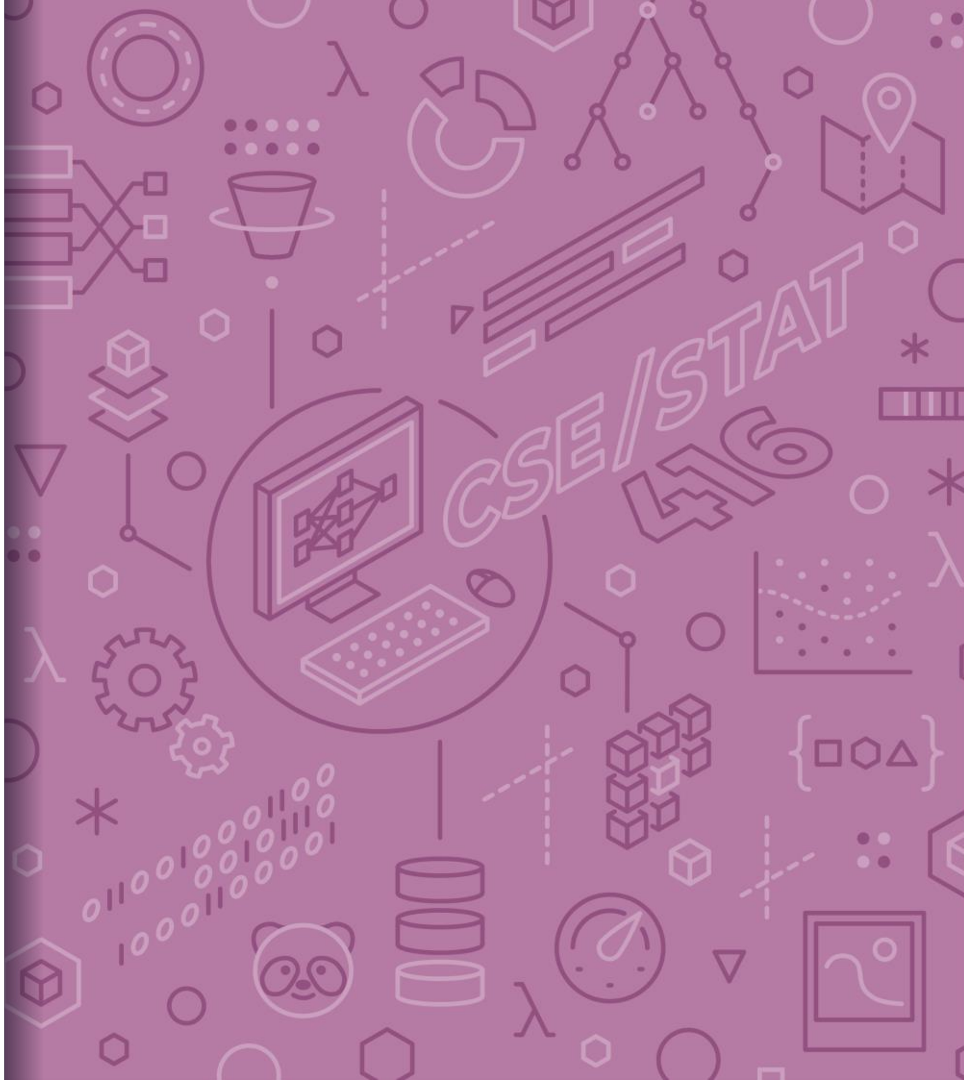
CSE/STAT 416

Logistic Regression

Amal Nanavati
University of Washington
July 13, 2022

Adapted from Hunter Schafer's slides



Administrivia

- Week 5: Other ML models for classification
- Week 6: Deep Learning
- Homework 2 due yesterday
  - Up to Thurs 11:59PM with late days
- HW3 Released today, due Tues 7/19 11:59PM
- Next week's homework, **HW4, will allow groups of up to 2 for the programming part**!
  - See Ed for information about group formation!
- LR4 Due Fri 11:59PM

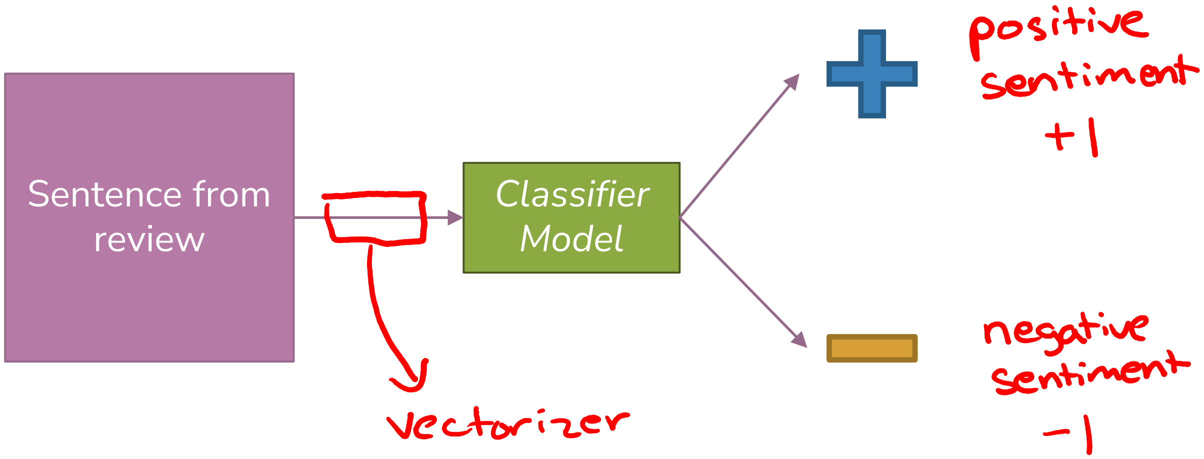
HW3 Walkthrough

Recap: Intro to Classification

Continuing from Lec5

Sentiment Classifier

In our example, we want to classify a restaurant review as positive or negative.



Input: x

Output: y
Predicted class

Converting Text to Numbers (Vectorizing):

Bag of Words

- **Idea**: One feature per word!

Example: ”Sushi was great, the food was awesome, but the service was terrible”

$h_1(x)$ $h_2(x)$ . . . . .

sushi	was	great	the	food	awesome	but	service	terrible
1	3	1	2	1	1	1	1	1

This **has** to be too simple, right?

- Stay tuned (today and ~~Wed~~ week 7) for issues that arise and how to address them ☺

Attempt 3: Linear Classifier

(Another View)

Idea: Only predict the sign of the output!

$$\text{Predicted } Sentiment = \hat{y} = sign(\, Score(x))$$

Linear Classifier

Input x: Sentence from review

- Compute $Score(x)$
- If $Score(x) > 0$: ← Threshold
  - $\hat{y} = +1$
- Else:
  - $\hat{y} = -1$

Earlier Example:

$Score(x) = 2$

$\hat{y} = +1$

Decision Boundary

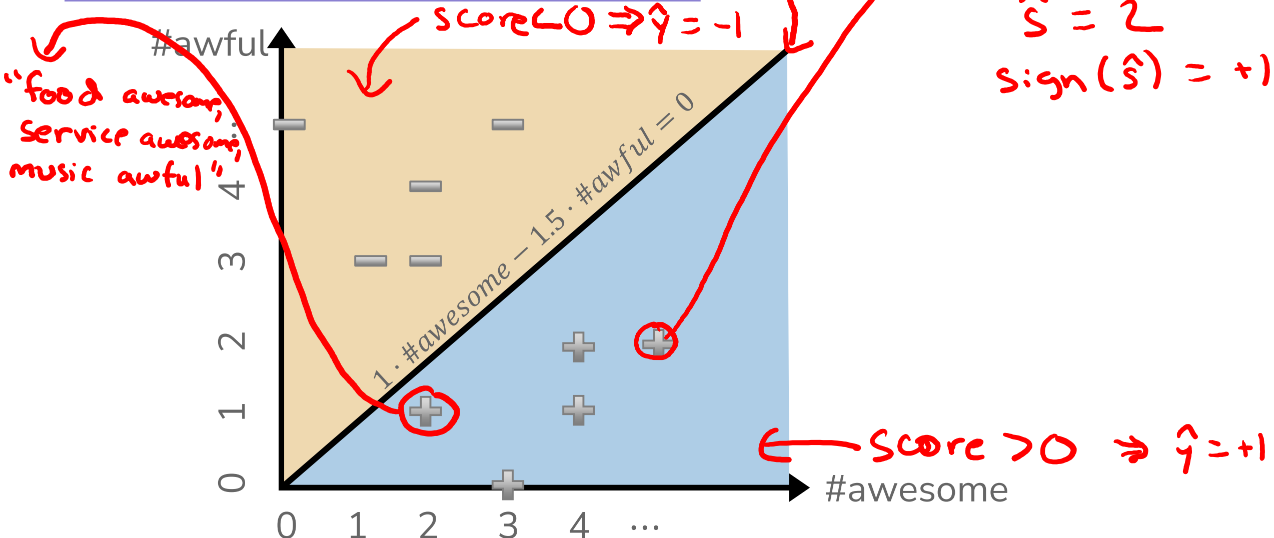
Consider if only two words had non-zero coefficients

Word	Coefficient	Weight
	w_0	0.0
awesome	w_1	1.0
awful	w_2	-1.5

$\hat{s} = 1 \cdot \#awesome - 1.5 \cdot \#awful$

defining this decision boundary

#awesome = 5
awful = 2
$\hat{s} = 2$
$\text{sign}(\hat{s}) = +1$



Classification Error

Ratio of examples where there was a mistaken prediction

What's a mistake?

- If the true label was positive ($y = +1$), but we predicted negative ($\hat{y} = -1$) → False Negative
- If the true label was negative ($y = -1$), but we predicted positive ($\hat{y} = +1$) → False Positive

Classification Error

$$\frac{\#\text{mistakes}}{\#\text{examples}} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i \neq \hat{y}_i\}}{n}$$

Classification Accuracy

$$\frac{\#\text{correct}}{\#\text{examples}} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = \hat{y}_i\}}{n} = 1 - \text{error}$$

Confusion Matrix

For binary classification, there are only two types of mistakes

- $\hat{y} = +1, \;\; y = -1$
- $\hat{y} = -1, \;\; y = +1$

Generally we make a **confusion matrix** to understand mistakes.

Complete the sentence: "my prediction was a..."

		Predicted Label	
		+	−
True Label	+	True Positive (TP)	False Negative (FN)
	−	False Positive (FP)	True Negative (TN)

Tip on remembering: complete the sentence "My prediction was a …"

Binary Classification Measures

Notation

- $C_{TP} = \#\mathrm{TP},\quad C_{\mathrm{FP}} = \#\mathrm{FP},\quad C_{\mathrm{TN}} = \#\mathrm{TN},\quad C_{\mathrm{FN}} = \#\mathrm{FN}$
- $N = C_{TP} + C_{FP} + C_{TN} + C_{FN}$
- $N_P = C_{TP} + C_{FN},\quad N_N = C_{FP} + C_{TN}$

Error Rate

$$\frac{C_{FP} + C_{FN}}{N}$$

Accuracy Rate

$$\frac{C_{TP} + C_{TN}}{N}$$

False Positive rate (FPR)

$$\frac{C_{FP}}{N_N}$$

False Negative Rate (FNR)

$$\frac{C_{FN}}{N_P}$$

True Positive Rate or Recall

$$\frac{C_{TP}}{N_P}$$

Precision

$$\frac{C_{TP}}{C_{TP} + C_{FP}}$$

F1-Score

$$2\,\frac{Precision \cdot Recall}{Precison + Recall}$$

See more!

Precision & Recall

Two particularly important metrics in binary classification are:

Precision: Of the ones I predicted positive, how many of them were actually positive?

- How precise is my model in its predictions?
- $TP / (\underbrace{TP + FP}_{\text{num you predicted to be positive}})$

Recall: Of all the things that are truly positive, how many of them did I correctly predict as positive?

- How good is your model at recalling the patterns in the training data?
- $TP / (\underbrace{TP + FN}_{\text{num datapoints that are actually positive}})$

Precision

What fraction of the examples I predicted positive were correct?

