LECTURE 18

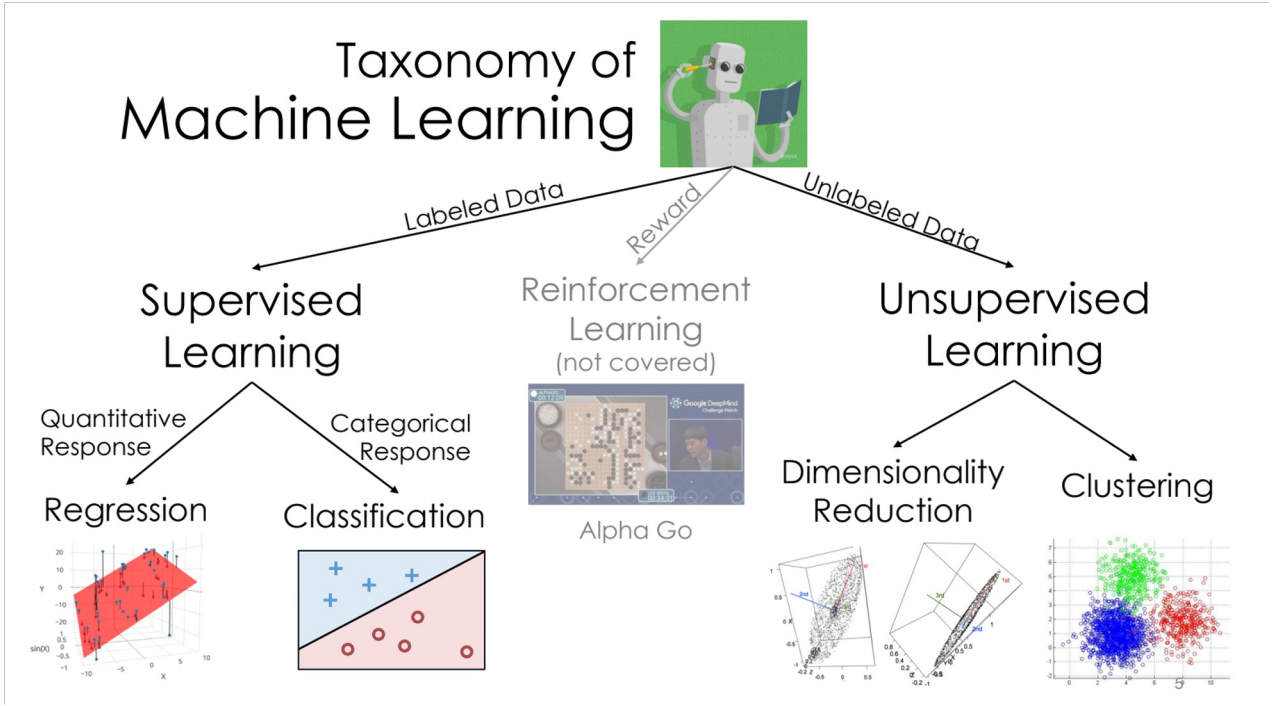
Logistic Regression

Moving from regression to classification.

Machine learning taxonomy

Regression and Classification are both forms of **supervised learning**.

Logistic regression, the topic of this lecture, is mostly used for **classification**, even though it has “regression” in the name.



Regression vs Classification

Regression vs Classification

Logistic regression model derivation

- logistic function (sigmoid)
- Parameter interpretation

Loss function

- Pitfalls of Squared Loss
- Cross Entropy

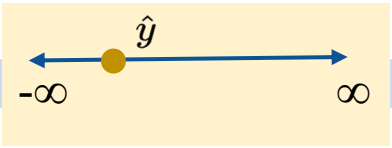
Maximum likelihood estimation

So Far: Regression

In regression, we use unbounded numeric features to predict an *unbounded numeric output*.

GAME_ID	TEAM_NAME	MATCHUP	REB	FTM	TOV	GOAL_DIFF	WON
21700001	Boston Celtics	BOS @ CLE	46	19	12	-0.049	0
21700002	Golden State Warriors	GSW vs. HOU	41	19	17	0.053	0
21700003	Charlotte Hornets	CHA @ DET	47	23	17	-0.030	0
21700004	Indiana Pacers	IND vs. BKN	47	25	14	0.041	1
21700005	Orlando Magic	ORL vs. MIA	50	22	15	0.042	1

$x^\top \theta$



Input: numeric features

Model: linear combination

Output: numeric prediction

Examples:

- Predict goal difference from turnover %
- Predict tip from total bill
- Predict mpg from hp

Now: Classification

In **classification**, we use unbounded numeric features to predict a *categorical class*.

