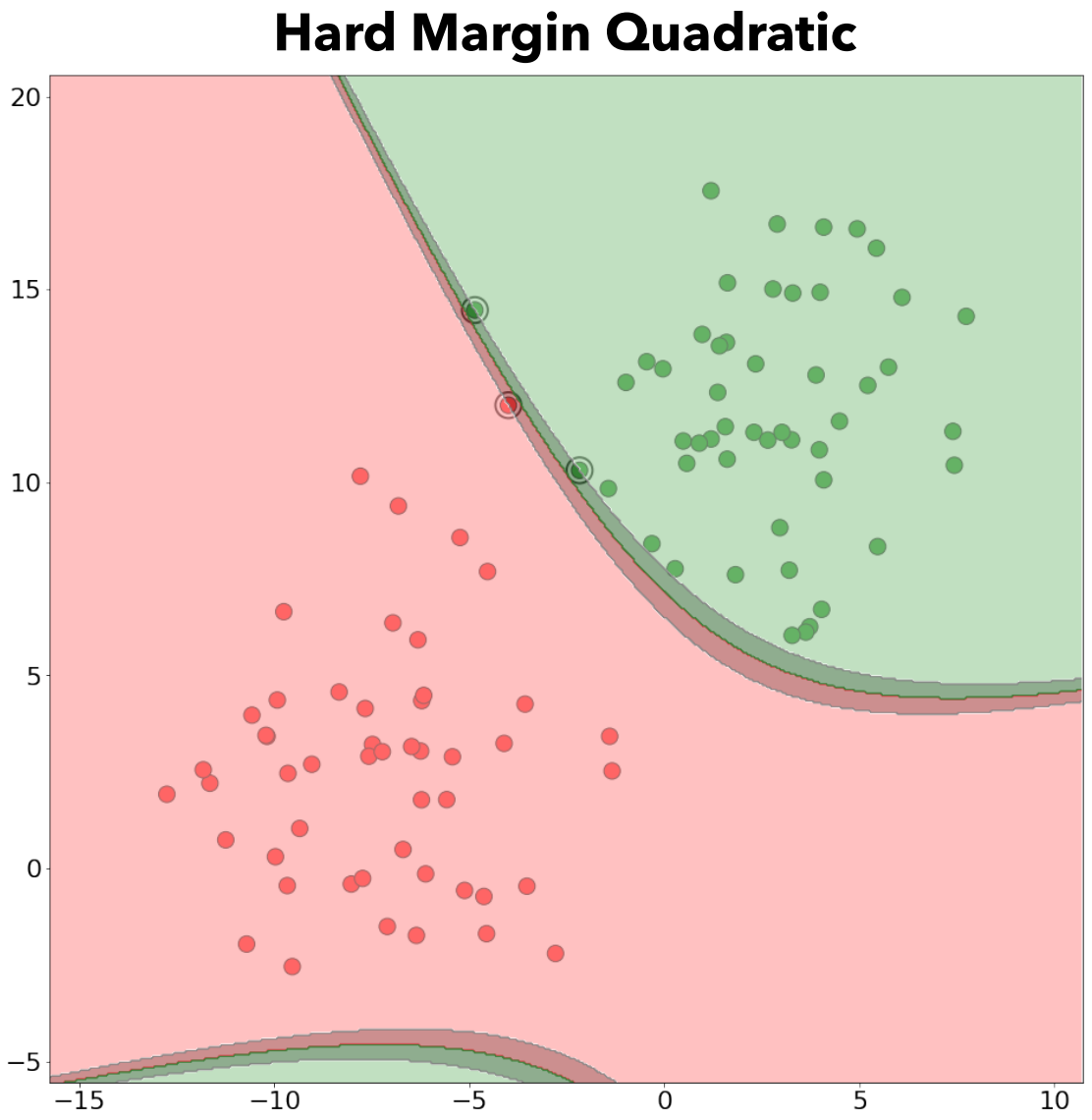
Soft Margin C=0.1



© Stefan Lee

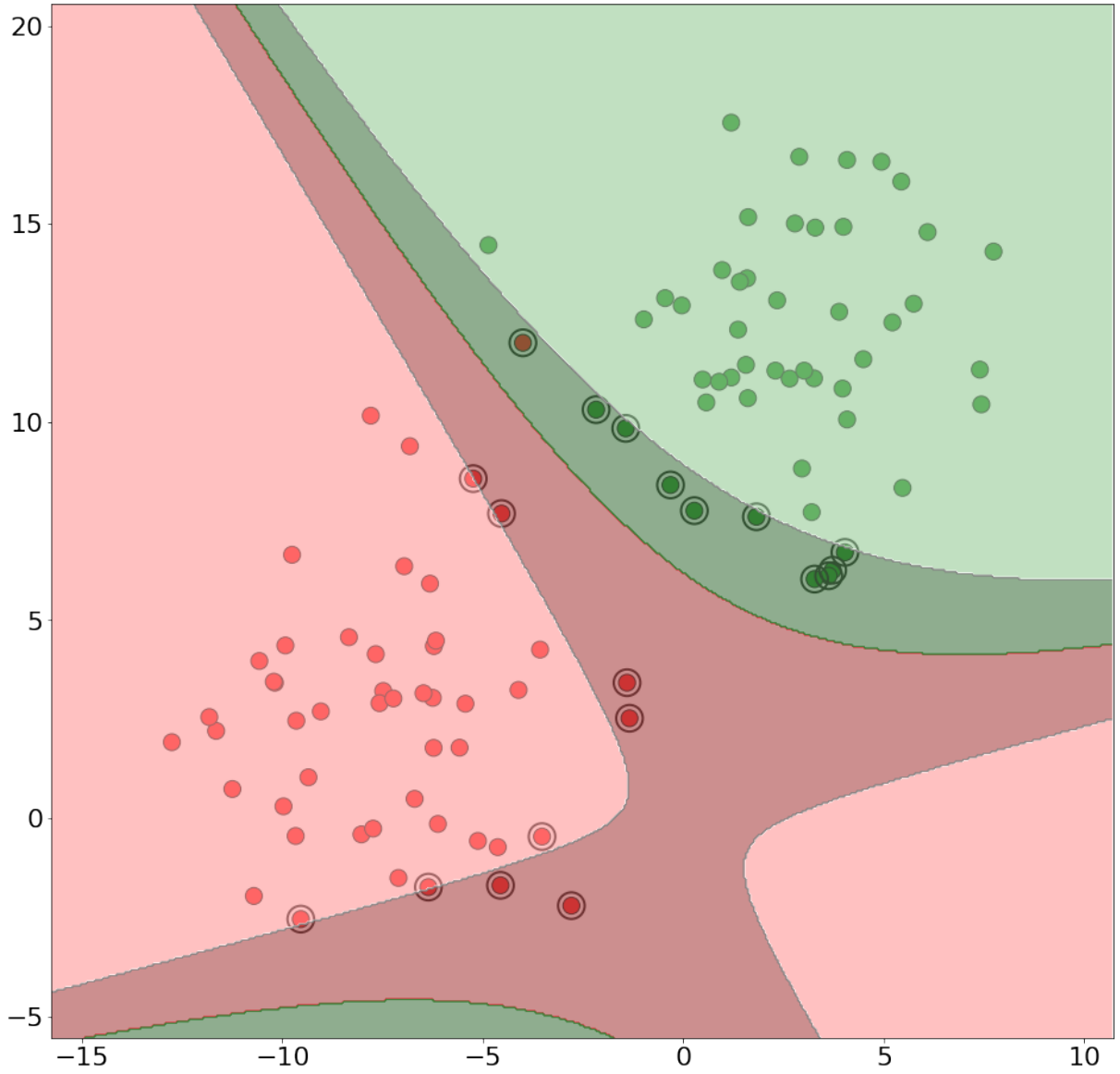
SVM Hyperpameters





© Stefan Lee

Soft Margin Quadratic C=0.0001



© Stefan Lee

Applying Basis functions to the SVM dual formulation and solution:

Soft-Margin SVM Dual:

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i}^{n} \sum_{j}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{\Phi}(\boldsymbol{x_i})^{\boldsymbol{T}} \boldsymbol{\Phi}(\boldsymbol{x_j})$$

$$s.t. \sum_{i=1}^{n} \alpha_i \, y_i = 0$$

$$0 \leq \alpha_i \leq C \quad \forall i$$

Seems like all that is going on here is computing similarity in some transformed feature space

Soft-Margin SVM Prediction:

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* \, y_i \boldsymbol{\Phi}(\boldsymbol{x_i}) \quad \Rightarrow \quad \boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{\Phi}(\boldsymbol{x}) + b = b + \sum_{i=1}^{n} \alpha_i^* \, y_i \boldsymbol{\Phi}(\boldsymbol{x_i})^{\boldsymbol{T}} \boldsymbol{\Phi}(\boldsymbol{x})$$