Sentences predicted to be positive: actual labels

$\hat{y}_i = +1$

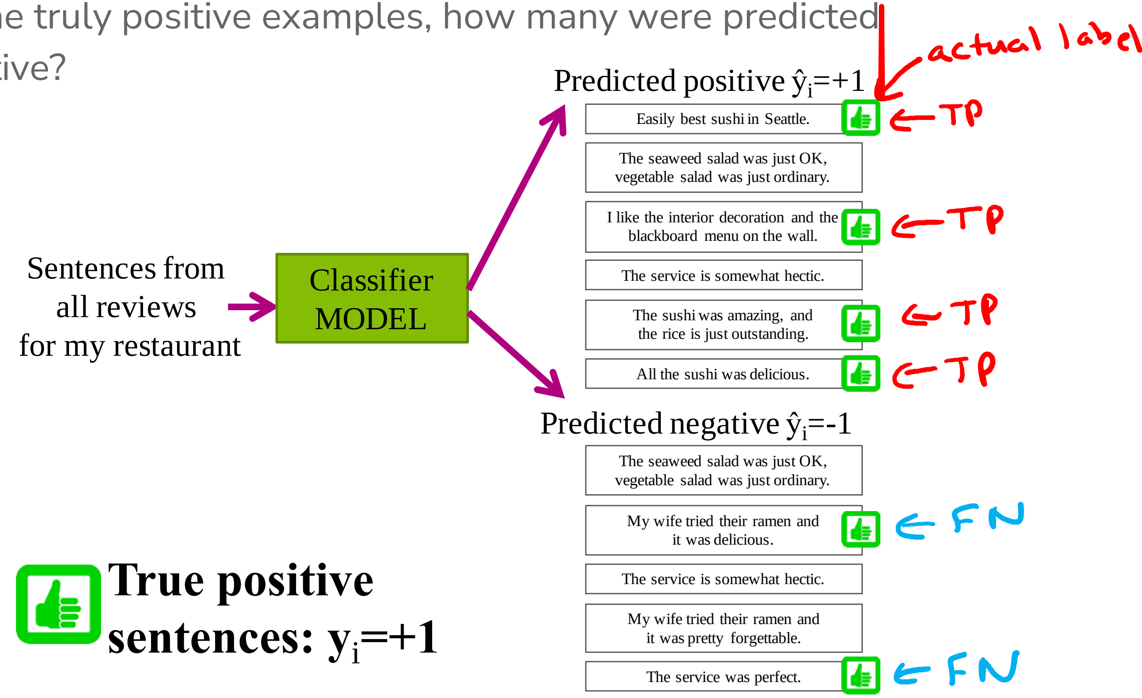
TP	Easily best sushi in Seattle.	✓
FP	The seaweed salad was just OK, vegetable salad was just ordinary.	✗
TP	I like the interior decoration and the blackboard menu on the wall.	✓
FP	The service is somewhat hectic.	✗
TP	The sushi was amazing, and the rice is just outstanding.	✓
TP	All the sushi was delicious.	✓

Only 4 out of 6 sentences predicted to be positive are actually positive

$$precision = \frac{C_{TP}}{C_{TP} + C_{FP}} = \frac{4}{4+2} = \frac{4}{6}$$

Recall

Of the truly positive examples, how many were predicted positive?



$$recall = \frac{C_{TP}}{N_P} = \frac{C_{TP}}{C_{TP} + C_{FN}} = \frac{4}{4+2} = \frac{4}{6}$$

Precision & Recall

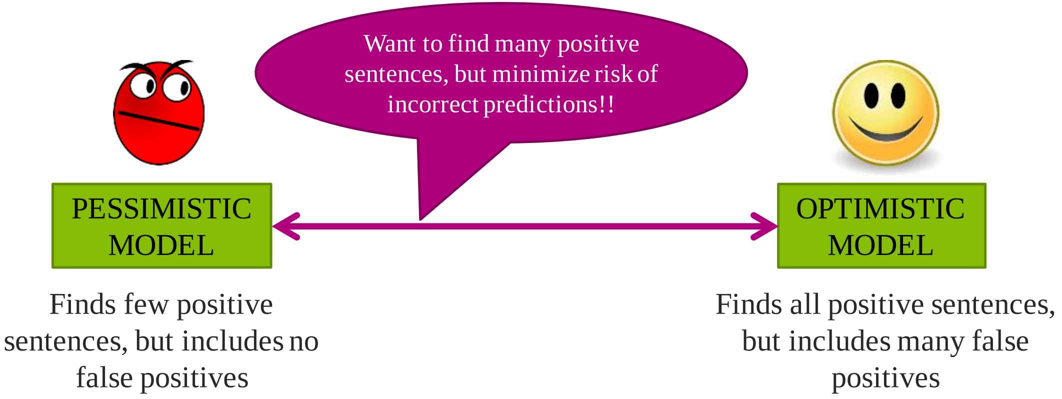
There is a tradeoff between precision and recall!

An optimistic model will predict almost everything as positive 

- High recall, low precision

A pessimistic model will predict almost everything as negative 

- High precision, low recall



Multiclass Confusion Matrix

Consider predicting (*Healthy, Cold, Flu*)

people who were healthy & predicted healthy

of people who were healthy but predicted to have flu

Predicted Label

True Label	Healthy	Cold	Flu
Healthy	60	8	2
Cold	4	12	4
Flu	0	2	8

correct

Poll Everywhere

Think

1 min

pollev.com/cs416

Suppose we trained a classifier and computed its confusion matrix on the training dataset. **Is there a class imbalance in the dataset and if so, which class has the highest representation?**

Predicted Label

True Label	Pupper	Doggo	Woofer
Pupper	2	27	4
Doggo	4	25	4
Woofer	1	30	2

Poll Everywhere

Pair

~~Think~~

2 min

pollev.com/cs416

Suppose we trained a classifier and computed its confusion matrix on the training dataset. **Is there a class imbalance in the dataset and if so, which class has the highest representation?**

Think:
73% doggo
27% no imbalance

Pair:
55% doggo
45% no imbalance

Predicted Label

True Label	Pupper	Doggo	Woofer	
Pupper	2	27	4	33
Doggo	4	25	4	33
Woofer	1	30	2	33
Total # model predicted to be that class	7	82	10	

Total # in dataset with that label

Logistic Regression

Can we use MSE for classification task?

$w_1 \approx 0$ $\hat{y} = \text{sign}(\text{Score}(w_1(\#\text{awesome}) - 1.5(\#\text{awful})))$

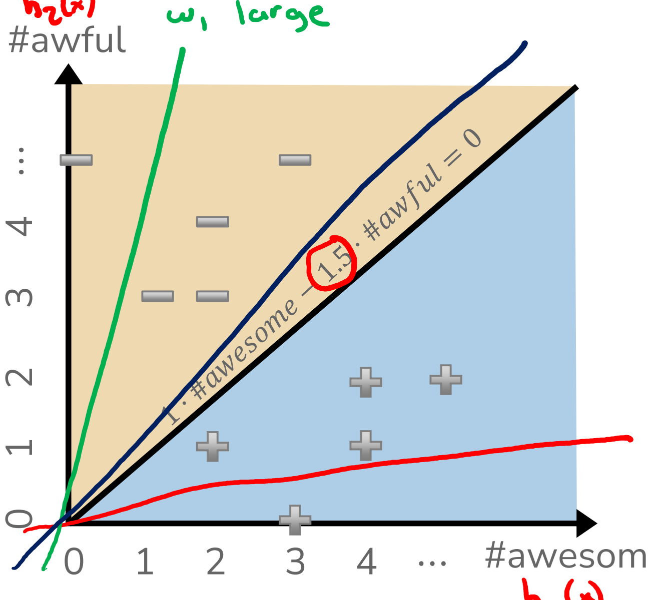
One idea is to just model the processing of finding $\hat{w}$ based on what we discussed in linear regression using MSE

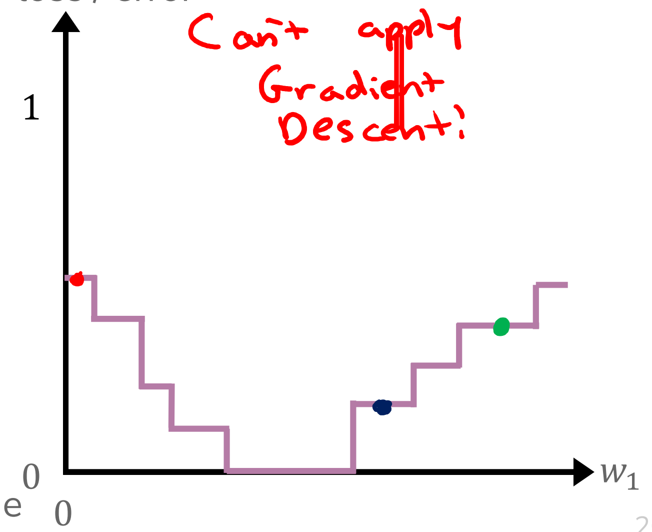
$\hat{y}_i = \text{sign}(\text{Score}(x_i))$

$$\hat{w} = \underset{w}{\text{argmin}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{y_i \neq \hat{y}_i\}^2}_{\text{loss/error}}$$

Will this work?

Assume $h_1(x) = \#awesome$ so w_1 is its coefficient and w_2 is fixed.





Quality Metric for Classification

The MSE loss function doesn't work because of different reasons:

- The outputs are discrete values with no ordered nature, so we need a different way to frame how close a prediction is to a certain correct category
- The MSE loss function for classification task is not continuous, differentiable or convex, so we can't use optimization algorithm like Gradient Descent to find an optimal set of weights

Note: Convexity is an important concept in Machine Learning. By minimizing error, we want to find where that global minimum is, and that's ideal in a convex function.

Let's frame this problem in term of probabilities instead.

Probabilities

$P(y \mid x) = \begin{cases} P(y=+1 \mid x) & \text{if } y = 1 \\ P(y=-1 \mid x) & \text{if } y = -1 \end{cases}$

$P(y=1 \mid x)$ probability that the true label is positive for x

Assume that there is some randomness in the world, and instead will try to model the probability of a positive/negative label.

Examples: $P(y=+1 \mid x) + P(y=-1 \mid x) = 1$

"The sushi & everything else were awesome!"

- Definite positive (+1)
- $P\left(y = +1 \mid x = \text{"The sushi \& everything else were awesome!"}\right) = 0.99$

"The sushi was alright, the service was OK"

- Not as sure
- $P\left(y = -1 \mid x = \text{"The sushi alright, the service was okay!"}\right) = 0.5$

Use probability as the measurement of certainty

$$P(y|x)$$

Probability Classifier

Idea: Estimate probabilities $\hat{P}(y|x)$ and use those for prediction

Probability Classifier

Input x: Sentence from review

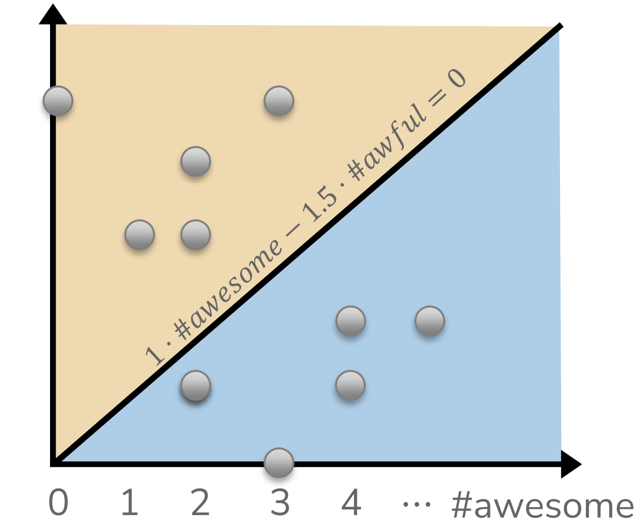
- Estimate class probability $\hat{P}(y = +1|x)$
- If $\hat{P}(y = +1|x) > 0.5$: ⟵ threshold
  - $\hat{y} = +1$
- Else:
  - $\hat{y} = -1$

Notes:

- Estimating the probability improves **interpretability.**
  - Unclear how much better a score of 5 is from a score of 3. Clear how much better a probability of 0.75 is than a probability of 0.5

Connecting Score & Probability

Idea: Let's try to relate the value of $Score(x)$ to $\hat{P}(y = +1|x)$



What if $Score(x)$ is positive?

If $Score(x) > 0$, then we want $\hat{P}(y=+1|x) > 0.5$

What if $Score(x)$ is negative?