Examples:

- Predict *which team won* from turnover %
- Predict *day of week* from total bill
- Predict *model of car* from hp

GAME_ID	TEAM_NAME	MATCHUP	REB	FTM	TOV	GOAL_DIFF	WON
21700001	Boston Celtics	BOS @ CLE	46	19	12	-0.049	0
21700002	Golden State Warriors	GSW vs. HOU	41	19	17	0.053	0
21700003	Charlotte Hornets	CHA @ DET	47	23	17	-0.030	0
21700004	Indiana Pacers	IND vs. BKN	47	25	14	0.041	1
21700005	Orlando Magic	ORL vs. MIA	50	22	15	0.042	1

Input: numeric features

$$p = \sigma\left(x^\top \theta\right)$$

Model: linear combination transformed by non-linear **sigmoid**

Win?
If p > 0.5: predict a win
Other: predict a loss

Decision rule

Output: **class**

An aside: we will use logistic "regression" to perform a *classification* task. Here, "regression" refers to the type of model, not the task being performed.

Kinds of Classification

We are interested in predicting some **categorical variable**, or **response**, y.

Binary classification

- Two classes
- **Responses** y are either 0 or 1

win or lose

disease or no disease

spam or ham

Multiclass classification

- Many classes
- Examples: Image labeling (Pishi, Thor, Hera), next word in a sentence, etc.

Structured prediction tasks

- Multiple related classification predictions
- Examples: Translation, voice recognition, etc.

Our new goal: predict a **binary** output (y_hat = 0 or y_hat = 1) given inputted numeric features

The Modeling Process

	Regression ($y \in \mathbb{R}$)	**Classification** ($y \in \{0, 1\}$)
1. Choose a model	Linear Regression $\hat{y} = f_\theta(x) = x^T\theta$	??
2. Choose a loss function	Squared Loss or Absolute Loss	??
3. Fit the model	Regularization Sklearn/Gradient descent	Regularization Sklearn/Gradient descent
4. Evaluate model performance	R^2, Residuals, etc.	?? (next lecture)

Logistic Regression model derivation

Regression vs Classification

Logistic regression model derivation

- logistic function (sigmoid)
- Parameter interpretation

Loss function

- Pitfalls of Squared Loss
- Cross Entropy

Maximum likelihood estimation

Example dataset

In this lecture, we will primarily use data from the 2017-18 NBA season.

Goal: Predict whether or not a team will win, given their FG_PCT_DIFF.

- This is the difference in field goal percentage between the two teams.
- Positive FG_PCT_DIFF: team made more shots than the opposing team.

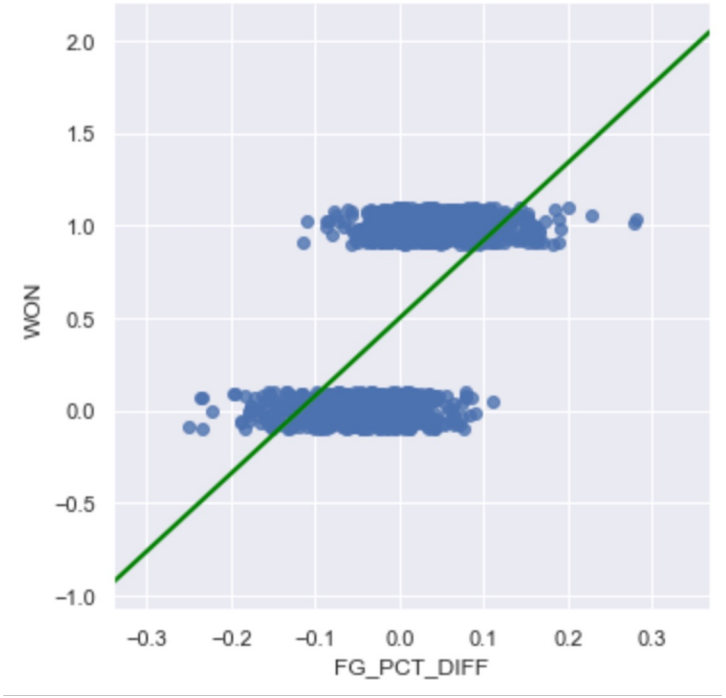
TEAM_NAME	MATCHUP	WON	FG_PCT_DIFF
Boston Celtics	BOS @ CLE	0	-0.049
Golden State Warriors	GSW vs. HOU	0	0.053
Charlotte Hornets	CHA @ DET	0	-0.030
Indiana Pacers	IND vs. BKN	1	0.041
Orlando Magic	ORL vs. MIA	1	0.042

1s represent wins, 0s represent losses.

