Linear & Logistic Regression

David Quigley

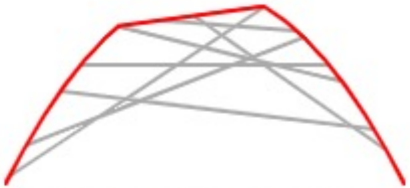
CSCI 5622

2021 Fall

Course Logistics

- Project Phase 1: Feedback returned
- Project Phase 2: Due 9/30
- Problem Sets: Problem Set 1 Feedback expected Thursday 9/23
    - Currently anticipating a minor delay.
- Problem Set 2: Due 10/7

Concave vs. Convex (Correction)



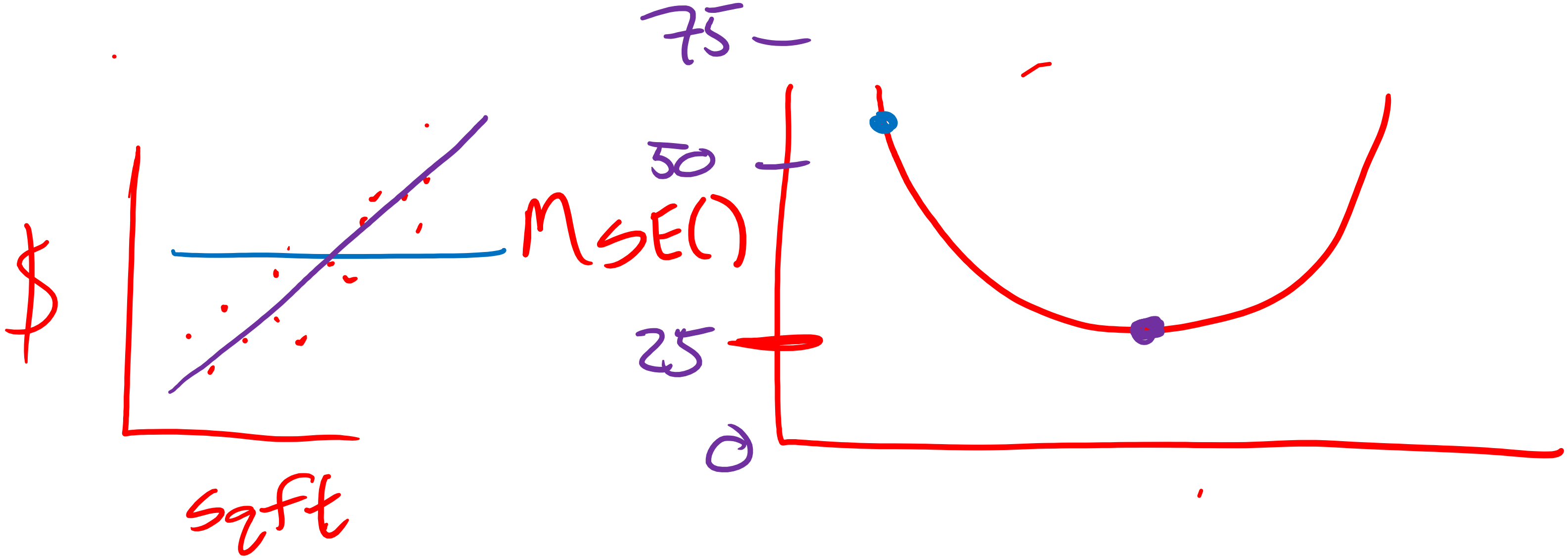
A concave function:
no line segment joining
two points on the graph
lies above the graph
at any point

A convex function:
no line segment joining
two points on the graph
lies below the graph
at any point

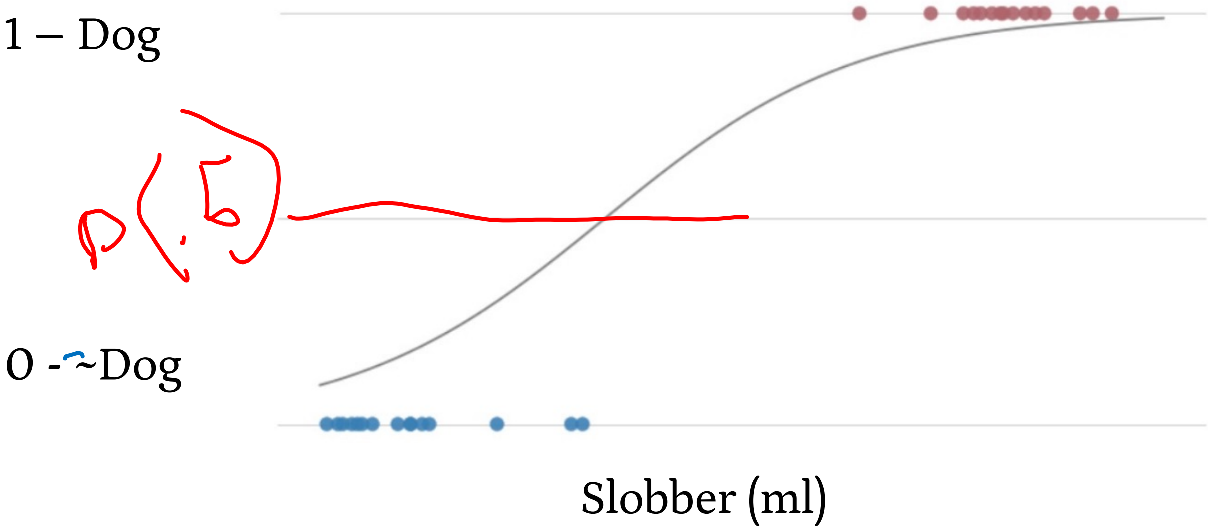
A function that is neither
concave nor convex:
the line segment shown lies
above the graph at some
points and below it at others

Mean Squared Error is Convex!

- We are reducing our MSE( ) as we improve
- We want to *minimize* the MSE( )
- Our same assumptions (local minimum = global minimum, etc.) still hold



Our Problem Space – Dog Slobber



if p(X) >= .5, y = 1

else, y = 0

Logistic Regression – log-odds *or* logit

Odds = e^{score}

ln(odds) = score

END

Logistic Regression – Impact

Feature	*Bias*	X_1 = “Santa”	X_2 = “Dreidel”	X_3 = “Christmas”	X_3 = “Bad”	X_4 = “Hate”
Weight	-0.1	10.0	15.0	12.0	-2.0	-4.0

Take the song line “Oh Dreidel, Dreidel, Dreidel, I made it out of clay…”
What is the impact of adding another “Dreidel”?

Logistic Regression – Impact

Feature	*Bias*	X_1 = "Santa"	X_2 = "Dreidel"	X_3 = "Christmas"	X_3 = "Bad"	X_4 = "Hate"
Weight	-0.1	10.0	15.0	12.0	-2.0	-4.0

Take the song line "Oh Dreidel, Dreidel, Dreidel, I made it out of clay... but you were Bad..."

What is the impact of adding another "Dreidel"?

$$\exp(w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5)$$

$$\text{Odds} = e^{-0.1 + 10.0 * 0 + 15.0 * 3 + 12.0 * 0 + -2.0 * 1 + -4.0 * 0}$$

vs.

$$\text{Odds} = e^{-0.1 + 10.0 * 0 + 15.0 * 4 + 12.0 * 0 + -2.0 * 1 + -4.0 * 0}$$

Logistic Regression – Impact

Feature	_Bias_	X_1 = “Santa”	X_2 = “Dreidel”	X_3 = “Christmas”	X_3 = “Bad”	X_4 = “Hate”
Weight	-0.1	10.0	15.0	12.0	-2.0	-4.0

Take the song line “Oh Dreidel, Dreidel, Dreidel, I made it out of clay... but you were Bad...”

What is the impact of adding another “Dreidel”?

$\exp(w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5)$

$\text{Odds} = \left[e^{-0.1 + 10.0 * 0 + 15.0 * 3 + 12.0 * 0 + -2.0 * 1 + -4.0 * 0}\right]$

vs.

$\text{Odds} = e^{-0.1 + 10.0 * 0 + 15.0 * 4 + 12.0 * 0 + -2.0 * 1 + -4.0 * 0}$

$\text{or Odds} = e^{15.0} * \left[e^{-0.1 + 10.0 * 0 + 15.0 * 3 + 12.0 * 0 + -2.0 * 1 + -4.0 * 0}\right]$

Odds improved by a factor of (approximately) 3,269,017

Probabilistic Classification

Generative Probability

Measure the *joint* probability p(x, y)

- requires assumptions about x, relationship to y

Naïve Bayes

More complex conclusions

Discriminative Probability

Model the *conditional* probability p(y | x)

Logistic Regression

Faster

Fewer Assumptions

Finding Weights

Best Guess

- Estimated from some background information or expertise

Finding Weights

Best Guess

- Estimated from some background information or expertise

Equal Numbers

- If we don't have any good ideas, just weight everything evenly
- *This approach isn't exactly the best, we'll explore why later.*

Finding Weights

Best Guess

- Estimated from some background information or expertise

Equal Numbers

- If we don't have any good ideas, just weight everything evenly
- *This approach isn't exactly the best, we'll explore why later.*

"Best" Guess

- What if we don't have any information? Take any ol' guess!
- Random numbers!*

*Random small numbers

Checking Weights – Logistic Regression

- This should *hopefully* end up feeling like Linear Regression...