If $Score(x) < 0$, then we want $\hat{P}(y=+1|x) < 0.5$

What if $Score(x)$ is 0?

we want $\hat{P}(y=+1|x) = 0.5$

Connecting Score & Probability

$$Score(x_i) = w^T h(x_i)$$

$-\infty$ | $\hat{y}_i = -1$ | 0 | $\hat{y}_i = +1$ | ∞

Very sure
$\hat{y}_i = -1$

Not sure if
$\hat{y}_i = -1 \; or \; \hat{y}_i = +1$

Very sure
$\hat{y}_i = +1$

$\hat{P}(y_i = +1 | x_i) = 0$ $\quad$ $\hat{P}(y_i = +1 | x_i) = 0.5$ $\quad$ $\hat{P}(y_i = +1 | x_i) = 1$

0 $\quad$ $\hat{P}(y = +1 \,| x)$ $\quad$ 1

Logistic Function

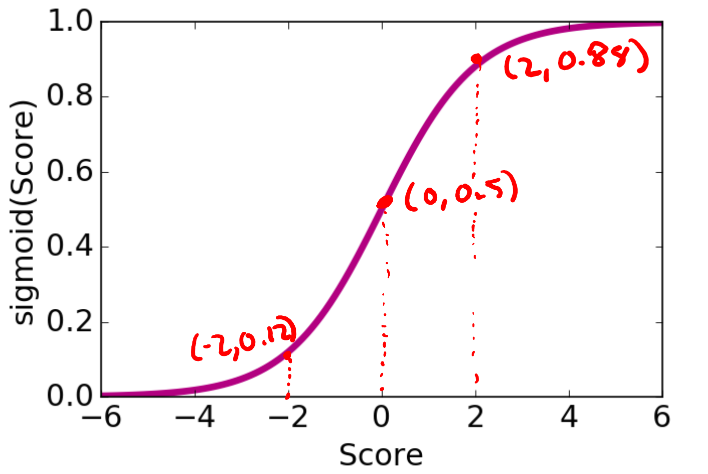
$\in (-\infty, \infty)$

Want: a function that takes numbers $\overbrace{\text{arbitrarily large/small}}^{\in(-\infty,\infty)}$ and maps them between 0 and 1.

Nice property:
$sigmoid(-x) = 1 - sigmoid(x)$

$$sigmoid(Score(x)) = \frac{1}{1 + e^{-Score(x)}}$$

$Score(x)$	$sigmoid(Score(x))$
$-\infty$	0
-2	0.12
0	0.5
2	0.88
∞	1.0



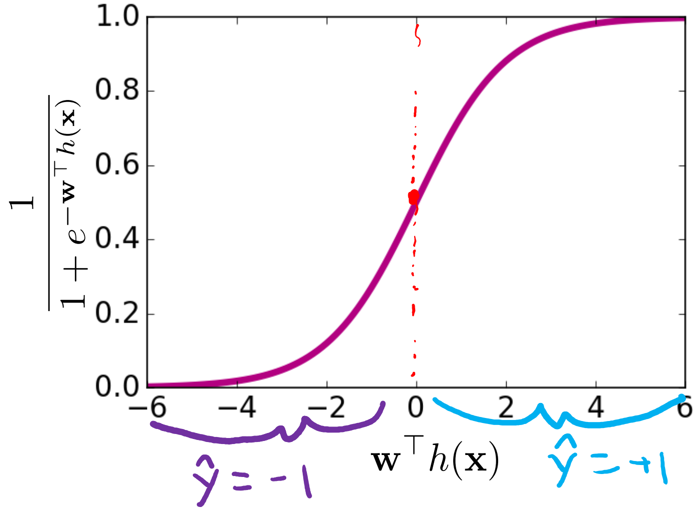
Logistic Regression Model

$$P(y_i = +1|x_i, w) = sigmoid\big(Score(x_i)\big) = \frac{1}{1 + e^{-w^T h(x_i)}}$$

Logistic Regression Classifier

Input x: Sentence from review

- Estimate class probability $\hat{P}(y = +1|x, \hat{w}) = sigmoid(\hat{w}^T h(x_i))$
- If $\hat{P}(y = +1|x, \hat{w}) > 0.5$:
  - $\hat{y} = +1$
- Else:
  - $\hat{y} = -1$



$P(y=+1 \mid x, w) = \text{sigmoid}(\text{score}(x))$

Poll Everywhere

Think

1 min

What would the Logistic Regression model predict for $P(y = -1 \mid x, w)$?

- "Sushi was great, the food was awesome, but the service was terrible"

a) ≈ 0

b) $sigmoid(-2) \approx 0.12$

c) ≈ 0.5

d) $sigmoid(2) \approx 0.88$

e) ≈ 1

Word	Weight
sushi	0
was	0
great	1
the	0
food	0
awesome	2
but	0
service	0
terrible	-1

$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$	$h_5(x)$	$h_6(x)$	$h_7(x)$	$h_8(x)$	$h_9(x)$
sushi	**was**	**great**	**the**	**food**	**awesome**	**but**	**service**	**terrible**
1	3	1	2	1	1	1	1	1

Poll Everywhere

Group

2 min

$P(y=+1 \mid x, w) = \text{sigmoid}(\text{score}(x))$

What would the Logistic Regression model predict for $P(y = -1 \mid x, w)$?

- "Sushi was great, the food was awesome, but the service was terrible"

$\text{Score} = w^T h(x)$

$= 1 \cdot 1 + 2 \cdot 1 + -1 \cdot 1$

$= 2$

a) ≈ 0

b) $sigmoid(-2) \approx 0.12$

c) ≈ 0.5

d) $sigmoid(2) \approx 0.88$

e) ≈ 1

Word	Weight
sushi	0
was	0
great	1
the	0
food	0
awesome	2
but	0
service	0
terrible	-1

$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$	$h_5(x)$	$h_6(x)$	$h_7(x)$	$h_8(x)$	$h_9(x)$
sushi	**was**	**great**	**the**	**food**	**awesome**	**but**	**service**	**terrible**
1	3	1	2	1	1	1	1	1

Poll Everywhere

Group

2 min

$P(y=+1\,|\,x,w) = \text{sigmoid}(\text{score}(x))$

What would the Logistic Regression model predict for $P(\underline{y = -1}\,|x, w)$?

- "Sushi was great, the food was awesome, but the service was terrible"

$\text{Score} = w^T h(x)$
$= 1\cdot 1 + 2\cdot 1 + -1\cdot 1$
$= 2$

a) ≈ 0

✓ b) $sigmoid(-2) \approx 0.12$

c) ≈ 0.5

d) $sigmoid(2) \approx 0.88$

e) ≈ 1

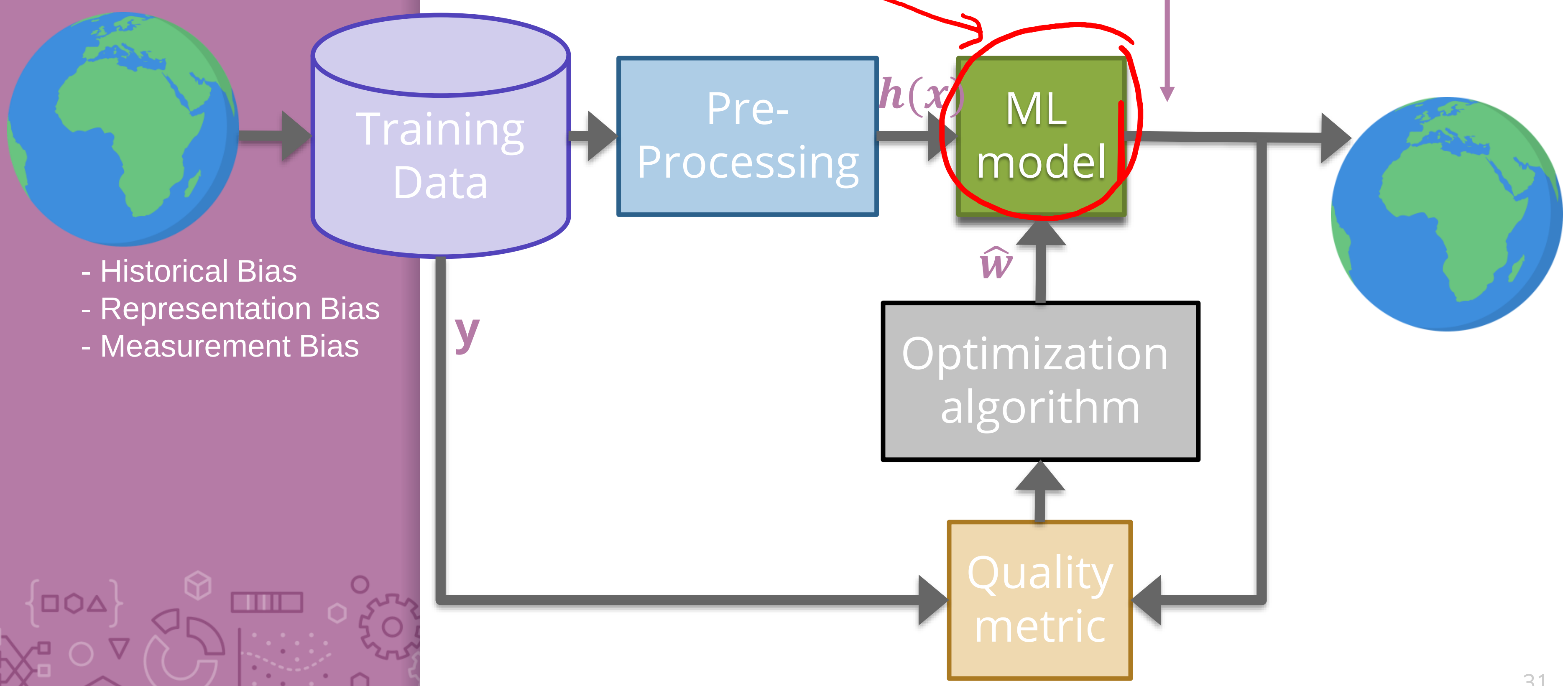
Word	Weight
sushi	0
was	0
great	1
the	0
food	0
awesome	2
but	0
service	0
terrible	-1

$P(y=+1\,|\,x,w) = \text{sigmoid}(2)$

$P(y=-1\,|\,x,w) = 1 - P(y=+1\,|\,x,w)$
$= 1 - \text{sigmoid}(2)$
$= \text{sigmoid}(-2)$

ML Pipeline

$$\hat{P}(y=+1|x,\hat{w}) = sigmoid\left(\hat{w}^T h(x)\right) = \frac{1}{1+e^{-\hat{w}^T h(x)}}$$