Why not use Ordinary Least Squares?

We already have a model that can predict any quantitative response. Why not use it here?

- The output can be outside of the range [0, 1]. What does a predicted WON value of -2 mean?
- Very sensitive to outliers/ imbalanced data.
- Many other statistical reasons.

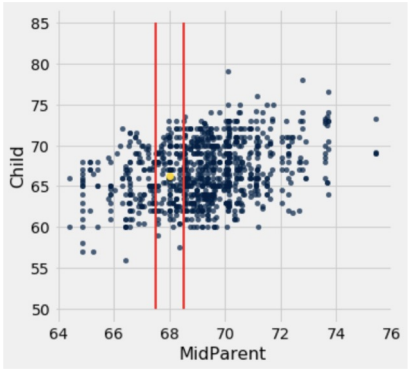


Graph of Averages

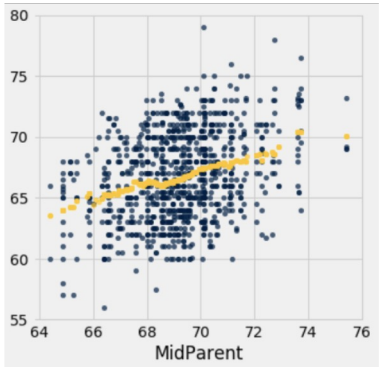
Clearly, least squares regression won't work here. We need to start fresh.

Let's revisit the linear regression by considering the **graph of averages**.

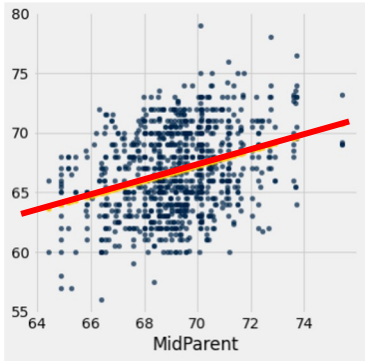
> For an input x, compute the **average value of y for all nearby x**, and predict that.



Bucket x-axis into bins



Average y vs. x bins is approximately linear



Parametric linear model

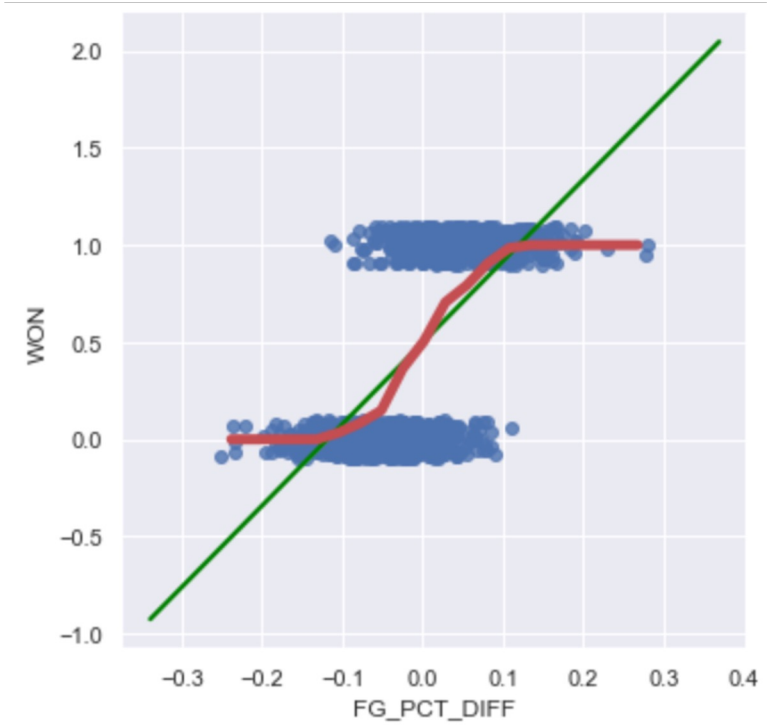
$$\hat{y} = x^T\theta$$

Graph of averages

If we **bin the x-axis**, and take the **average** y-value for each bin, and tried to model that.

Doing so here yields a curve that resembles an s.

- Since our true y is either 0 or 1, this curve models the **probability that WON = 1**, given FG_PCT_DIFF.
  - WON = 1 means “belong to class 1”.
- **Our goal is to model this red curve as best as possible**.