Checking Weights – Logistic Regression

Likelihood

$p(y=1|x; w) = 1 / 1 + e^{-(w \cdot x)}$

$p(y=0|x; w) = 1 - (1 / 1 + e^{-(w \cdot x)})$

Checking Weights – Logistic Regression

Likelihood

$p(y=1|x; w) = 1 / 1 + e^{-(w \cdot x)}$

$p(y=0|x; w) = 1 - (1 / 1 + e^{-(w \cdot x)})$

$\$p(y|x; w) = (1 / (1 + e^{-(w \cdot x)}))^{y} * (1 - (1 / (1 + e^{-(w \cdot x)})))^{1-y} \quad \$$

$\$$Bernoulli Random Variable$\$$

Likelihood of Weights

Given $\{x_i, y_i\}_{i=1 \rightarrow n}$, chosen weights w

$$L(w) = p(\{y_i\}_{i=1 \rightarrow n} | \{x_i\}_{i=1 \rightarrow n}; w)$$
$$= \Pi_{i=1 \rightarrow n}(p(y_i|x_i, w))$$
$$= \Pi_{i=1 \rightarrow n}((1 / 1 + e^{-(w \cdot xi)})^{yi} * (1 - (1 / 1 + e^{-(w \cdot xi)}))^{1-yi}$$

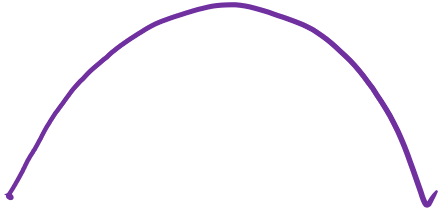
Likelihood of Weights

Given $\{x_i, y_i\}_{i=1\rightarrow n}$, chosen weights w

$$
\begin{aligned}
L(w) \quad &= p(\{y_i\}_{i=1\rightarrow n} | \{x_i\}_{i=1\rightarrow n}; w) \\
&= \Pi_{i=1\rightarrow n}[p(y_i|x_i, w)] \\
&= \Pi_{i=1\rightarrow n}[(1 / 1 + e^{-(w \cdot xi)})^{yi} * (1 - (1 / 1 + e^{-(w \cdot xi)})^{1-yi}]
\end{aligned}
$$

We want to maximize L(w)

- *We want the weights that are most likely to generate our y (answers) given our x (training examples)*



Log-Likelihood of Weights

$$
\begin{aligned}
LL(w) &= \log(L(w)) \\
&= \log[\Pi_{i=1 \rightarrow n}[(1 / 1 + e^{-(w \cdot xi)})^{yi} * (1 - (1 / 1 + e^{-(w \cdot xi)})^{1-yi}]] \\
&= \Sigma_{i=1 \rightarrow n}[\log[(1 / 1 + e^{-(w \cdot xi)})^{yi}] + \log[(1 - (1 / 1 + e^{-(w \cdot xi)})^{1-yi}]]
\end{aligned}
$$

Taking the log of a product...

Log-Likelihood of Weights

$LL(w) = \log(L(w))$

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

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

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

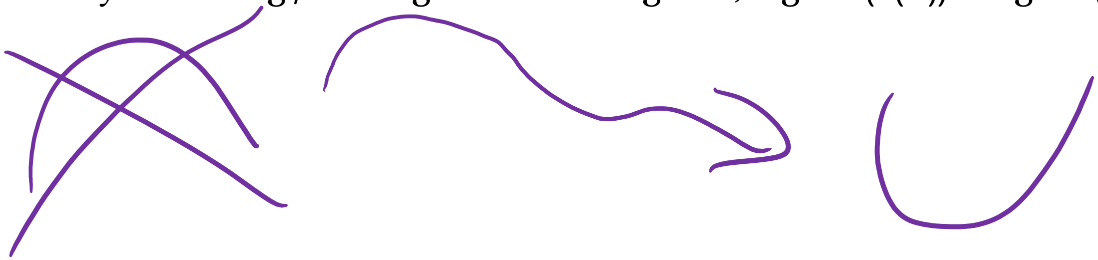
Pulling out the exponent (y_i *and* $1 - y_i$)...

Negative Log-Likelihood of Weights

$$
\begin{aligned}
NLL(w) &= -\log(L(w)) \\
&= -\log[\Pi_{i=1 \rightarrow n}[(1 / 1 + e^{-(w \cdot xi)})^{yi} * (1 - (1 / 1 + e^{-(w \cdot xi)})^{1-yi}]] \\
&= -\Sigma_{i=1 \rightarrow n}[\log[(1 / 1 + e^{-(w \cdot xi)})^{yi}] + \log[(1 - (1 / 1 + e^{-(w \cdot xi)})^{1-yi}]] \\
&= -\Sigma_{i=1 \rightarrow n}[y_i * \log[(1/1+e^{-(w \cdot xi)})] + (1 - y_i) * \log[(1 - (1/1+e^{-(w \cdot xi)}))]]
\end{aligned}
$$

In math, optimization is typically done as a *minimization* problem*...

*thanks to log() being monotonically increasing / making our values negative, argmax(L(w)) = argmin(NLL(w))



Optimizing using gradient descent

```
w = //initial weights (vector)
x = //training examples (2d matrix)
y = //training answers (1d vector)
Lambda = //rate
While not converged():
        For k in range(0,D): //for each dimension
                update[k] = 0
                for i in range(0,n): //for each training example
```

$$\text{update}[k] = \Sigma_{i=1 \rightarrow n}\left[\left(1/1+e^{-(w \cdot xi)}\right) - y_i\right] * x_{ik}$$

```
w = w – lambda * update
```

Gradient Descent – Predicting the weather

We have relevant features (precipitation, humidity, wind, latitude, longitude, time of day, day of year…

- say 99 features, across 19,354 “incorporated places” in us
- current temperature regression

We have sampled data points

- 1 sample per site per minute for 30 years

Gradient Descent – Predicting the weather

We have relevant features (temperature, humidity, wind, latitude, longitude, time of day, day of year...

- say 99 features, across 19,354 “incorporated places” in us
- current precipitation classification

We have sampled data points

- 1 sample per site per minute for 30 years

Gradient Descent – Predicting the weather

We have relevant features (temperature, humidity, wind, latitude, longitude, time of day, day of year...

- say 99 features, across 19,354 “incorporated places” in us
- plus current precipitation

We have sampled data points

- 1 sample per site per minute for 30 years

3.05e+13 features

There are bigger problems out there!

We’ll solve that problem now!

Stochastic Gradient Descent

$w \leftarrow w - \lambda * \nabla_w NLL(w)$

But we can't really fit everything into memory to calculate the true gradient… So we estimate the gradient using one training example at a time.

$w \leftarrow w - \lambda * \nabla_w NLL(w \mid x_i, y_i)$

all dimensions ~~entire dataset~~

$w_k \leftarrow w_k - \lambda * [\,(1/1+e^{-(w \cdot xi)}) - y_i] * x_{ik}$ for $k = 0 \rightarrow D$

Compared to $w_k \leftarrow w_k - \lambda * \Sigma_{i=1 \rightarrow n}[\,(1/1+e^{-(w \cdot xi)}) - y_i] * x_{ik}$

Optimizing using stochastic gradient descent

```
w = //initial weights
x = //training examples (2d Matrix)
y = //training answers (1d vector)
Lambda = //rate
While not converged():
        shuffle(x, y) // get a random (equivalent) order of examples,
                answers
        For i in range(0,n): //for each point
                gradient = [(1/1+e^-(w · xi)) - y_i]
                for k in range(0,D): //for each dimension
                        w[k] = w[k] – lambda * gradient * x[i,k]
```

Optimizing using gradient descent COMPARE

```
w = //initial weights
x = //training examples (2d matrix)
y = //training answers (1d vector)
Lambda = //rate
While not converged():
        For k in range(0,D): //for each dimension
                update[k] = 0
                for i in range(0,n): //for each training example
```

$$update[k] = \Sigma_{i=1 \rightarrow n}[\,(1/1+e^{-(w \cdot xi)}) - y_i] * x_{ik}$$

```
        w = w – lambda * update
```

Stochastic Gradient Descent

So now we're going through, updating our weights piecemeal, always getting closer to convergence...

Every loop through the training set is an **epoch**

Improvement: Mini-Batch Stochastic Gradient Descent

If we only work with one piece of data at a time, we're working very inefficiently

- *we don't have Teras of memory, but we have Gigs...*

Work with a small batch (e.g. 256 samples) of our data!