- Historical Bias
- Representation Bias
- Measurement Bias

Demo

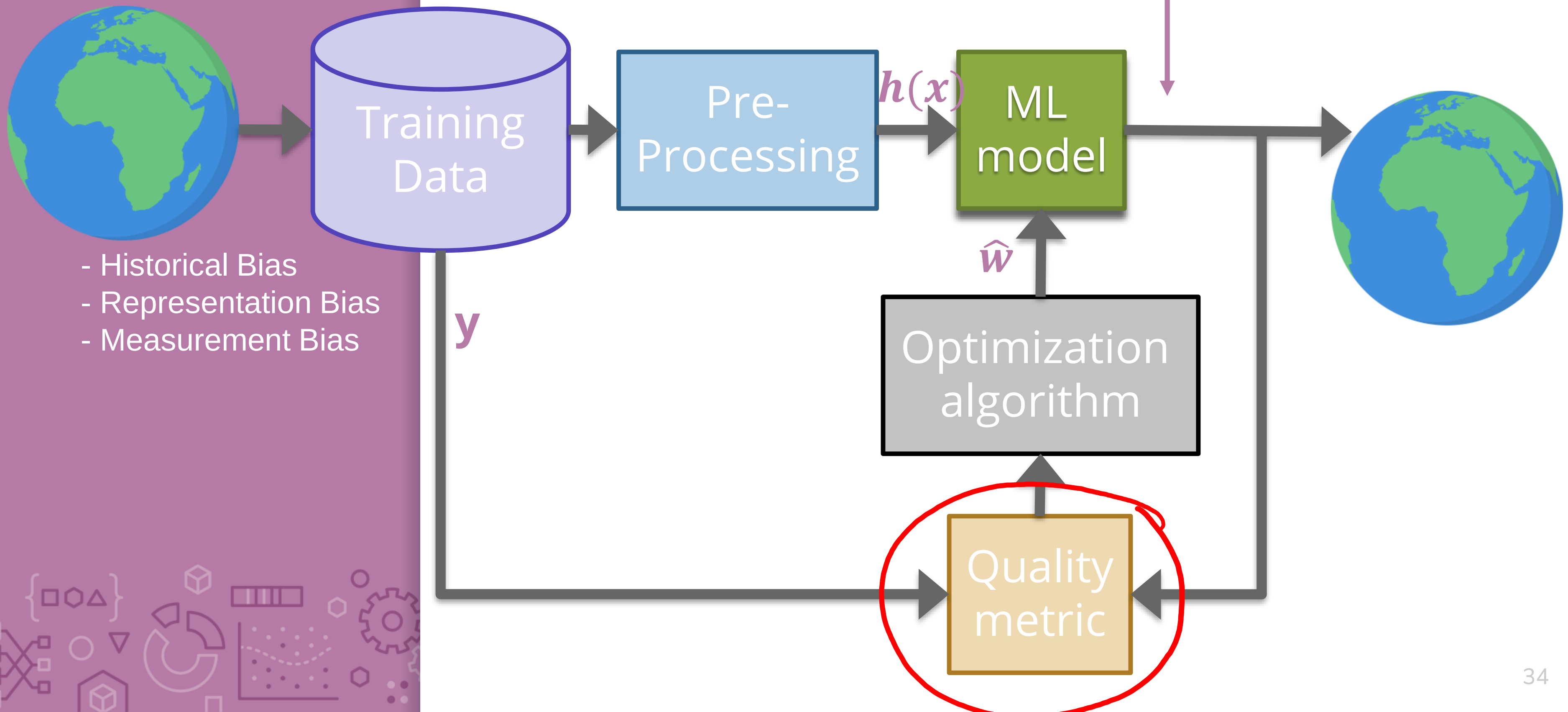
Show logistic demo (see course website)

3:33

Brain Break

ML Pipeline

$$\hat{P}(y=+1|x,\hat{w}) = sigmoid\left(\hat{w}^T h(x)\right) = \frac{1}{1+e^{-\hat{w}^T h(x)}}$$



- Historical Bias
- Representation Bias
- Measurement Bias

Quality Metric = Maximum Likelihood Estimate

$\hat{P}(y_i=+1 \mid x, w) = \frac{1}{1+e^{-w^T h(x_i)}}$ | $P(y_i=-1 \mid x, w) = \frac{e^{-w^T h(x_i)}}{1+e^{-w^T h(x_i)}}$

Quality Metric = Likelihood

Want to compute the probability of seeing our dataset for every possible setting for w. Find w that makes data most likely!

features | true label

Data Point	$h_1(x)$	$h_2(x)$	y	Choose w to maximize
$x^{(1)}, y^{(1)}$	2	1	+1	$\hat{P}(y^{(1)} = +1 \mid x^{(1)}, w)$
$x^{(2)}, y^{(2)}$	0	2	−1	$\hat{P}(y^{(2)} = -1 \mid x^{(2)}, w)$
$x^{(3)}, y^{(3)}$	3	3	−1	$\hat{P}(y^{(3)} = -1 \mid x^{(3)}, w)$
$x^{(4)}, y^{(4)}$	4	1	+1	$\hat{P}(y^{(4)} = +1 \mid x^{(4)}, w)$

$$\ell(w) = P(y_1 \mid x_1, w) \cdot P(y_2 \mid x_2, w) \cdot P(y_3 \mid x_3, w) \cdot P(y_4 \mid x_4, w)$$

$$= \prod_{i=1}^{n} P(y_i \mid x_i, w)$$

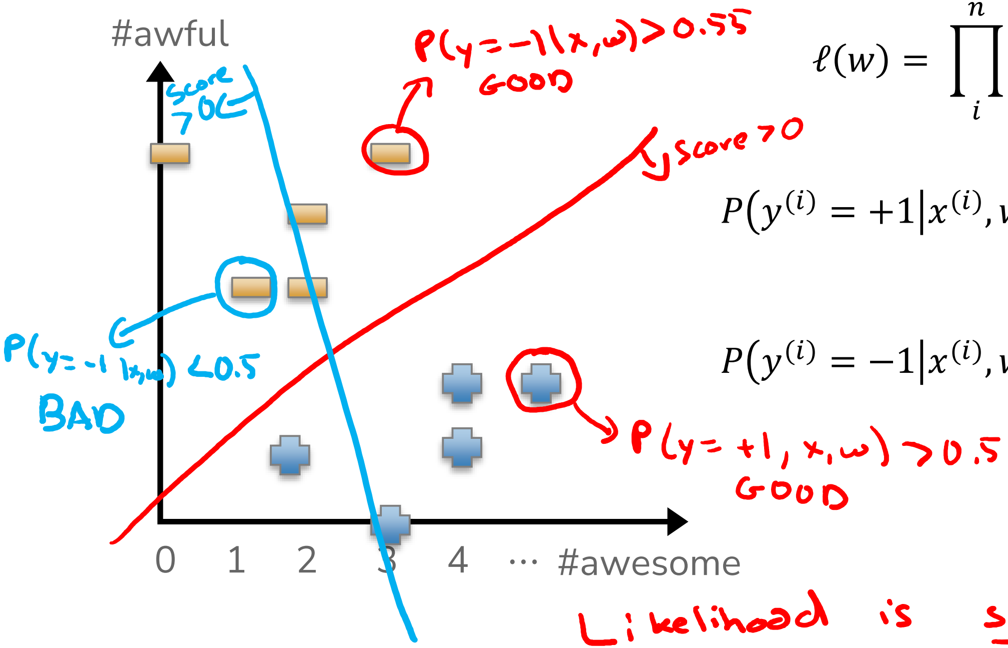
Learn $\hat{w}$

likelihood of seeing data given the predictor is high

likelihood of seeing data given predictor is low

Now that we have our new model, we will talk about how to choose $\hat{w}$ to be the "best fit".

- The choice of w affects how likely seeing our dataset is



$$\ell(w) = \prod_{i}^{n} P(y^{(i)} | x^{(i)}, w)$$

$$P\left(y^{(i)} = +1 \middle| x^{(i)}, w\right) = \frac{1}{1 + e^{-w^T h(x^{(i)})}}$$

$$P\left(y^{(i)} = -1 \middle| x^{(i)}, w\right) = \frac{e^{-w^T h(x^{(i)})}}{1 + e^{-w^T h(x^{(i)})}}$$

Likelihood is smooth

Maximum Likelihood Estimate (MLE)

log(ab) = log(a) + log(b)

Find the w that maximizes the likelihood

$$\hat{w} = \underset{w}{\operatorname{argmax}}\, \ell(w) = \underset{w}{\operatorname{argmax}} \prod_{i=1}^{n} P(y_i | x_i, w)$$

Generally, we maximize the log-likelihood which looks like

$$\hat{w} = \underset{w}{\operatorname{argmax}}\, \ell(w) = \underset{w}{\operatorname{argmax}} \log(\ell(w)) = \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} \log(P(y_i | x_i, w))$$

Also commonly written by separating out positive/negative terms

$$\hat{w} = \underset{w}{\operatorname{argmax}} \sum_{i=1: y_i = +1}^{n} \underbrace{\ln\left(\frac{1}{1 + e^{-w^T h(x)}}\right)}_{\substack{\log P(y_i = +1 | x, w) \\ \text{for pos terms}}} + \sum_{i=1: y_i = -1}^{n} \underbrace{\ln\left(1 - \frac{1}{1 + e^{-w^T h(x)}}\right)}_{\substack{\log P(y_i = -1 | x, w) \\ \text{for neg terms}}}$$

Likelihood vs Error/Loss

- In understanding how to measure error for the classification problem, we want to understand how close a prediction is to the correct class, which means we want to assign a high probability for a correct prediction, and low probability for an incorrect prediction
- Likelihood and error are the inverse of each other:

  Maximizing likelihood = Minimizing Error

SKIPPED

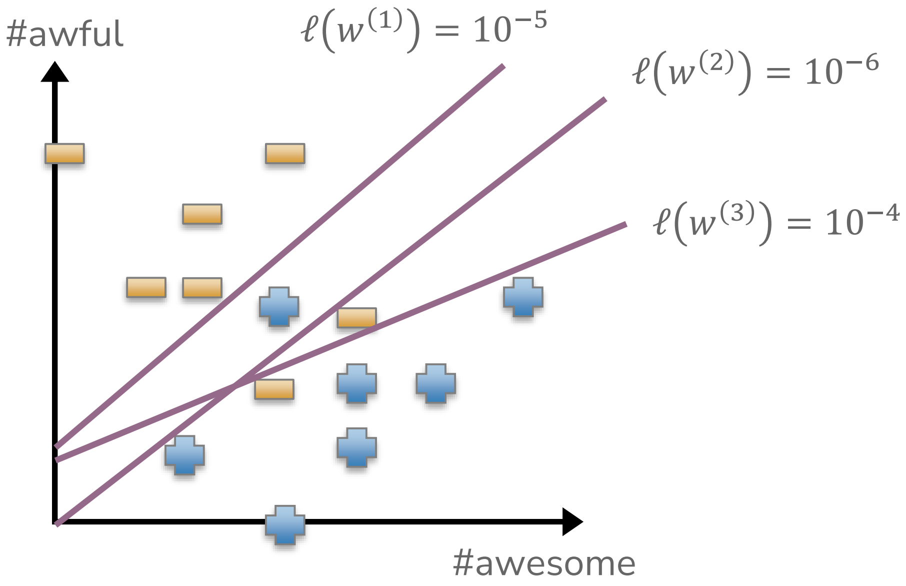
Poll Everywhere

Think

1 min

pollev.com/cs416

Which setting of w should we use?