Log-odds of probability is roughly linear

we noticed that the **log-odds of the probability of belonging to class 1 was linear**. **This is the assumption that logistic regression is based on**.

$$\text{odds}(p) = \frac{p}{1-p} \qquad \text{log-odds}(p) = \log\left(\frac{p}{1-p}\right)$$

For now, let's let t denote our linear function (since log-odds is linear). Solving for p:

$$t = \log\left(\frac{p}{1-p}\right)$$

$$e^t = \frac{p}{1-p}$$

$$e^t - pe^t = p$$

$$p = \frac{e^t}{1+e^t} = \frac{1}{1+e^{-t}}$$

With logistic regression, we are always referring to log base e ("ln").

Log-odds of probability is roughly linear

we noticed that the **log-odds of the probability of belonging to class 1 was linear**. **This is the assumption that logistic regression is based on**.

$$\text{odds}(p) = \frac{p}{1-p} \qquad \text{log-odds}(p) = \log\left(\frac{p}{1-p}\right)$$

For now, let's let t denote our linear function (since log-odds is linear). Solving for p:

$$t = \log\left(\frac{p}{1-p}\right)$$

$$e^t = \frac{p}{1-p}$$

$$e^t - pe^t = p$$

$$p = \frac{e^t}{1+e^t} = \frac{1}{1+e^{-t}}$$

This is called the **logistic function**, σ(t).

Arriving at the logistic regression model

We know how to model linear functions quite well.

- We can substitute $t = x^T\theta$, since t was just a placeholder.

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

p represents the probability of belonging to class 1.

- We are modeling $P(Y=1|x)$.

Putting this all together:

$$P(Y=1|x) = \frac{1}{1+e^{-x^T\theta}} = \sigma(x^T\theta)$$

Looks just like the linear regression model, with a $\boldsymbol{\sigma}(\,)$ wrapped around it.
We call logistic regression a **generalized linear model**, since it is a non-linear transformation of a linear model.

Logistic Regression

1. Choose a model

2. Choose a loss function

3. Fit the model

4. Evaluate model performance

Logistic Regression $\hat{y} = f_\theta(x) = P(Y = 1|x) = \sigma(x^T\theta)$

Linear vs. logistic regression

In a **linear regression** model, we predict a **quantitative** variable (i.e., some real number) as a linear function of features.

- Our output, or **response**, y, could be any real number.

$$\hat{y} = f_\theta(x) = x^T\theta$$

In a **logistic regression** model, our goal is to predict a binary **categorical** variable (class 0 or class 1) as a linear function of features, passed through the logistic function.

- Our **response** is the probability that our observation belongs to class 1.
- Haven't yet done classification!

$$\hat{y} = f_\theta(x) = P(Y = 1|x) = \sigma(x^T\theta)$$

Logistic Function

Regression vs Classification

Logistic regression model derivation

- **logistic function (sigmoid)**
- Parameter interpretation

Loss function

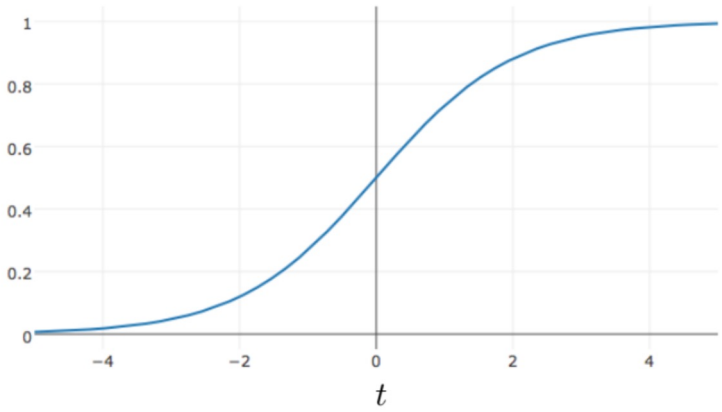
- Pitfalls of Squared Loss
- Cross Entropy

Maximum likelihood estimation

Properties of the logistic function

The logistic function is a type of **sigmoid**, a class of functions that share certain properties.

$$\sigma(t) = \frac{1}{1 + e^{-t}} \qquad -\infty < t < \infty$$



- Its output is bounded between 0 and 1, no matter how large t is.
  - Fixes an issue with using linear regression to predict probabilities.
- We can interpret it as mapping real numbers to probabilities.