$w_k \leftarrow w_k - \lambda * \Sigma_{i=1 \rightarrow b}[(1/1+e^{-(w \cdot xi)}) - y_i] * x_{ik}$

Good linear algebra libraries make this *fast*.

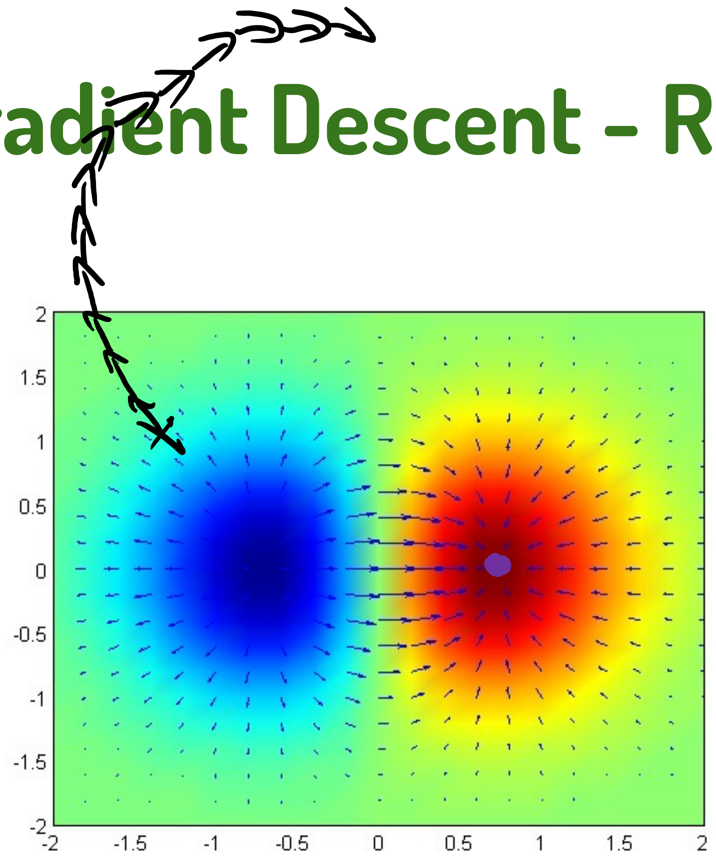
Note: Why don't we use even weights? 0?

In short: It doesn't work as well thanks to *colinearity*.

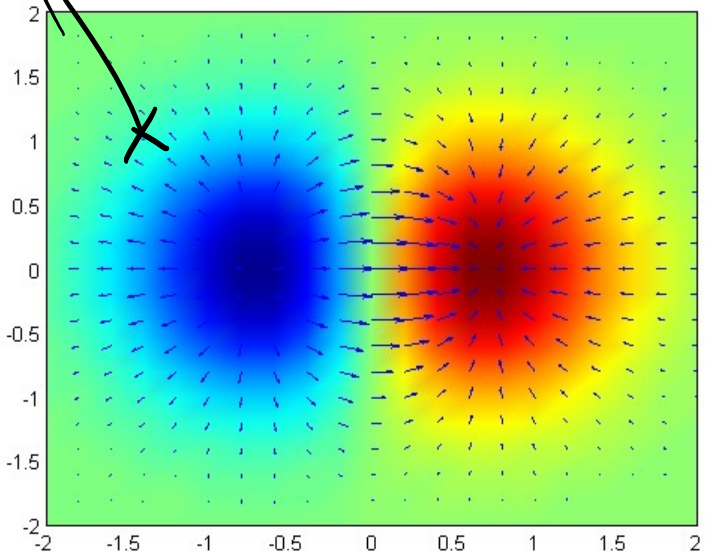
But how do we get our random numbers? It's an open question...

https://machinelearningmastery.com/why-initialize-a-neural-network-with-random-weights/

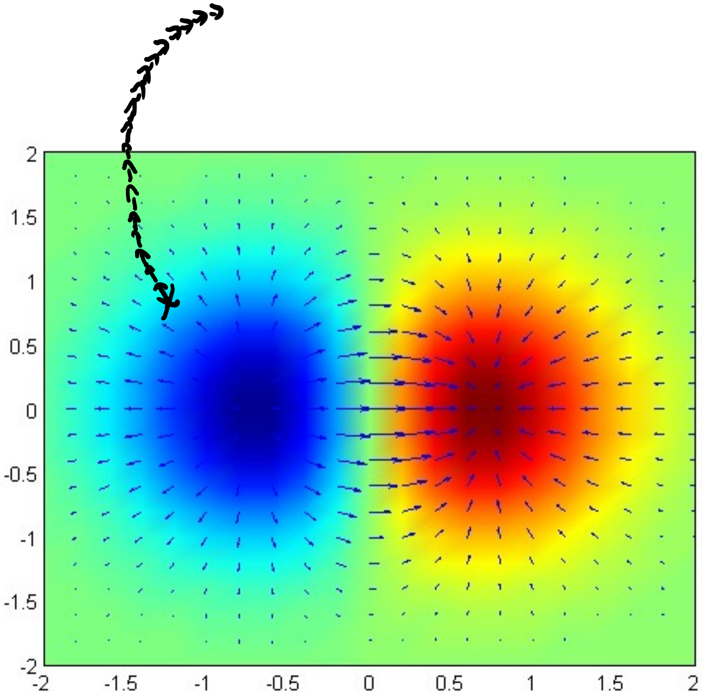
Stochastic Gradient Descent – Rates (1)



Stochastic Gradient Descent – Rates



Stochastic Gradient Descent - Rates



Choosing a Learning Rate

Try some out!

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

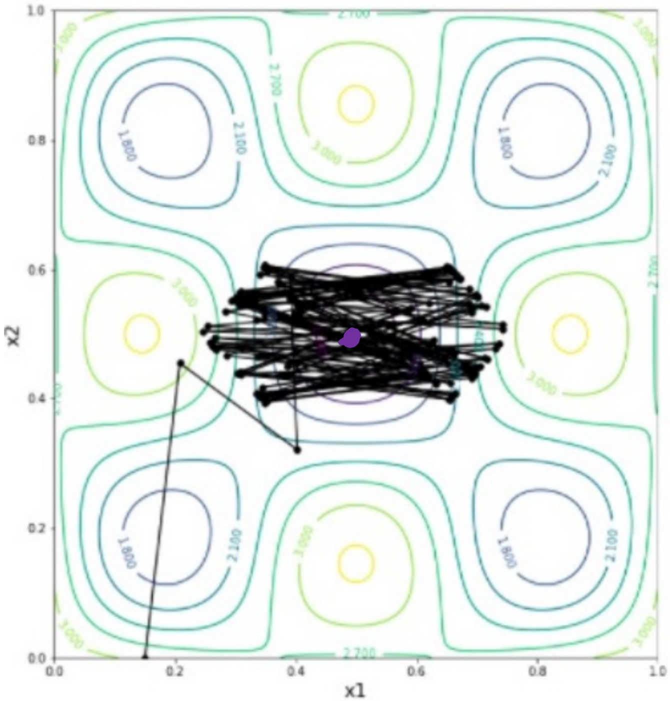
Scale back over time

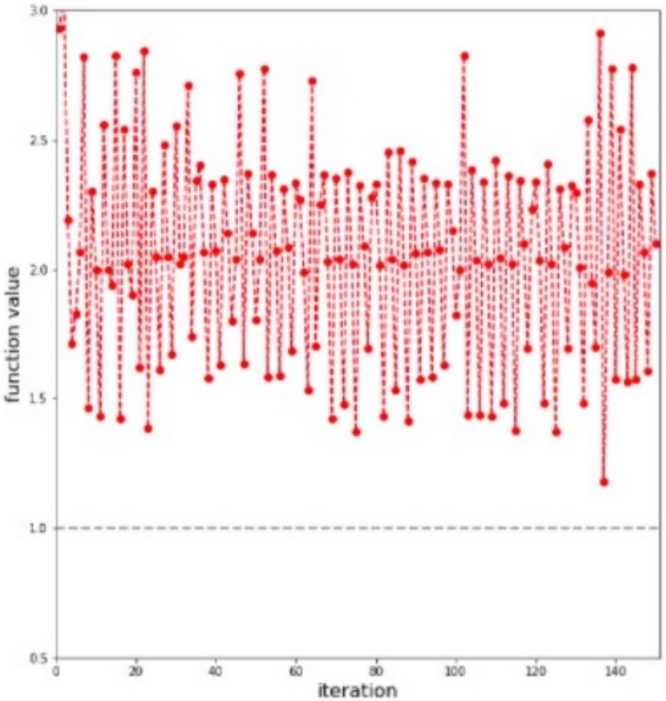
- You can easily start bouncing around the true minimum