SKIPPED

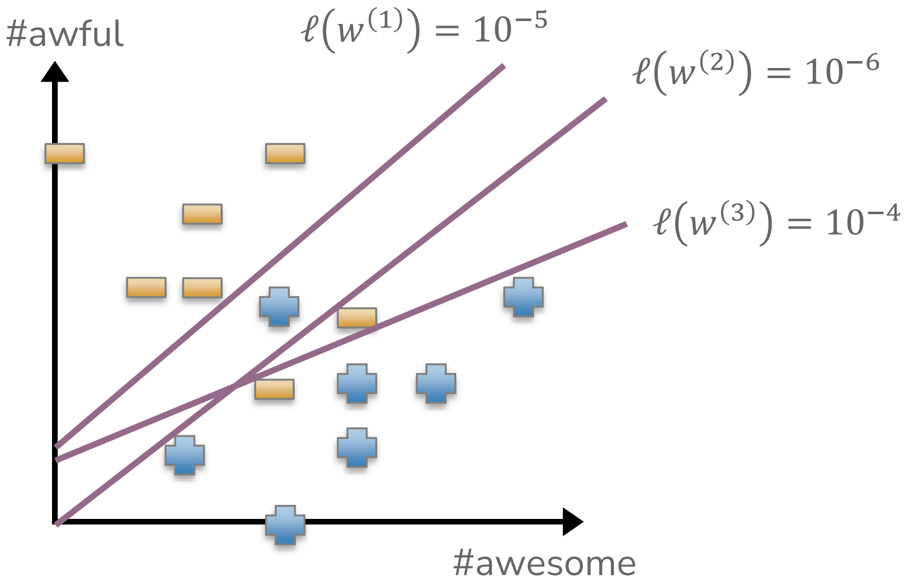
Poll Everywhere

Group

1.5 min

pollev.com/cs416

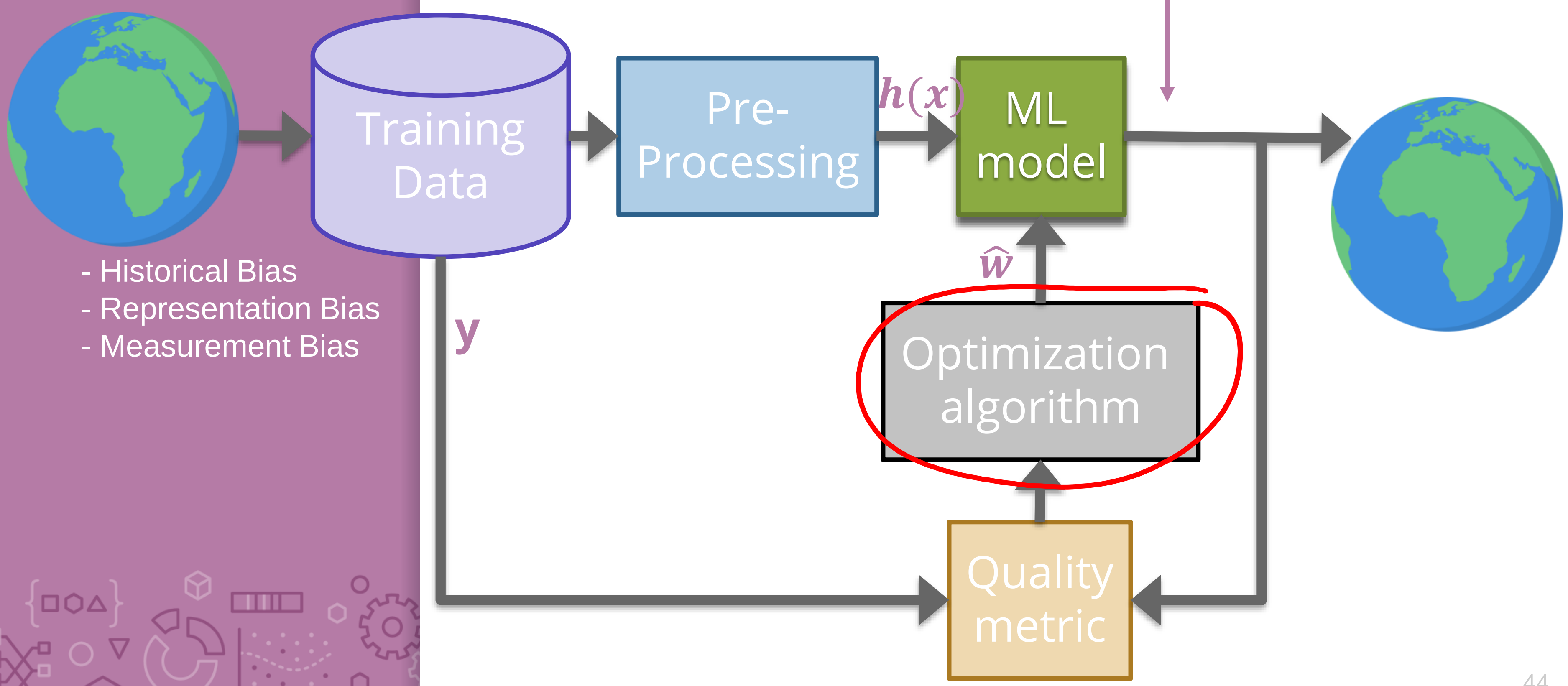
Which setting of w should we use?



Revisiting Gradient Descent / Ascent

ML Pipeline

$$\hat{P}(y=+1|x,\hat{w}) = sigmoid\left(\hat{w}^T h(x)\right) = \frac{1}{1+e^{-\hat{w}^T h(x)}}$$



- Historical Bias
- Representation Bias
- Measurement Bias

Is Gradient Descent *Really* Used in Linear Regression?

- No!
- It **can be**, but isn't in practice.
- Linear regression has a closed form solution. The best weights are:

$$\hat{w} = (XX^T)^{-1}X^Ty$$

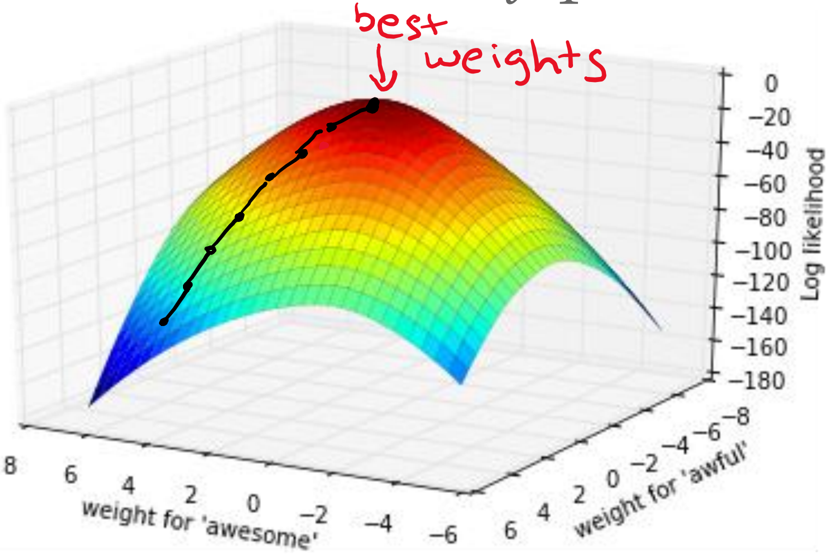
- You don't need to know the formula. What you need to know is that for Linear Regression a closed-form solution, or a solution we can write out with simple mathematical expressions, exists.
- This is not the case with Logistic Regression.
We *must* use Gradient Ascent/Descent!

Finding MLE

No closed-form solution, have to use an iterative method.

Since we are **maximizing** likelihood, we use gradient **ascent**.

$$\hat{w} = \underset{w}{\operatorname{argmax}} \prod_{i=1}^{n} P(y_i | x_i, w)$$



Gradient Ascent

Gradient ascent is the same as gradient descent, but we go "up the hill".

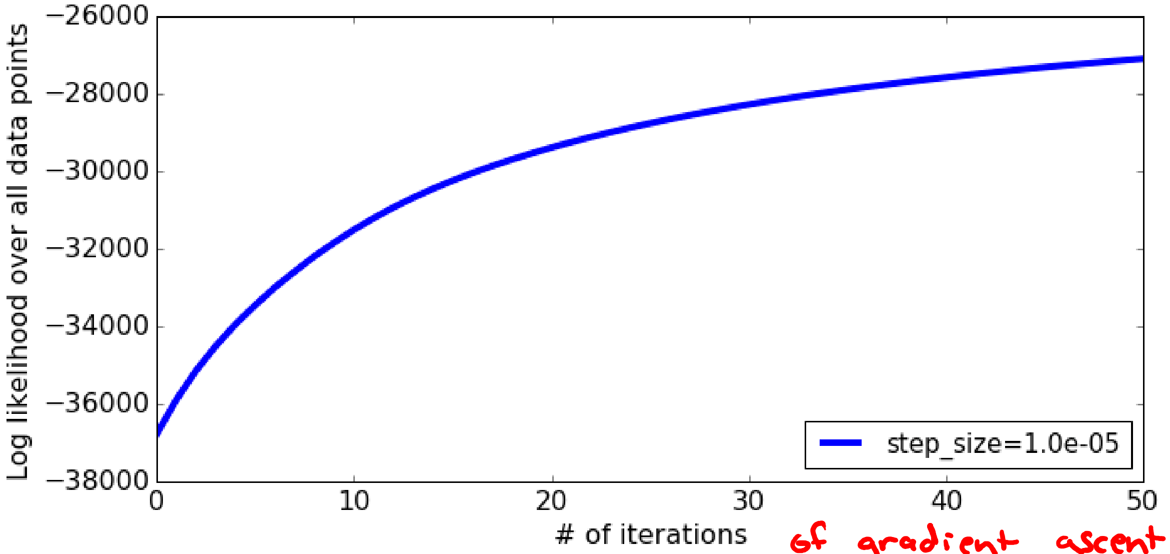
```
start at some (random) point w^(0) when t = 0
while we haven't converged
    w^(t+1) ← w^(t) + η∇ℓ(w^(t))
    t ← t + 1
```

$$w^{(t+1)} \leftarrow w^{(t)} + \underbrace{\eta}_{\text{learning rate}} \underbrace{\nabla \ell(w^{(t)})}_{\text{Gradient of likelihood}}$$

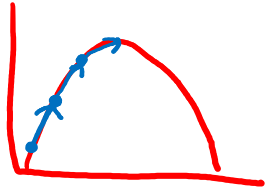
This is just describing going up the hill step by step.

η controls how big of steps we take, and picking it is crucial for how well the model you learn does!