Properties of the logistic function

Definition

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

Range

$$0 < \sigma(t) < 1$$

Inverse

$$t = \sigma^{-1}(p) = \log\left(\frac{p}{1-p}\right)$$

Reflection and Symmetry

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

Derivative

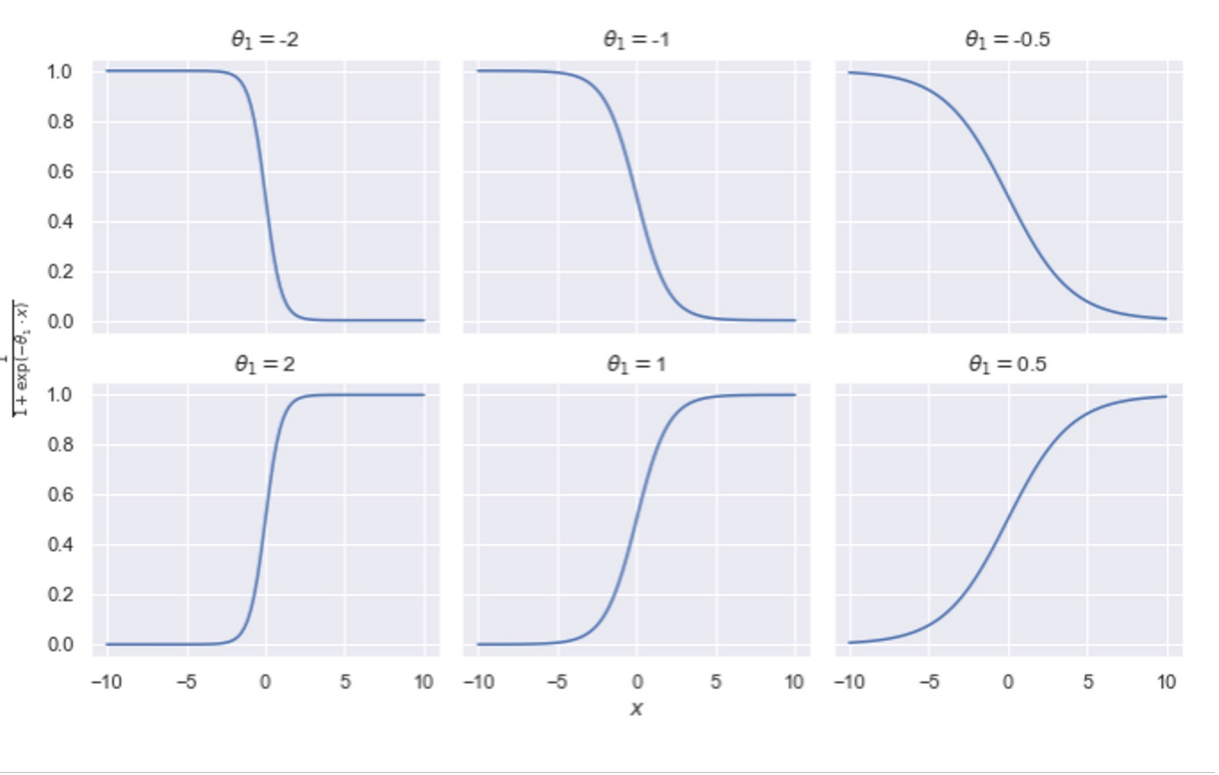
$$\frac{d}{dt}\sigma(t) = \sigma(t)(1-\sigma(t)) = \sigma(t)\sigma(-t)$$

Shape of the logistic function

Consider the plot of $\sigma(\theta_1 x)$, for several different values of θ_1.

- If θ_1 is positive, the curve increases to the right.
- The further θ_1 is from 0, the steeper the curve.

In the notebook, we explore more sophisticated logistic curves.



Parameter interpretation

Regression vs Classification

Logistic regression model derivation

- logistic function (sigmoid)
- **Parameter interpretation**

Loss function

- Pitfalls of Squared Loss
- Cross Entropy

Maximum likelihood estimation

Parameter interpretation

Recall, we arrived at the model by assuming that the **log-odds of the probability of belonging to class 1 was linear.**

$$P(Y=1|x) = \sigma(x^T\theta) \quad \Leftarrow \quad \log\left(\frac{P(Y=1|x)}{P(Y=0|x)}\right) = x^T\theta \quad \Leftarrow \quad \frac{P(Y=1|x)}{P(Y=0|x)} = e^{x^T\theta}$$

This is the same as $\frac{p}{1-p}$ because

$$P(Y=1|x) + P(Y=0|x) = 1$$

(Remember, we are dealing with binary classification – we are predicting 1 or 0.)

Parameter interpretation

Let's suppose our linear component has just a single feature, along with an intercept term.

$$\frac{P(Y=1|x)}{P(Y=0|x)} = e^{\theta_0+\theta_1 x}$$

What happens if you increase x by one unit?

- Odds is multiplied by e^{θ_1}.
- If $\theta_1 > 0$, the odds increase.
- If $\theta_1 < 0$, the odds decrease.

What happens if $x^T\theta = \theta_0 + \theta_1 x = 0$?