© Stefan Lee

Dot Product in Quadratic Feature Space

$$\Phi(\boldsymbol{a}) \cdot \Phi(\boldsymbol{b}) = \begin{bmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_d \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_d^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_d \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_2a_d \\ \vdots \\ \sqrt{2}a_{d-1}a_d \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_d \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_d^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_d \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_2b_d \\ \vdots \\ \sqrt{2}b_{d-1}b_d \end{bmatrix} \begin{matrix} \} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} 1 \\ + \\ \sum_{i=1}^{d} 2a_ib_i \\ + \\ \sum_{i=1}^{d} a_i^2b_i^2 \\ + \\ \sum_{i=1}^{d}\sum_{j=i+1}^{d} 2a_ia_jb_ib_j \end{matrix}$$

$$\Phi(\boldsymbol{a}) \cdot \Phi(\boldsymbol{b}) = 1 + 2\sum_{i=1}^{d} a_ib_i + \sum_{i=1}^{d} a_i^2b_i^2 + \sum_{i=1}^{d}\sum_{j=i+1}^{d} 2a_ia_jb_ib_j$$

Consider the following function: $(\boldsymbol{a} \cdot \boldsymbol{b} + 1)^2$

$$(\boldsymbol{a} \cdot \boldsymbol{b} + \mathbf{1})^2 = \mathbf{1} + 2\boldsymbol{a} \cdot \boldsymbol{b} + (\boldsymbol{a} \cdot \boldsymbol{b})^2$$

$$= 1 + 2\left(\sum_{i=1}^{d} a_ib_i\right) + \left(\sum_{i=1}^{d} a_ib_i\right)^2$$

$$= 1 + 2\sum_{i=1}^{d} a_ib_i + \sum_{i=1}^{d}\sum_{j=1}^{d} 2a_ia_jb_ib_j$$

$$= 1 + 2\sum_{i=1}^{d} a_ib_i + \sum_{i=1}^{d} a_i^2b_i^2 + \sum_{i=1}^{d}\sum_{j=i+1}^{d} 2a_ia_jb_ib_j$$

$$(\boldsymbol{a} \cdot \boldsymbol{b} + \mathbf{1})^2 = \Phi(\boldsymbol{a}) \cdot \Phi(\boldsymbol{b})$$

© Stefan Lee

Kernel Functions: The Kernel Trick

Definition: A function $\kappa(\boldsymbol{a}, \boldsymbol{b})$ is called a kernel function if $\kappa(\boldsymbol{a}, \boldsymbol{b}) = \Phi(\boldsymbol{a}) \cdot \Phi(\boldsymbol{b})$ for some mapping function Φ. Can be intuitively thought of as computing some similarity between **a** and **b**.

Not all mappings you might propose have a corresponding kernel function and not every kernel you might propose has a corresponding mapping.

However, for many popular mapping functions we might be interested in, we can find a corresponding kernel function that efficiently computes similarity.

Why we care? We can compute similarity in the high-dimensional mapping space without actually transforming the data into that space – gaining significant computational speedup. Call doing this the kernel trick.

© Stefan Lee

Closure Properties of Kernels

If κ_1 and κ_2 are kernel functions then the following are all also kernel functions:

- $\kappa(a, b) = \kappa_1(a, b) + \kappa_2(a, b)$
  - $\Phi = \Phi_1 + \Phi_2$
- $\kappa(a, b) = c\kappa_1(a, b)$, for c>0
  - $\Phi = \sqrt{a}\Phi_1$
- $\kappa(a, b) = \kappa_1(a, b)\kappa_2(a, b)$
  - $\Phi_{ij} = \Phi_{1i}\Phi_{2j}$ ← full cross product

© Stefan Lee

To apply to SVMs, we just replace the feature projected dot product with a kernel!

Soft-Margin SVM Dual:

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i}^{n} \sum_{j}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{\kappa}(\boldsymbol{x_i}, \boldsymbol{x_j})$$

$$s.t. \sum_{i=1}^{n} \alpha_i \, y_i = 0$$

$$0 \leq \alpha_i \leq C \quad \forall i$$

Just use the similarity from the kernel to do our learning / prediction.

Soft-Margin SVM Prediction:

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* \, y_i \boldsymbol{\Phi}(\boldsymbol{x_i}) \quad \Rightarrow \quad \boldsymbol{w^T \Phi}(\boldsymbol{x}) + b = b + \sum_{i=1}^{n} \alpha_i^* \, y_i \boldsymbol{\kappa}(\boldsymbol{x_i}, \boldsymbol{x_j})$$

© Stefan Lee

Evaluating Classifiers

© Stefan Lee

Two useful composite measures:

Recall: What fraction of positive does your model predict as positives:

$$Recall = \frac{\#TP}{\#TP + \#FN}$$

Precision: Of things your model predicts as positives, what fraction are correct?

$$Precision = \frac{\#TP}{\#TP + \#FP}$$

© Stefan Lee

Next time

Next Time: Midterm!

© Stefan Lee