Learning Curve

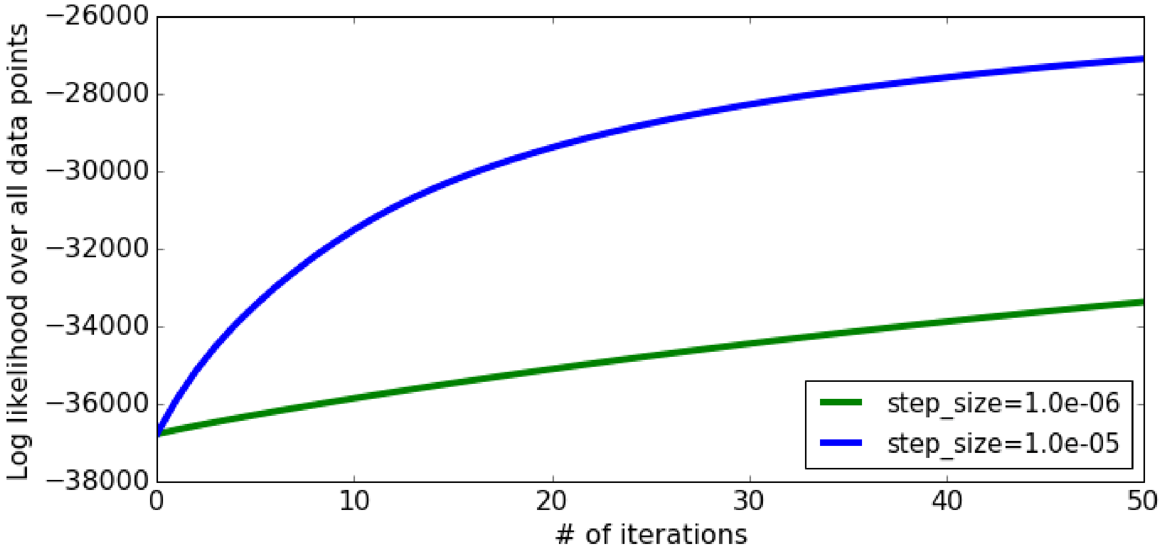


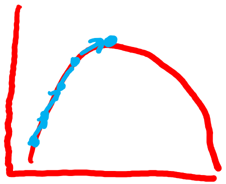
of gradient ascent



Choosing η

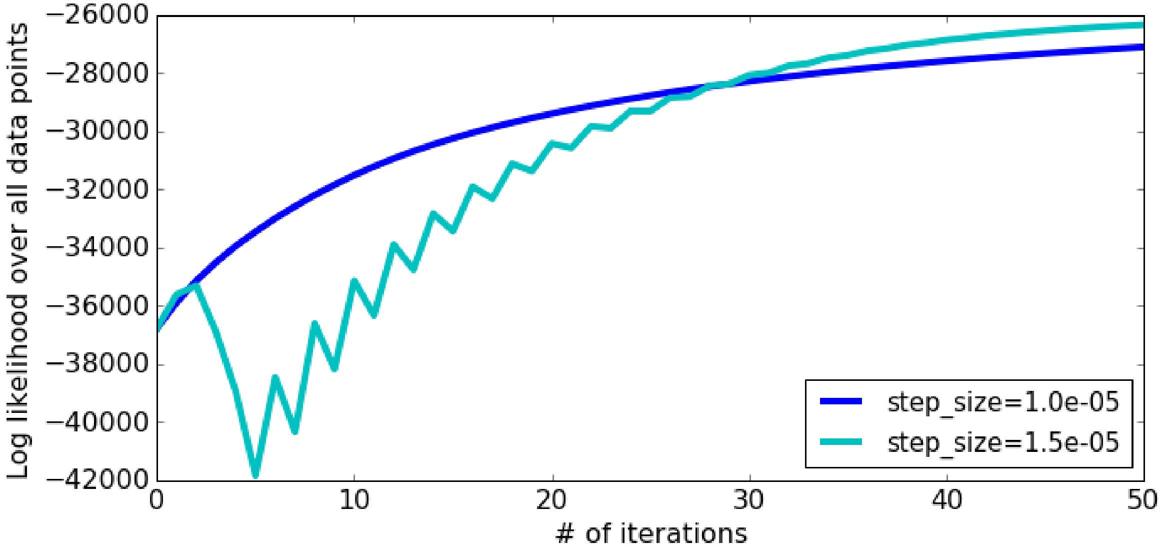
Step-size too small

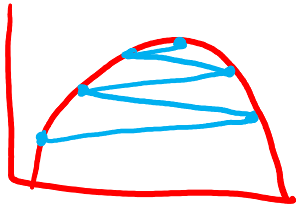




Choosing η

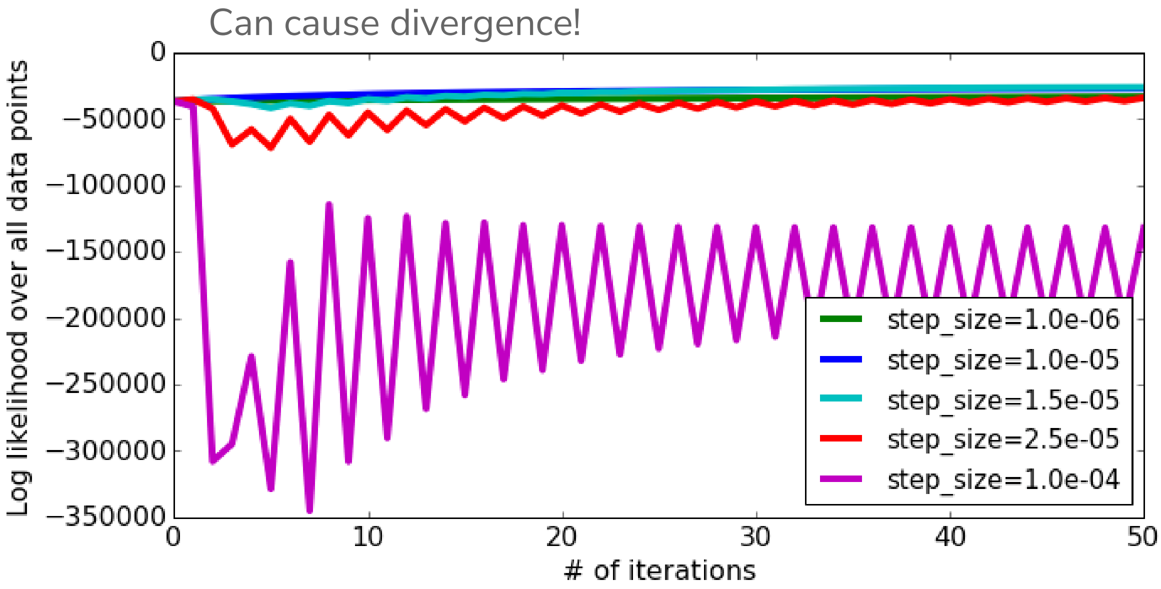
What about a larger step-size?

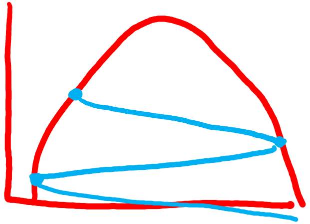




Choosing η

What about a larger step-size?





Choosing η

Unfortunately, you have to do a lot of trial and error ☹

Try several values (generally exponentially spaced)

- Find one that is too small and one that is too large to narrow search range. Try values in between!

Advanced: Divergence with large step sizes tends to happen at the end, close to the optimal point. You can use a decreasing step size to avoid this

$$\eta_t = \frac{\eta_0}{t}$$

annealing

Grid Search

We have introduced yet another hyperparameter that you have to choose, that will affect which predictor is ultimately learned.

If you want to tune multiple hyperparameters at once (e.g., both a Ridge penalty and a learning rate), you will need to try all pairs of settings!

- For example, suppose you wanted to try using a validation set to select the right settings out of:
  - $\lambda \in [0.01, 0.1, 1, 10, 100]$ = 5
  - $\eta_t \in \left[0.001, 0.01, 0.1, 1, \frac{1}{t}, \frac{10}{t}\right]$ → 6
- You will need to train 30 different models and evaluate each one!

Train $5 \cdot 6 = 30$ models

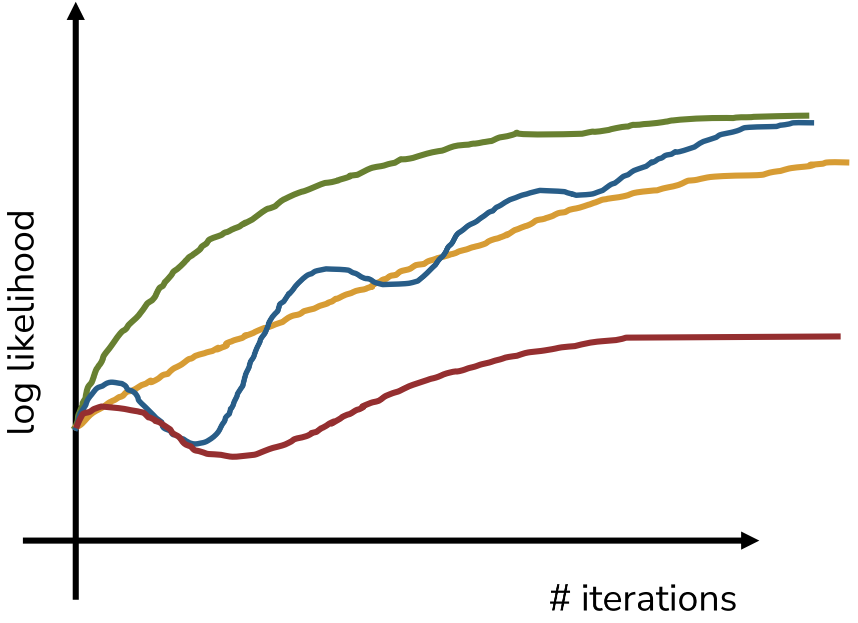
SKIPPED

Poll Everywhere

Think

1 min

- Match the below lines to the following labels:
  - "Very High Learning Rate"
  - "High Learning Rate"
  - "Good Learning Rate"
  - "Low Learning Rate"



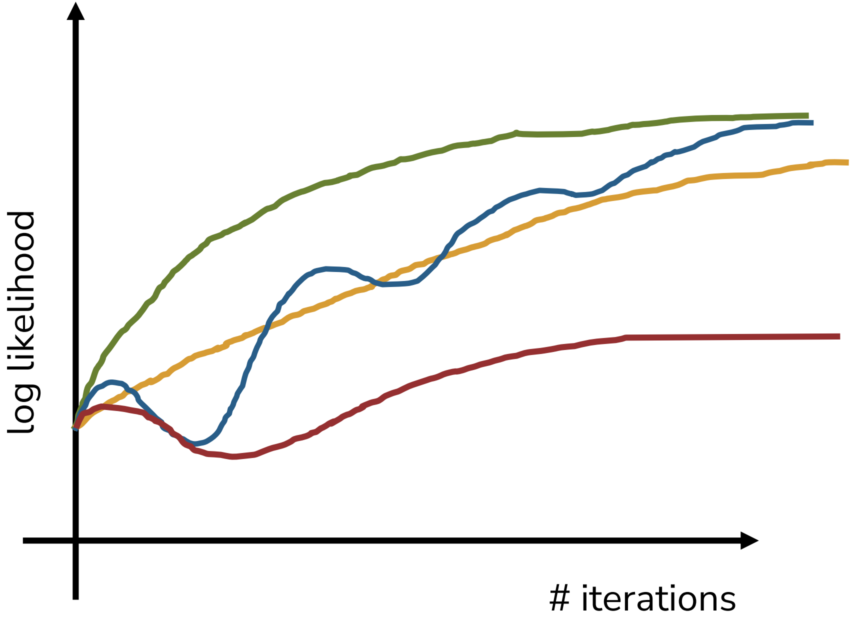
Poll Everywhere

Group

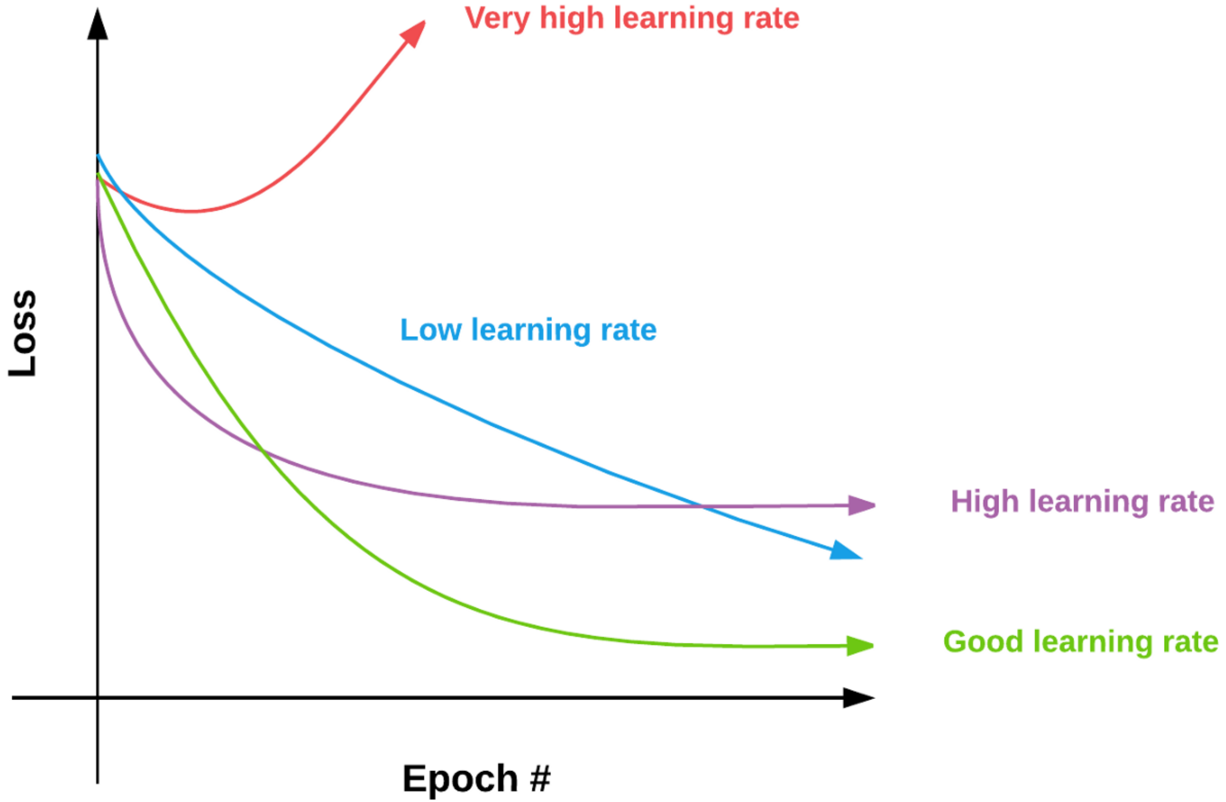
2 min

SKIPPED

- Match the below lines to the following labels:
  - “Very High Learning Rate”
  - “High Learning Rate”
  - “Good Learning Rate”
  - “Low Learning Rate”



Likelihood vs. Loss



Overfitting - Classification

More Features

Linear

Like with regression, we can learn more complicated models by including more features or by including more complex features.