Scale rates for each feature

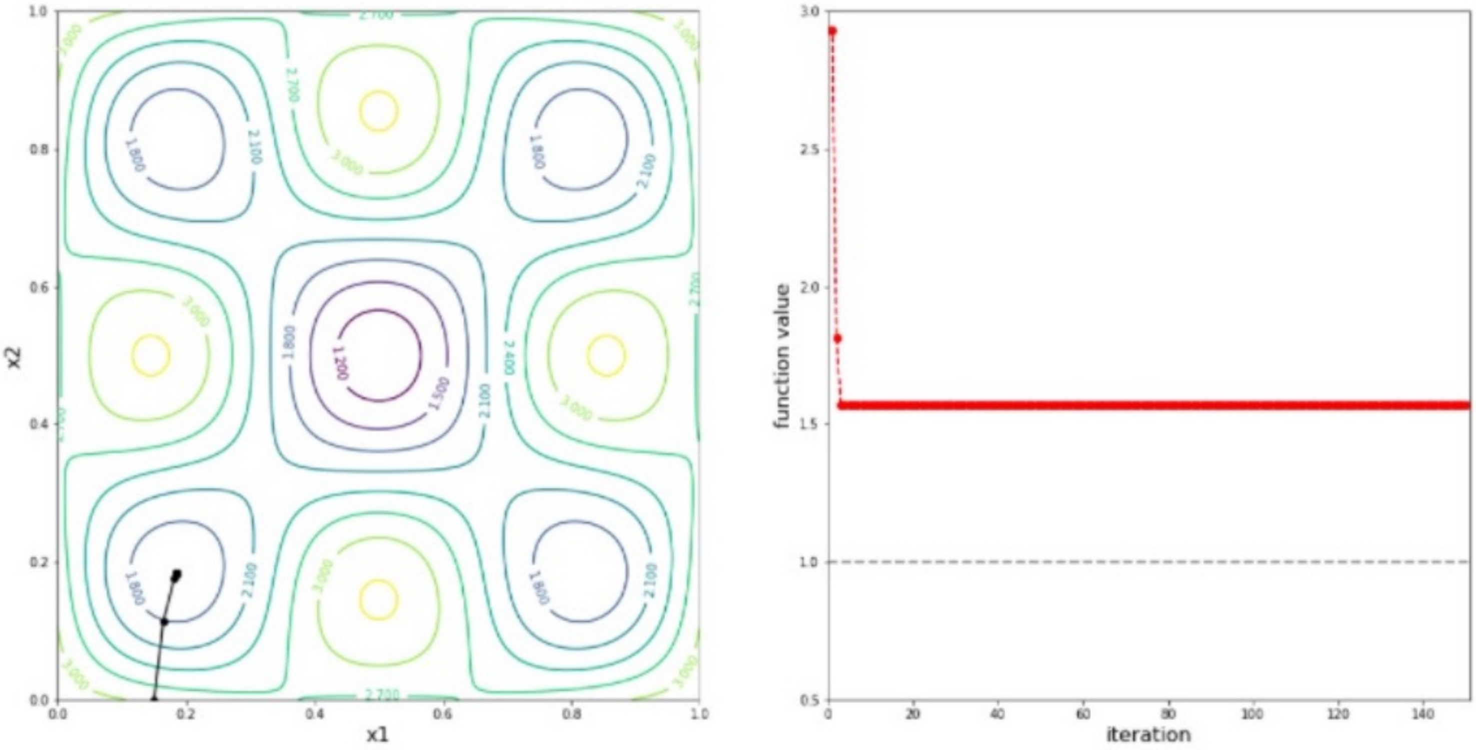
- *this is an advanced technique, we won't get into it today.*