- This means class 1 and class 0 are equally likely.
- $$e^0 = 1 \implies \frac{P(Y=1|x)}{P(Y=0|x)} = 1 \implies P(Y=1|x) = P(Y=0|x)$$

The odds ratio can be interpreted as the "number of successes for each failure."

Loss Function

Regression vs Classification

Logistic regression model derivation

- logistic function (sigmoid)
- Parameter interpretation

Loss Function

- Pitfalls of Squared Loss
- Cross Entropy

Maximum likelihood estimation

Logistic Regression with squared loss?

1. Choose a model

Logistic Regression $\hat{y} = f_\theta(x) = P(Y = 1|x) = \sigma(x^T\theta)$

2. Choose a loss function

$$R(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \sigma(\mathbb{X}_i^T\theta))^2$$?

3. Fit the model

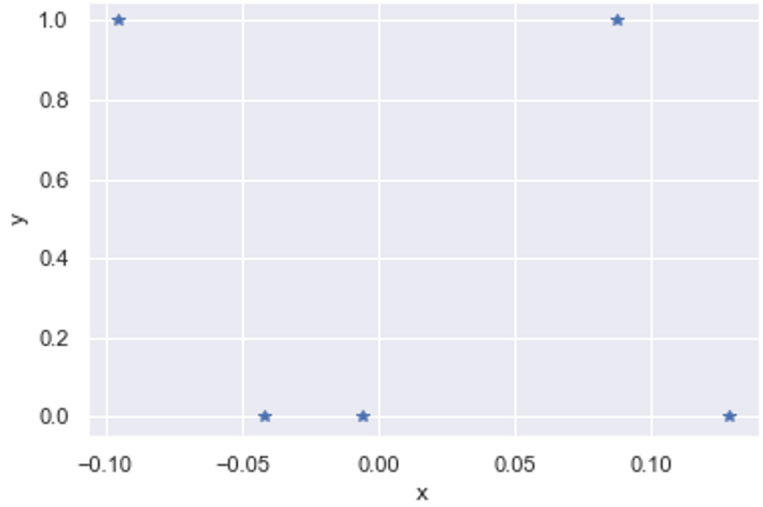
4. Evaluate model performance

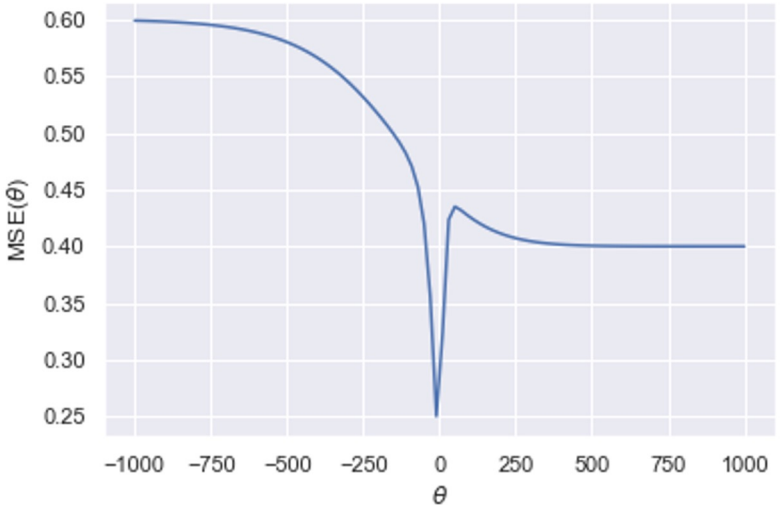
Pitfalls of squared loss with logistic regression

On the left, we have a toy dataset (i.e. we've plotted the original data, y vs. x).
On the right, we have a plot of the mean squared error of this dataset when fitting a single-feature logistic regression model, for different values of θ (i.e. the loss surface).

What is the issue?