Instead of just using

$h_1(x) = \#awesome$

$h_2(x) = \#awful$

We could use

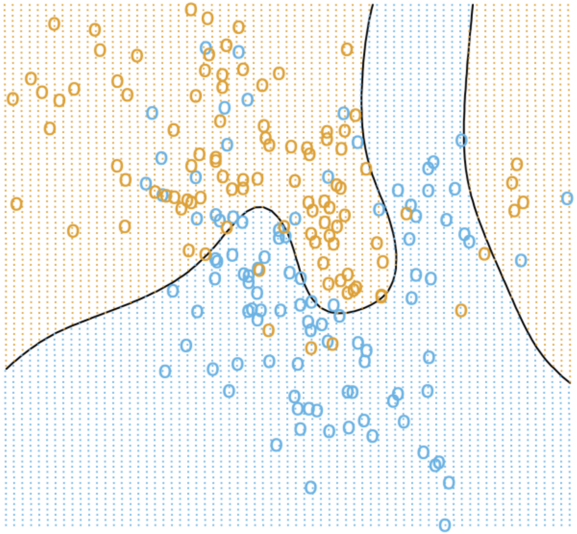
$h_1(x) = \#awesome$

$h_2(x) = \#awful$

$h_3(x) = \#awesome^2$

$h_4(x) = \#awful^2$

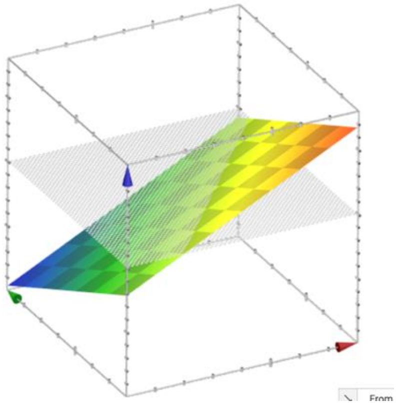
…

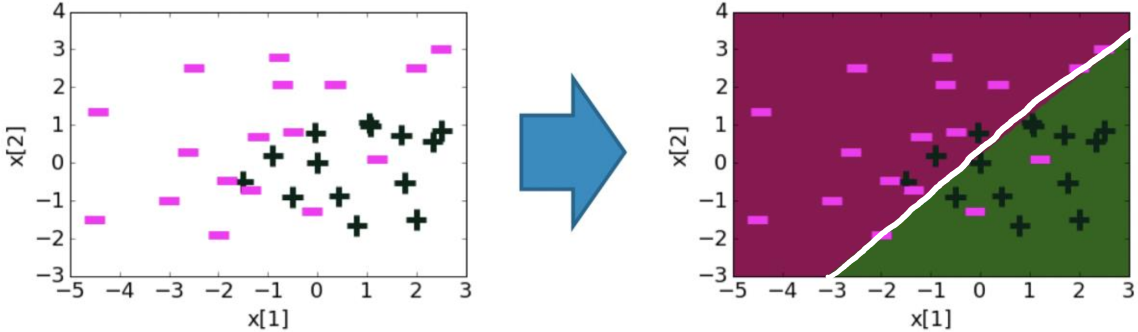


Decision Boundary

$$w^T h(x) = 0.23 + 1.12x[1] - 1.07x[2]$$

Feature	Value	Coefficient learned
$h_0(x)$	1	0.23
$h_1(x)$	x[1]	1.12
$h_2(x)$	x[2]	-1.07

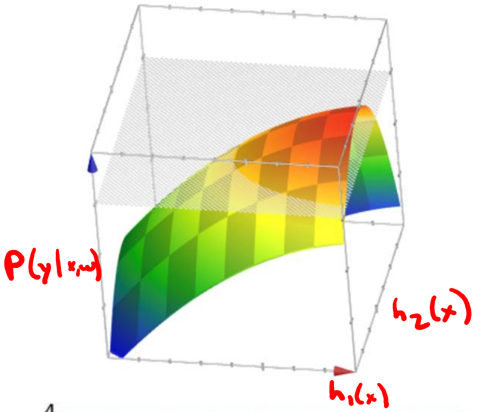


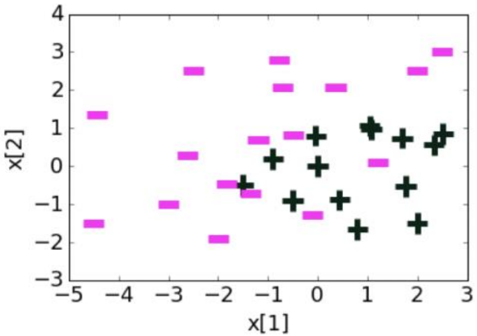


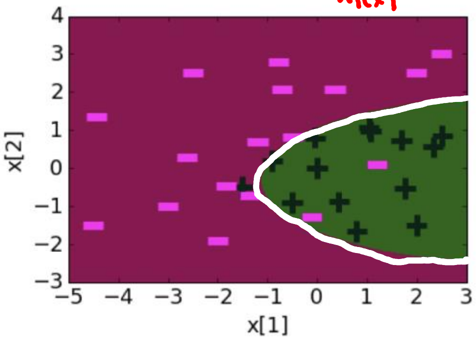
Decision Boundary

$$w^T h(x) = 1.68 + 1.39x[1] - 0.59x[2] - 0.17x[1]^2 - 0.96x[2]^2$$

Feature	Value	Coefficient learned
$h_0(x)$	1	1.68
$h_1(x)$	x[1]	1.39
$h_2(x)$	x[2]	-0.59
$h_3(x)$	$(x[1])^2$	-0.17
$h_4(x)$	$(x[2])^2$	-0.96



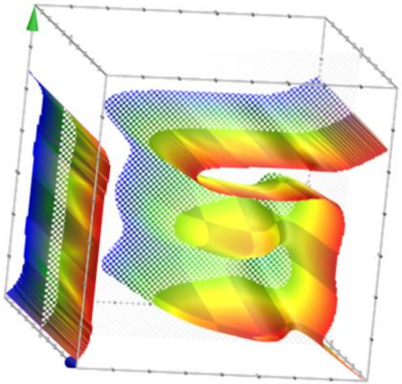


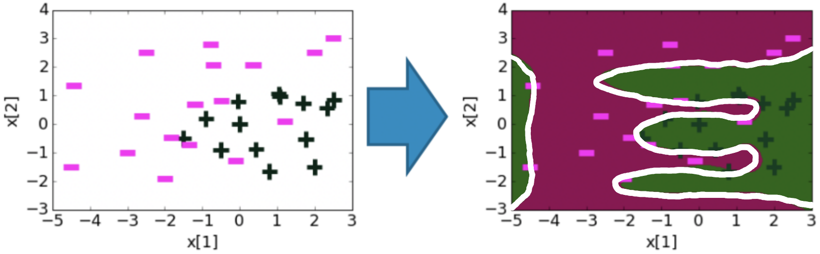


Decision Boundary

$$w^T h(x) = \cdots$$

Feature	Value	Coefficient learned
$h_0(x)$	1	21.6
$h_1(x)$	x[1]	5.3
$h_2(x)$	x[2]	-42.7
$h_3(x)$	$(x[1])^2$	-15.9
$h_4(x)$	$(x[2])^2$	-48.6
$h_5(x)$	$(x[1])^3$	-11.0
$h_6(x)$	$(x[2])^3$	67.0
$h_7(x)$	$(x[1])^4$	1.5
$h_8(x)$	$(x[2])^4$	48.0
$h_9(x)$	$(x[1])^5$	4.4
$h_{10}(x)$	$(x[2])^5$	-14.2
$h_{11}(x)$	$(x[1])^6$	0.8
$h_{12}(x)$	$(x[2])^6$	-8.6

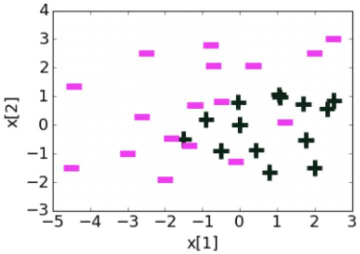


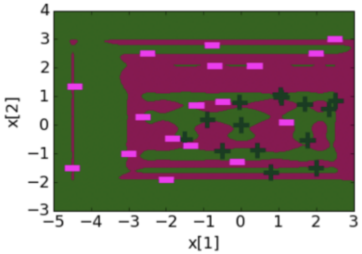


Decision Boundary

$$w^T h(x) = \cdots$$

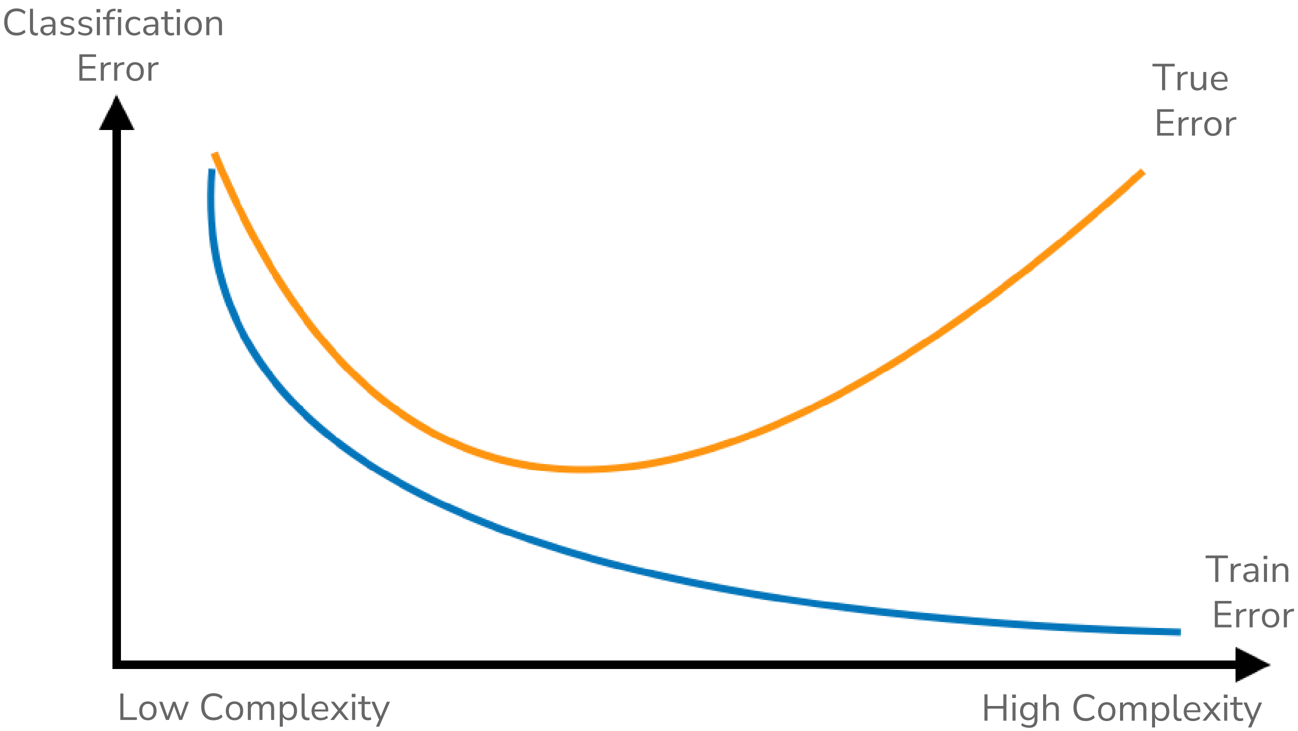
Feature	Value	Coefficient learned
$h_0(x)$	1	8.7
$h_1(x)$	x[1]	5.1
$h_2(x)$	x[2]	78.7
…	…	…
$h_{11}(x)$	$(x[1])^6$	-7.5
$h_{12}(x)$	$(x[2])^6$	3803
$h_{13}(x)$	$(x[1])^7$	21.1
$h_{14}(x)$	$(x[2])^7$	-2406
…	…	…
$h_{37}(x)$	$(x[1])^{19}$	$-2*10^{-6}$
$h_{38}(x)$	$(x[2])^{19}$	-0.15
$h_{39}(x)$	$(x[1])^{20}$	$-2*10^{-8}$
$h_{40}(x)$	$(x[2])^{20}$	0.03





Overfitting

Just like with regression, we see a similar pattern with complexity



In this case, complexity = polynomial degree

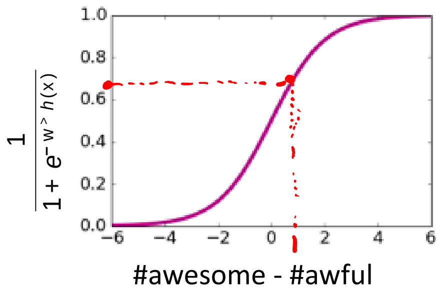
Effects of Overfitting

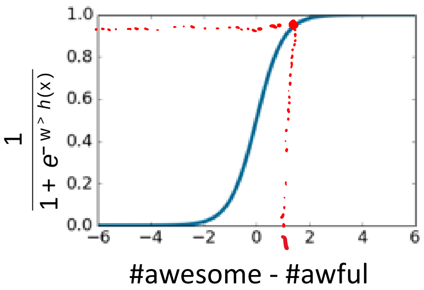
The logistic function become "sharper" with larger coefficients.