Optimizing Weights – changing λ over time

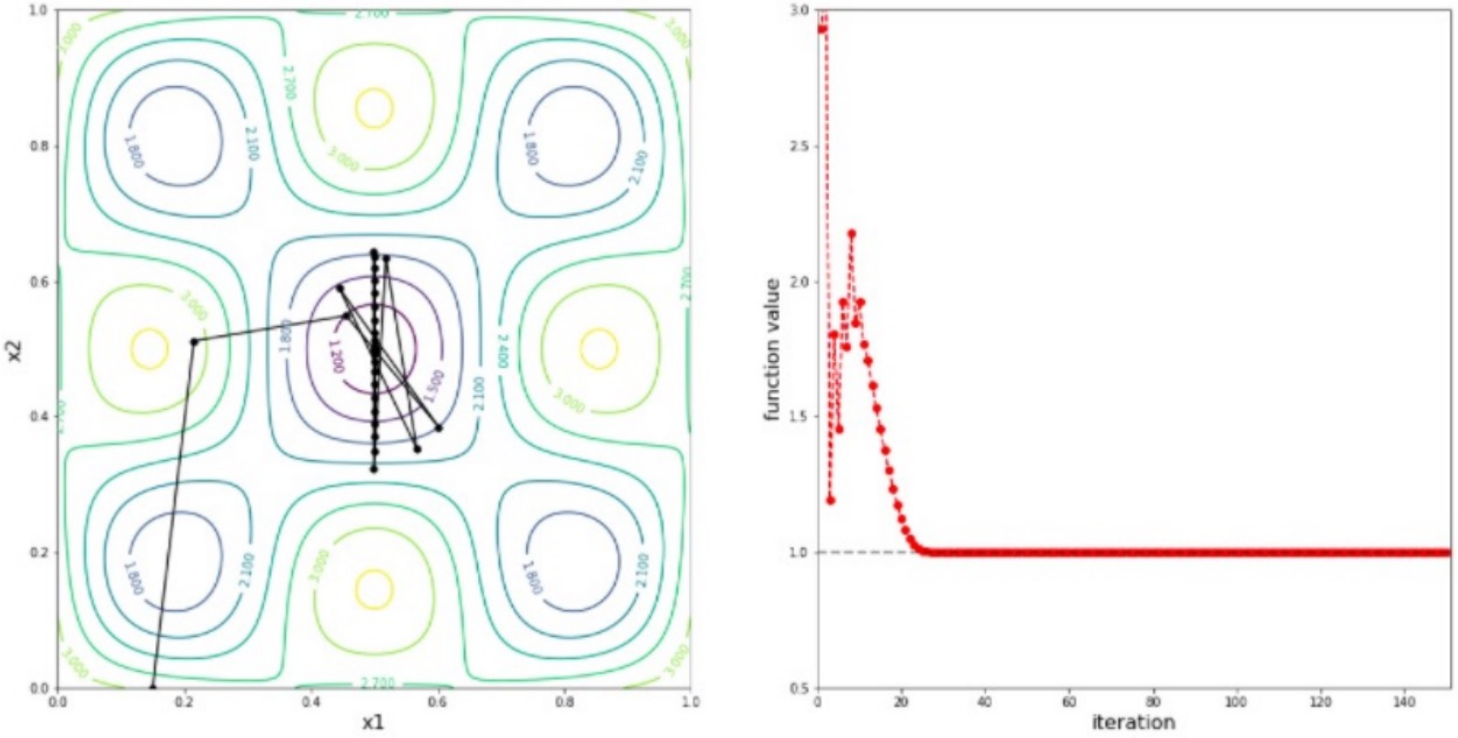




Optimizing Weights – changing λ over time

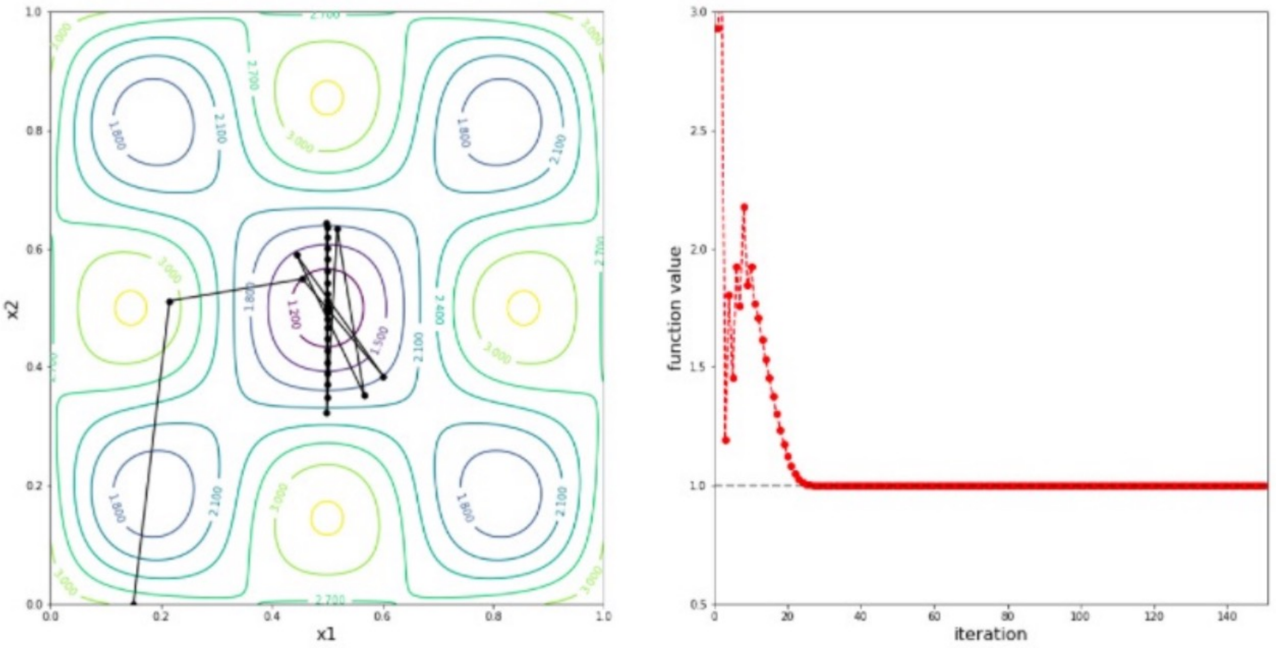


Optimizing Weights – changing λ over time



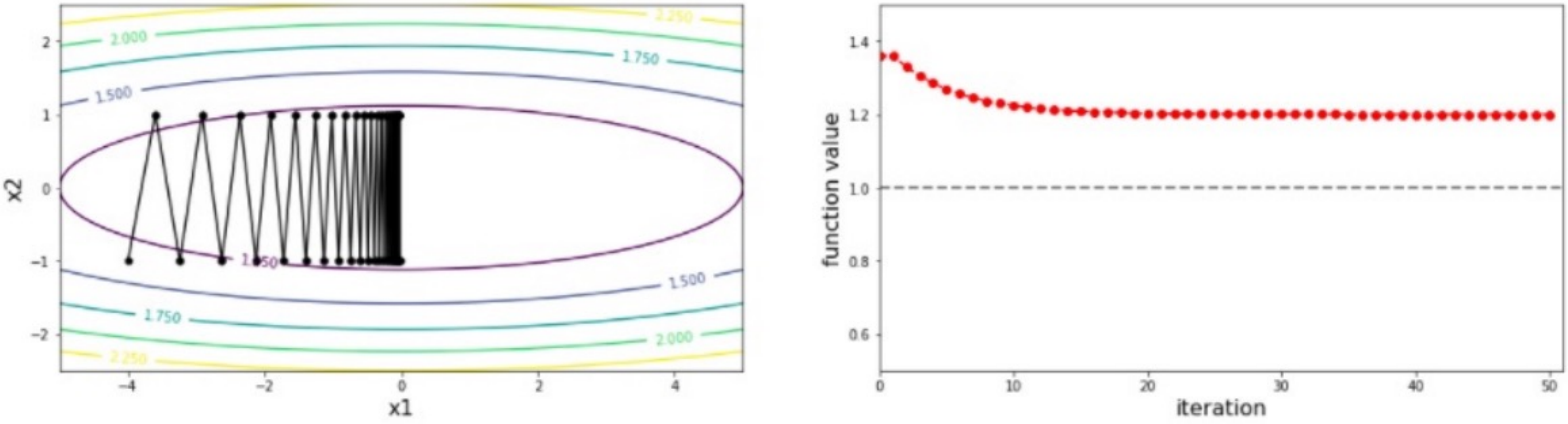
Optimizing Weights – changing λ over time

$\lambda_k = \lambda_0 / (1 + \alpha * (k / n))$



Optimizing Weights – changing λ over time

SGD is *memoryless* – each iteration is unaffected by previous rounds



Adding a memory term is a common approach. *We will not discuss this approach today.*

Choosing a Stopping Time

Set up a **hold-out set** for testing

- Stop when you stop seeing an improvement in your hold-out

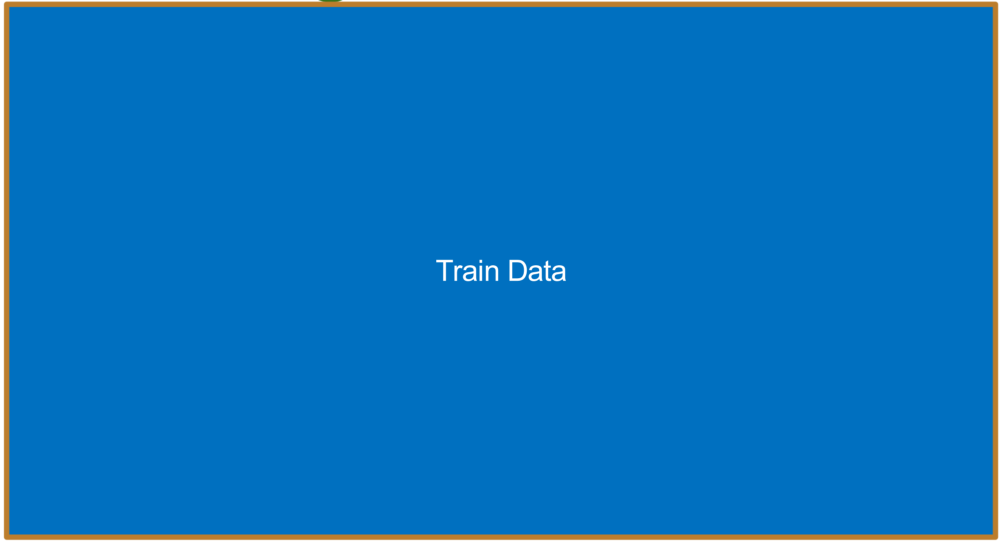
Sidebar – Training, Hold-out, Testing sets

1) Training Set – used in the (usually iterative) training process

2) Hold-Out Set – kept out of the training process, used to meter your training process

3) Testing Set – Used to evaluate the accuracy, etc. of your learning

Training & Test Sets

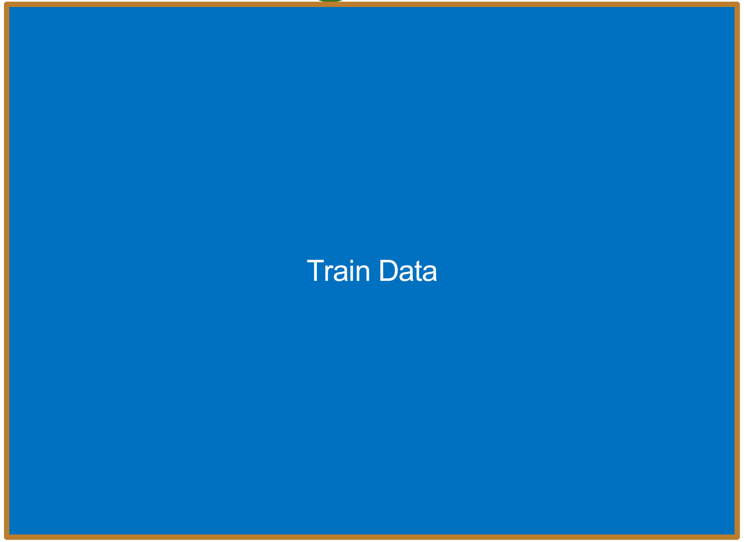


I pull out a random 20% of my data

Now I have something (probably) representative, AND

I'm not just testing inherent bias of my model

Training & Hold-Out & Test Sets



I pull out another random 20% of my data

I'm starting to run out of data… *To Be Continued!*

Choosing a Stopping Time

Set up a **hold-out set** for testing

- Stop when you stop seeing an improvement in your hold-out

Stop when you reach **10 epochs**

Optimizing Weights

Remember our overfitting problem...

Overfitting

Overfitting – A Scenario

- All of my samples are from the "Easy Listening" channel on TotallyNotSpotify
  - How do you think that will affect my data?

other channels underrepresented

calm ~angry

unrepesentative instruments

Overfitting – A Scenario

- All of my samples are from the "Easy Listening" channel on TotallyNotSpotify
  - How do you think that will affect my data?

- Scenario.5: All of my samples are from "Holiday Jams"

happy

holiday vocab

not representative output

Overfitting – B Scenario

- We took our samples from the twitter firehose between 12PM and 4AM in Boulder (because that's when server time was cheaper)!
    - Who might have been tweeting (in English) at that time?
        - HINT: Who was awake at that time?

UG

Australia - Legend

India - Bollywood features

    - What would you guess might happen in your data based on this timing?

Optimizing Weights

Remember our overfitting problem...

We don’t want a weight to spiral out of control due to some oddity in our training data...