$$\text{MSE}(\theta) = \frac{1}{5}\sum_{i=1}^{5}(y_i - \sigma(\theta x_i))^2$$

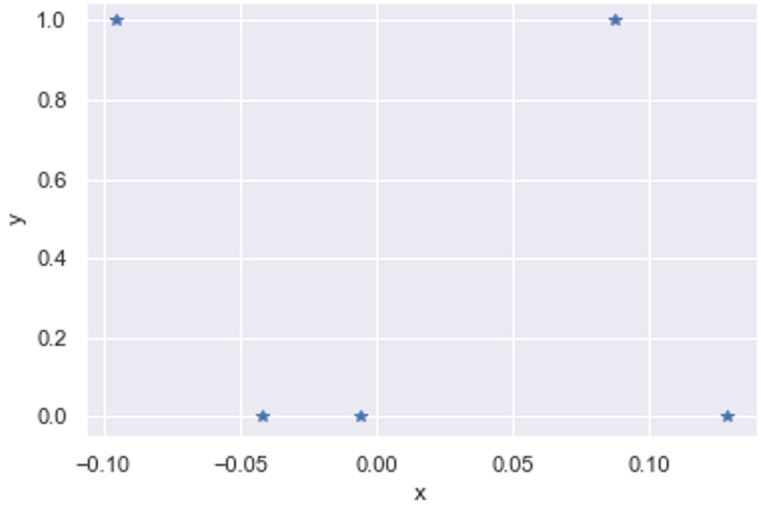


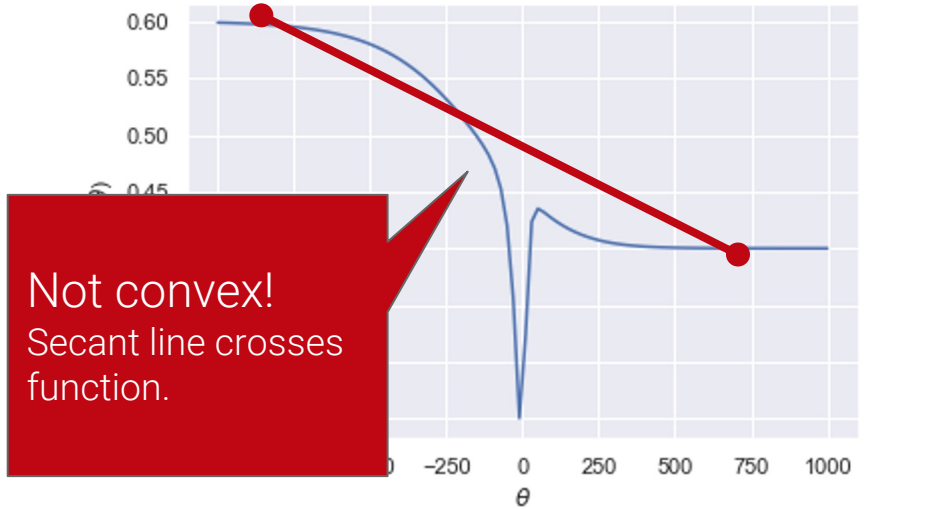


Pitfalls of squared loss with logistic regression

On the left, we have a toy dataset (i.e. we've plotted the original data, y vs. x).
On the right, we have a plot of the mean squared error of this dataset when fitting a single-feature logistic regression model, for different values of θ (i.e. the loss surface).

$$\text{MSE}(\theta) = \frac{1}{5}\sum_{i=1}^{5}(y_i - \sigma(\theta x_i))^2$$





Pitfalls of squared loss with logistic regression

For this particular loss surface, different initial guesses for thetahat yield different “optimal values”, as per scipy.optimize.minimize:

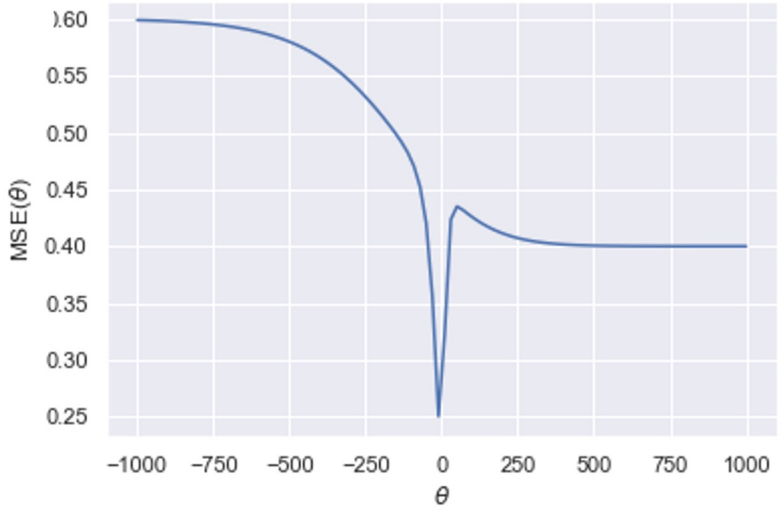
```
1  minimize(mse_loss_single_arg_toy, x0 = 0)["x"][0]
```

```
-4.801981341432673
```

```
1  minimize(mse_loss_single_arg_toy, x0 = 500)["x"][0]
```

```
500.0
```

This loss surface is not convex. We’d like it to be.