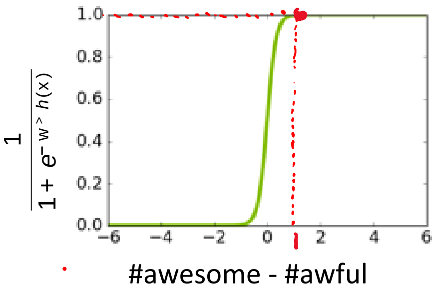
w_0	0
$w_{\#awesome}$	+1
$w_{\#awful}$	-1

w_0	0
$w_{\#awesome}$	+2
$w_{\#awful}$	-2

w_0	0
$w_{\#awesome}$	+6
$w_{\#awful}$	-6





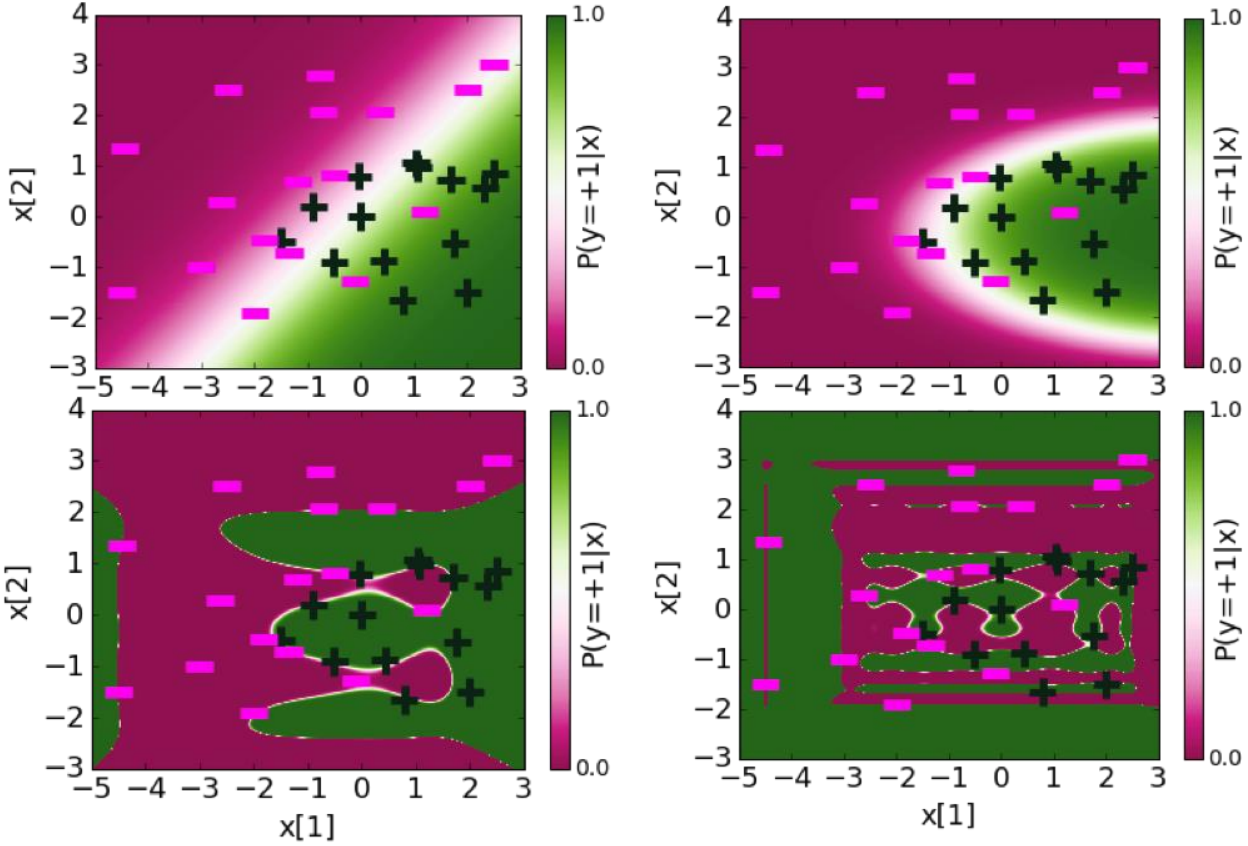


What does this mean for our predictions?

Because the $Score(x)$ is getting larger in magnitude, the probabilities are closer to 0 or 1!

Plotting Probabilities

$$P(y = +1|x) = \frac{1}{1 + e^{-\hat{w}^T h(x)}}$$



Poll Everywhere

Think

0.5 mins

SKIPPED

- What ideas do you have for preventing overfitting in Logistic Regression?
  - (Many possible answers)

Poll Everywhere

Group

1.5 mins

- What ideas do you have for preventing overfitting in Logistic Regression?
  - (Many possible answers)

Same as in Regression

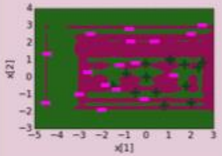
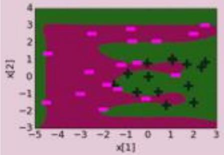
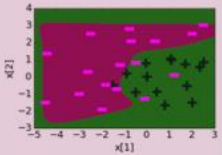
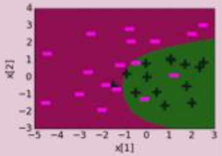
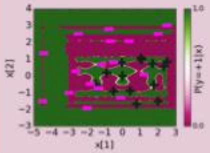
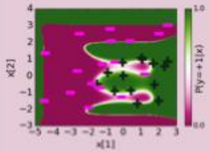
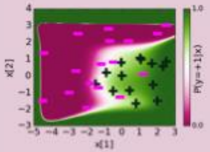
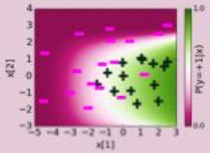
Regularization

L2 Regularized Logistic Regression

Just like in regression, can change our quality metric **during training** to lower the likelihood of learning an overfit model

$$\hat{w} = \underset{w}{\operatorname{argmax}}\, \ell(w) - \lambda||w||_2^2$$

subtract the penalty

Regularization	λ = 0	λ = 0.00001	λ = 0.001	λ = 1
Range of coefficients	-3170 to 3803	-8.04 to 12.14	-0.70 to 1.25	-0.13 to 0.57
Decision boundary				
Learned probabilities				

Some Details

Why do we subtract the L2 Norm?

$$\hat{w} = \underset{w}{\operatorname{argmax}}\, \ell(w) - \lambda ||w||_2^2$$

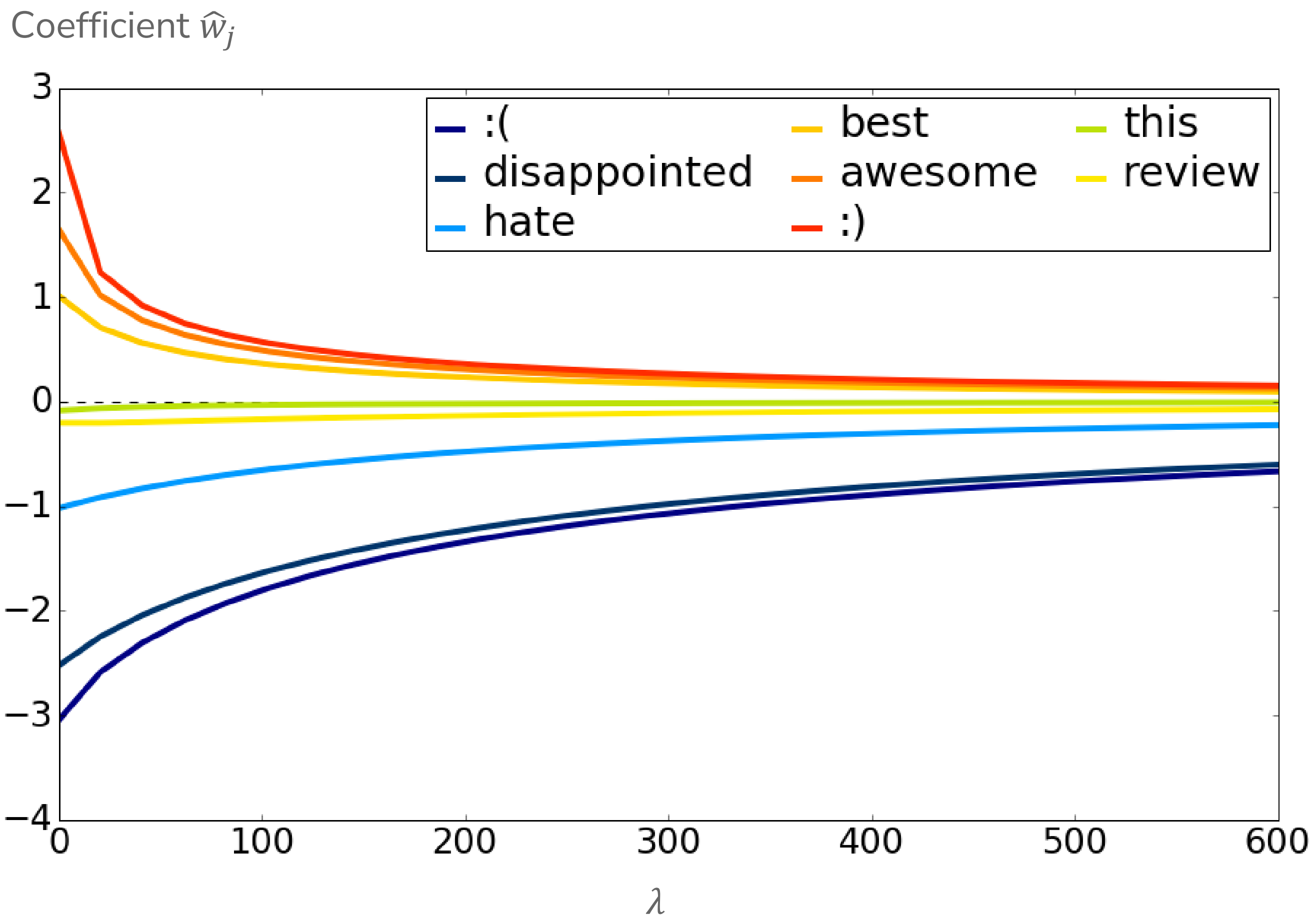
How does λ impact the complexity of the model?

Same as Ridge

How do we pick λ?

Validation, CV

Coefficient Path: L2 Penalty



Other Regularization Penalties?

Could you use the L1 penalty instead? Absolutely!

$$\hat{w} = \underset{w}{\operatorname{argmax}}\, \ell(w) - \lambda||w||_1$$

This is **L1 regularized logistic regression**

It has the same properties as the LASSO

- Increasing λ decreases $||\hat{w}||_1$
- The L1 penalty favors sparse solutions

Poll Everywhere

Think

1 min

- Max wants to find the best Logistic Regression model for a sentiment analysis dataset by tuning the regularization parameter $\lambda \in [0, 10^{-2}, 10^{-1}, 1, 10]$ and the learning rate $\eta \in [10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}]$. He does the following:
  - Runs cross-validation on λ to get the best value for the regularization parameter.
  - For that value of λ, run cross-validation on η to get the best value for the learning rate.
- After running this procedure, he is convinced he has the best Logistic Regression model for his dataset, given the hyper-parameter values he wanted to test.
- **What did Max do wrong?**

Poll Everywhere

Group

2 min

- Max wants to find the best Logistic Regression model for a sentiment analysis dataset by tuning the regularization parameter $\lambda \in [0, 10^{-2}, 10^{-1}, 1, 10]$ and the learning rate $\eta \in [10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}]$. He does the following:
  - Runs cross-validation on λ to get the best value for the regularization parameter.
  - For that value of λ, run cross-validation on η to get the best value for the learning rate.
- After running this procedure, he is convinced he has the best Logistic Regression model for his dataset, given the hyper-parameter values he wanted to test.
- **What did Max do wrong?**

Recap

Theme: Details of logistic classification and how to train it

Ideas:

- Predict with probabilities
- Using the logistic function to turn Score to probability
- Logistic Regression
- Minimizing error vs maximizing likelihood
- Gradient Ascent
- Effects of learning rate
- Overfitting with logistic regression
  - Over-confident (probabilities close to 0 or 1)
  - Regularization