Thursday

Course Logistics

- Project Phase 1: Feedback returned
- Project Phase 2: Due 9/30
- Problem Sets: Problem Set 1 Feedback expected Thursday 9/23
  - Currently anticipating a minor delay.
- Problem Set 2: Due 10/7

Model Comparisons

END

Optimizing Weights – Penalty Term

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

Add a penalty term

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

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

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

NOTE: Don't penalize the bias weight w_0

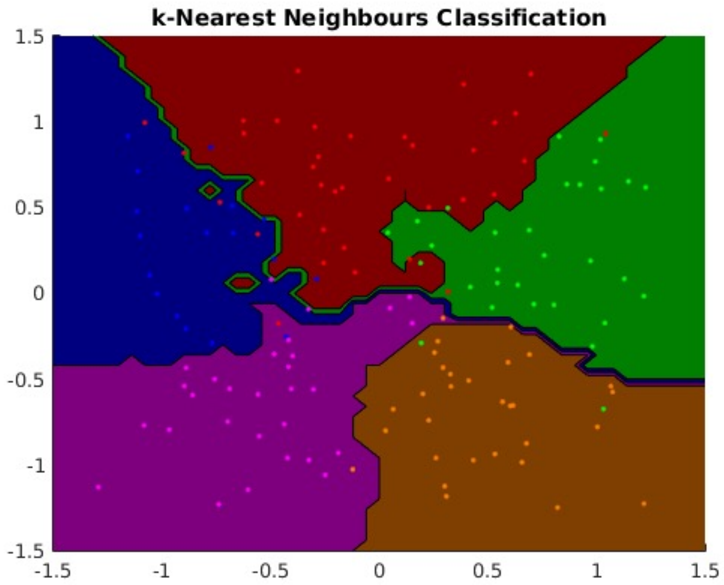
What we've discussed so far...

Maximizing the probability of our weights given the evidence

- Recast as a minimization of the negative log-likelihood
- Using Gradient Descent to tune our weights
- Using a Stochastic (mini-batch) approach to utilize memory
- Adjusting our learning rate

Multi-Class Classification

Multi-Class Classification - KNN



Classification vs. Binary Classification

Some methods are inherently binary

Logistic Regression

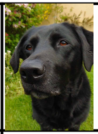
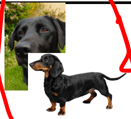
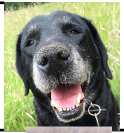
Perceptron

We'll see more as the course develops

Multi-Class: One vs. All / One vs. Rest

For each class, we create a *binary* model

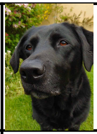
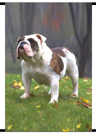
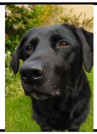
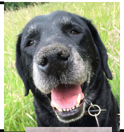
c vs. ~c

	 vs 	 vs 	 vs 
	-	-	+
	+	-	-
	-	+	-

Multi-Class: One vs. Another (All-Pairs)

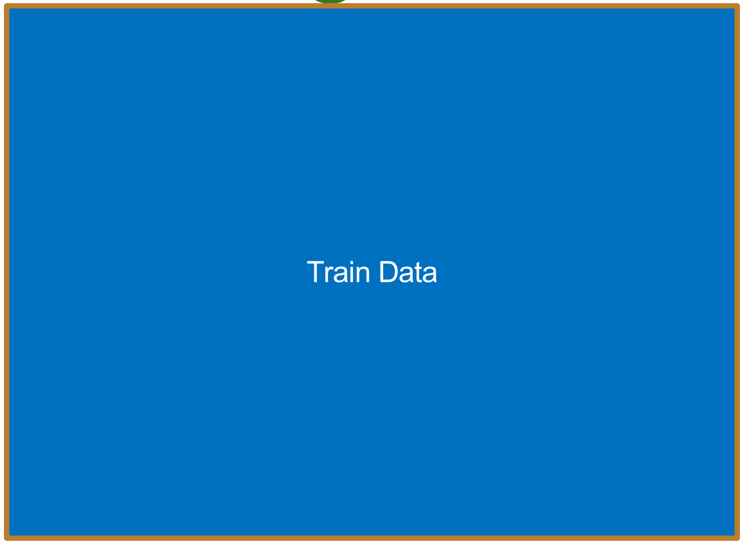
For each pair of classes, we create a *binary* model

1 vs. 2, 1 vs. 3, 2 vs. 3, 1 vs. 4, 2 vs. 4, 3 vs. 4, etc.

	 vs 	 vs 	 vs 
	? 50%	B & 50%	A 90%
	A	A	?
	B	?	B

Evaluating Models

Training & Hold-Out & Test Sets



I pull out another random 20% of my data
I'm starting to run out of data... *Continued!*

What's in a test set?

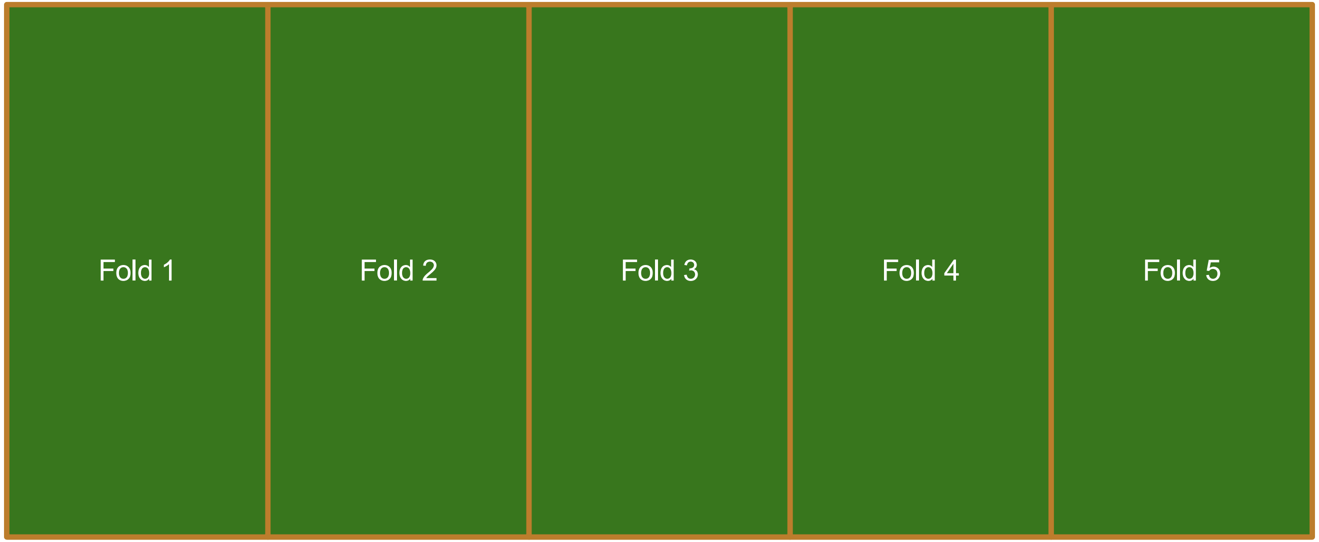
Suddenly, you've got very little data with which to train

K-Fold Cross Validation

Create “folds” of your data, or K equal size groups

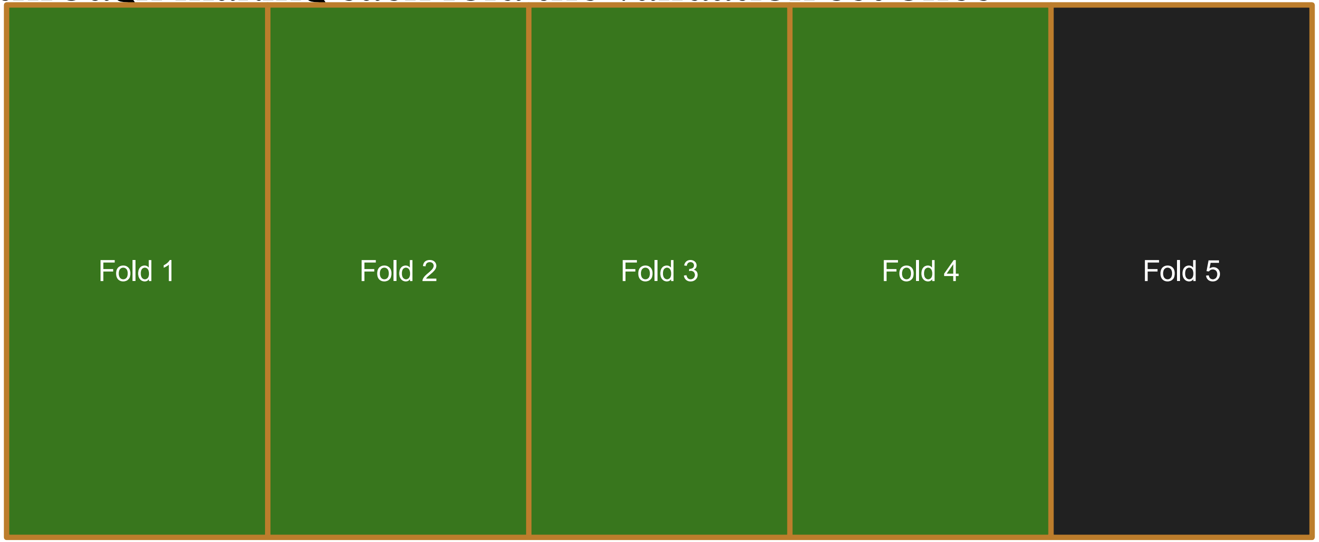
K-Fold Cross Validation

Create “folds” of your data, or K equal size groups



K-Fold Cross Validation

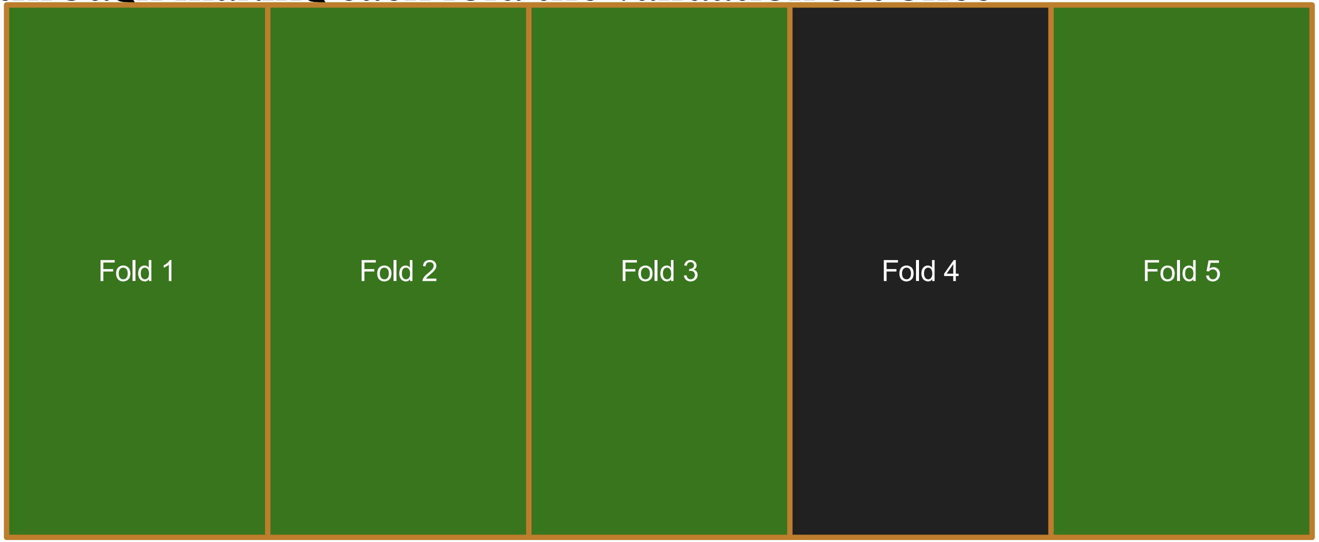
Iterate through making each fold the validation set once



Accuracy = 11 / 20

K-Fold Cross Validation

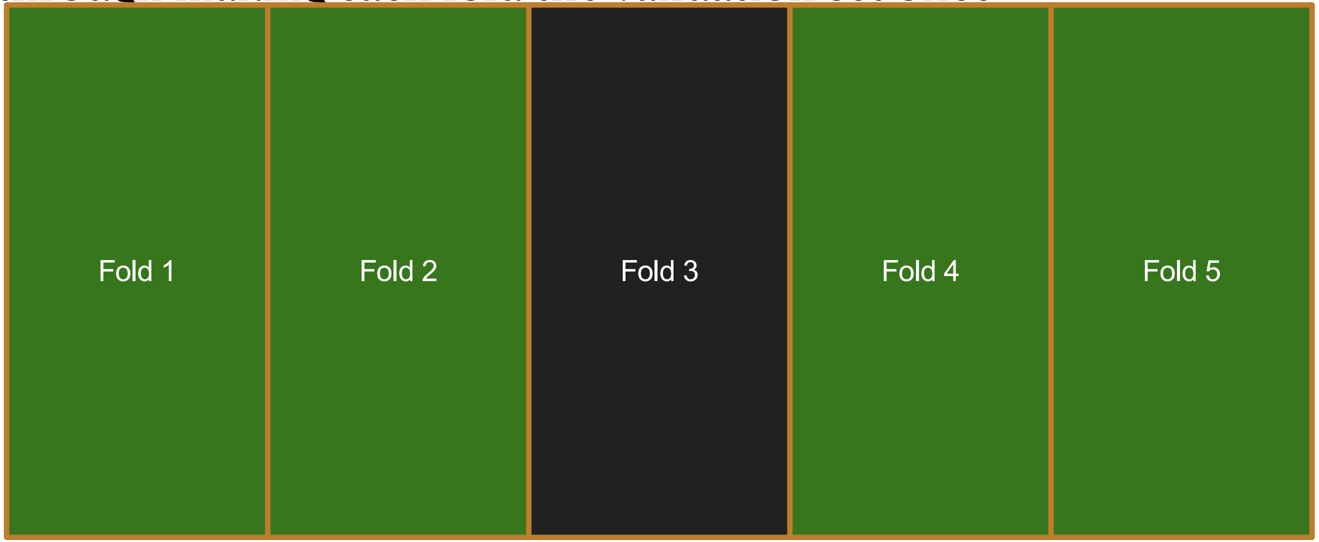
Iterate through making each fold the validation set once



Accuracy = 17 / 20

K-Fold Cross Validation

Iterate through making each fold the validation set once



Accuracy = 16 / 20

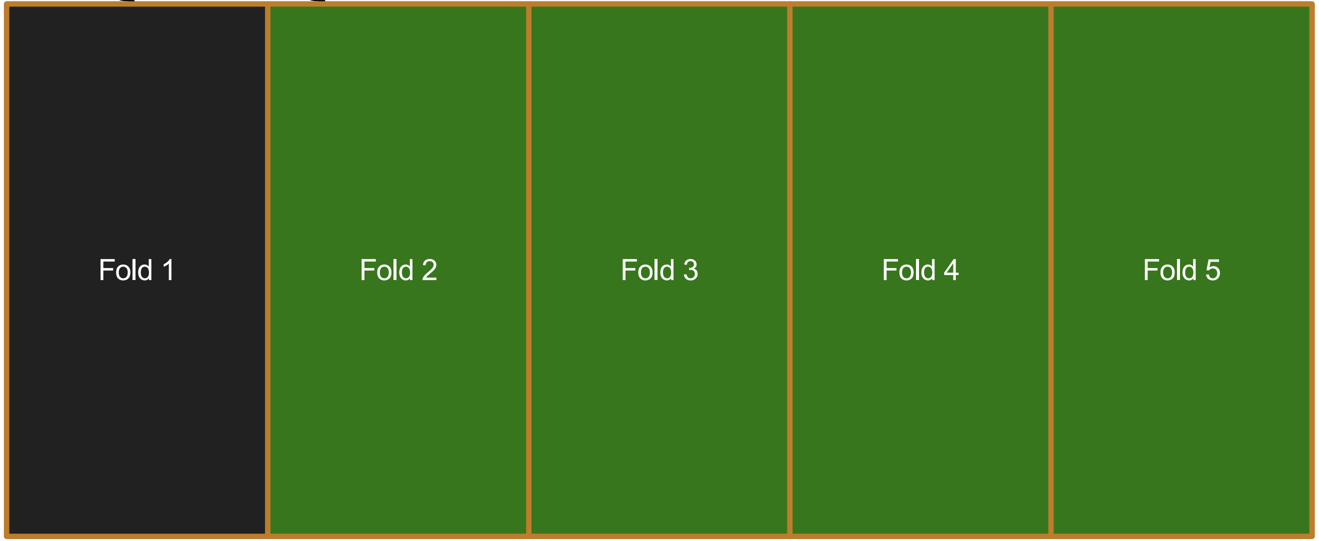
K-Fold Cross Validation

Iterate through making each fold the validation set once

Accuracy = 13 / 20

K-Fold Cross Validation

Iterate through making each fold the validation set once



Accuracy = 16 / 20

K-Fold Cross Validation

Iterate through making each fold the validation set once

Trial	Accuracy
1	11/20
2	17/20
3	16/20
4	13/20
5	16/20

K-Fold Cross Validation

Iterate through making each fold the validation set once

Trial	Accuracy
1	11/20
2	17/20
3	16/20
4	13/20
5	16/20

Take the average of errors for each trial as your expected average

K-Fold Cross Validation

Iterate through making each fold the validation set once

Trial	Accuracy
1	11/20
2	17/20
3	16/20
4	13/20
5	16/20

Take the average of errors for each trial as your expected average

Error = .27, Accuracy = .73

K-Fold Cross Validation

Iterate through making each fold the validation set once

Trial	Accuracy
1	11/20
2	17/20
3	16/20
4	13/20
5	16/20

Take the average of errors for each trial as your expected average

Error = .27, Accuracy = .73

But do we trust this number?

K-Fold Cross Validation

Iterate through making each fold the validation set once

Trial	Accuracy
1	11/20
2	17/20
3	16/20
4	13/20
5	16/20

Take the standard error for our trials, 95% Confidence Interval is approximately +/- 2SE

K-Fold Cross Validation

Iterate through making each fold the validation set once

Trial	Accuracy
1	11/20
2	17/20
3	16/20
4	13/20
5	16/20

Take the standard error for our trials, 95% Confidence Interval is approximately +/- 2SE

2SE = .1, confidence interval = [.17, .37]

K in K-Fold Cross Validation

There's no hard-and-fast rule, there's no universally accepted solution

K = 5 or K = 10 are popular, but it also depends on your dataset size

People will do multiple rotations / *bootstrapping* as well.

What is an Error?

We’ve looked at trying it out on a test set and getting an “accuracy”

Accuracy = # correct / # total

1) What is our “test set”?
2) Are all mistakes created equal?

Feature Engineering

Problems With Data

Categorical Data

Categorical Data – Ways to Encode

Match to a corresponding numeric value

Color → R, G, B values (0 – 256)

Categorical Data – Ways to Encode

Match to a corresponding numeric value

Color → R, G, B values (0 – 256)

Will this work for every type of categorical data?

- choosing a piece of furniture to recommend based on room

Categorical Data – Ways to Encode

Create a new numeric value for each category

White	1
Brown	2
Gold	3
Red	4

Categorical Data – Ways to Encode

Create a new numeric value for each category

White	1
Brown	2
Gold	3
Red	4

Problems?

Interlude – Types of Data

Continuous Numeric

- Maps to a “real” number
- Distances between numbers have meaning

Discrete Numeric

- Maps to a discrete, infinite set (e.g. integers)
- Distances between numbers have meaning

Ordinal

- Entries have an *order*, but distances lose meaning

Categorical

- Entries are each unique, but don’t have further restrictions

Categorical Data – Ways to Encode

Create a new binary feature (*dummy* variable) for each category

- one-hot encoding

Color	**isWhite**	**isBrown**	**isGolden**	**isRed**
White	1	0	0	0
Golden	0	0	1	0
Brown	0	1	0	0
Red	0	0	0	1

“Categorical” Data – Working with Time

Person	Time	isWorking
David	9/21/2021 3PM	1
Ayouk	9/23/2021 3PM	0
David	9/25/2021 12PM	0
David	9/27/2021 12PM	1

“Categorical” Data – Working with Time

So many representations of time!

- 02/06/2019 3:40PM MST
- 06/02/2019 15:40 MST
- 201906021540
- 2019/06/02 22:40 GMT
- 1549492800
- 1549492800000

Working with Time

Time in Unix Timestamp

- Numerical

Working with Time

Person	Time	Timestamp	isWorking
David	9/21/2021 3PM	X	1
Ayouk	9/23/2021 3PM	X	0
David	9/25/2021 12PM	X – Y	0
David	9/27/2021 12PM	X – Z	1

Working with Time

Breaking into multiple features

- e.g. “year”, “month”, “Day of Week”, “Date”, “Time (s)”
- Still some sense of numeric differences
- Allows you to capture cyclical events (seasons, holidays, ...)

Working with Time

Person	Time	Timestamp	Month	Day of Week	Date	Time	isWorking
David	9/21/2021 3PM	X	9	1	21	15	1
Ayouk	9/23/2021 3PM	X	9	4	23	15	0
David	9/25/2021 12PM	X – Y	9	6	25	12	0
David	9/27/2021 12PM	X – Z	9	1	27	12	1

Numeric Data

Normalization

Sq Ft	**# Bedrooms**	**House Lat.**	**House Long.**	***Cost***
1600	4	121.33	47.34	*500K*
1250	3	121.33	55.23	*450K*
750	2	121.33	55.34	*200K*
1150	2	130.99	47.34	*1500K*

Numeric Data

Normalization – Min-max Scaling

Sq Ft	**# Bedrooms**	**House Lat.**	**House Long.**	***Cost***
1600	4	121.33	47.34	*500K*
1250	3	121.33	55.23	*450K*
750	2	121.33	55.34	*200K*
1150	2	130.99	47.34	*1500K*

Numeric Data

Normalization – Log-Transform

Log Sq Ft	**# Bedrooms**	**House Lat.**	**House Long.**	***Cost***
3.2	4	121.33	47.34	*500K*
3.09	3	121.33	55.23	*450K*
2.87	2	121.33	55.34	*200K*
3.06	2	130.99	47.34	*1500K*

Numeric Data

Normalization – Box-Cox Transform

$$\frac{X^\lambda - 1}{\lambda}$$

Log Sq Ft	**# Bedrooms**	**House Lat.**	**House Long.**	***Cost***
3.2	4	121.33	47.34	*500K*
3.09	3	121.33	55.23	*450K*
2.87	2	121.33	55.34	*200K*
3.06	2	130.99	47.34	*1500K*

Find λ using a likelihood function

Numeric Data

City 1 Lat.	**City 1 Long.**	**City 2 Lat.**	**City 2 Long.**	***Drivable?***
123.24	46.71	121.33	47.34	*Yes*
123.24	56.91	121.33	55.23	*Yes*
123.24	46.71	121.33	55.34	*No*
123.24	46.71	130.99	47.34	*No*

Combining Numeric Variables

City 1 Long.	**City 1 Lat.**	**City 2 Long.**	**City 2 Lat.**	**City Long. Diff.**	**City Lat. Diff.**	***Drivable?***
123.24	46.71	121.33	47.34	~2.0	~0.5	*Yes*
123.24	56.91	121.33	55.23	~2.0	~1.5	*Yes*
123.24	46.71	121.33	55.34	~2.0	~9.5	*No*
123.24	46.71	130.99	47.34	~9.0	~0.5	*No*

Numeric Data

City 1 Long.	**City 1 Lat.**	**City 2 Long.**	**City 2 Lat.**	***Drivable?***
123.24	46.71	121.33	47.34	*Yes*
123.24	56.91	121.33	55.23	*Yes*
123.24	46.71	121.33	55.34	*No*
123.24	46.71	130.99	47.34	*No*
-90	89.9	90	89.9	*Yes*

Numeric Data

City 1 Long.	**City 1 Lat.**	**City 2 Long.**	**City 2 Lat.**	***Drivable?***
123.24	46.71	121.33	47.34	*Yes*
123.24	56.91	121.33	55.23	*Yes*
123.24	46.71	121.33	55.34	*No*
123.24	46.71	130.99	47.34	*No*
-90	89.9	90	89.9	*Yes*
179	-17.7	-179	-17.7	*Yes*

Numeric Data - Binarization

Genre	**Artist**	**Plays**
Pop	Rihanna	1056219
Pop	Dr. Quigley	0
R&B	Tupac	11234
Jazz	Dave Brubeck	183

Numeric Data - Rounding

Genre	Artist	Plays
Pop	Rihanna	1056219
Pop	Dr. Quigley	0
R&B	Tupac	11234
Jazz	Dave Brubeck	183

Text Data

That album was great! I loved it the best out of all my music collection.
Others think it was cool, but I’m tired of Rihanna, I thought the album was bad.

Text Data – Tokenization

That album was great! I loved it the best out of all my music collection.
Others think it was cool, but I’m tired of Rihanna, I thought the album was bad.

Text Data – Tokenization

Natural Language Toolkit – nltk.org

Positive	Negative	Negative	Negative	Positive
Great	Cool	Worst	Depressing	Cool
Love	Tired	Sad	Worst	Amazing
Best	Bad	Angry	No	Sweet

Text Data - Featurization

Linguists, etc. have worked to group words under various categories

- Part of speech, sentiment, others

Leverage details *about* the words as features

Positive	**Negative**	**Negative**	**Negative**	**Positive**
Great	Cool	Worst	Depressin g	Cool
Love	Tired	Sad	Worst	Amazing
Best	Bad	Angry	No	Sweet

Text Data – Sentiment Tagging

Tokenize our sentences

Look up the features for each token

Use these as new features in a classifier

- *There's a whole field around developing databases of features*

Positive	Negative	Negative	Negative	Positive
Great	Cool	Worst	Depressing	Cool
Love	Tired	Sad	Worst	Amazing
Best	Bad	Angry	No	Sweet

http://www.nltk.org/howto/sentiment.html

Sequential Data

I have a series of inputs that I want to combine to do classification

- We have a series of locations, can we gauge trajectory?
- We have 100 readings from an accelerometer, can we get gait?
- We have recent weather readings, can we better predict it?

Sequential Data

I have a series of inputs that I want to combine to do classification

- We have a series of locations, can we gauge trajectory?
- We have 100 readings from an accelerometer, can we get gait?
- We have recent weather readings, can we better predict it?

For those who need advanced techniques quickly for their projects, consider one of the textbooks for the class, Bayesian Reasoning & Machine Learning Ch. 23 - 26

Sequential Data

END

Timestamp	**User**	***Action***
100	David	*Copy*
150	David	*Paste*
200	David	*Copy*
250	David	*Paste*