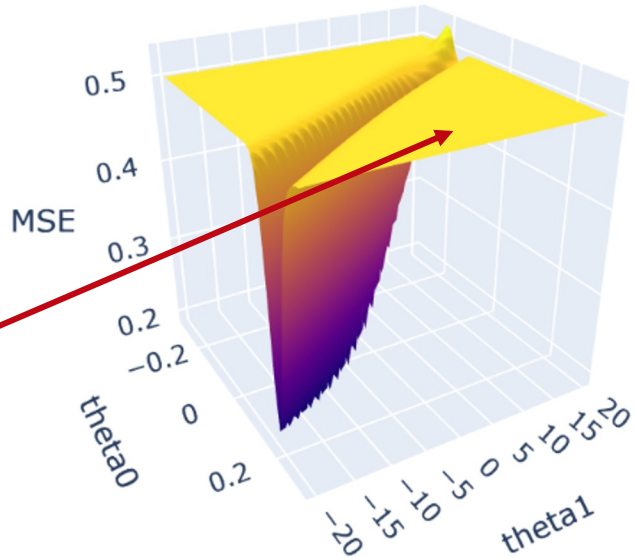
Pitfalls of squared loss with logistic regression

The loss surface of MSE for a logistic regression model with a single slope plus an intercept often looks something like this.

If your initial guess for $\hat{\theta}$ is way out in the flat yellow region, routine can get stuck. The model gets stuck in local minima

If the gradient is 0, your update rule will stop changing.

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla_\theta R(\theta, \mathbb{X}, \mathbb{Y})$$



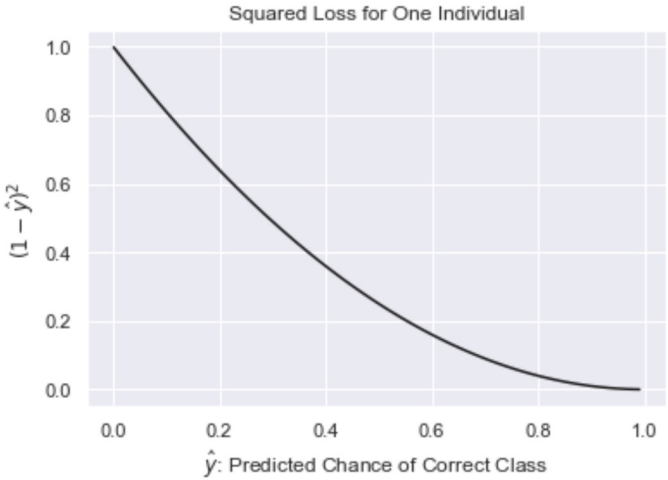
Pitfalls of squared loss with logistic regression

Another issue: since y_i is either 0 or 1, and $\hat{y}_i$ is between 0 and 1, $(y_i - \hat{y}_i)^2$ is also bounded between 0 and 1.

- Even if our probability is nowhere close, the loss isn't that large in magnitude.
  - If we say the probability is 10^-6, but it happens anyway, error should be large.
- We want to penalize wrong answers significantly.

Suppose the observation we're trying to predict is actually in **class 1**.

On the right, we have a plot of $(1 - \hat{y})^2$ vs $\hat{y}$.
This is the squared loss for a single prediction.



Summary of issues with squared loss and logistic regression

While it can work, squared loss is not the best choice of loss function for logistic regression.

- Average squared loss is not convex.
  - Numerical methods will struggle to find a solution.
- Wrong predictions aren't penalized significantly enough.
  - Squared loss (and hence, average squared loss) are bounded between 0 and 1.

Fortunately, there's a solution.

Loss in Classification

Let y be a binary label {0, 1}, and p be the model's predicted probability of the label being 1.

In a classification task, how do we want our loss function to behave?

- When the true y is 1, we should incur *low* loss when the model predicts large p.
- When the true y is 0, we should incur *high* loss when the model predicts large p.

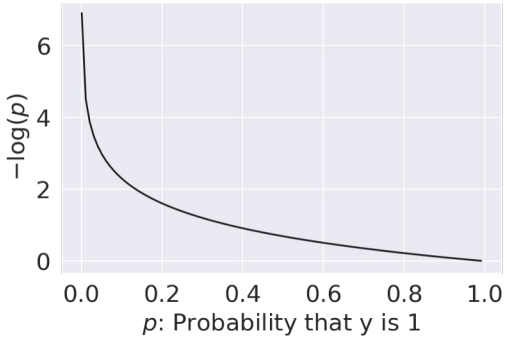
In other words, the behavior we need from our loss function depends on the value of the true class, y.

Cross-Entropy Loss

Let y be a binary label {0, 1}, and p be the probability of the label being 1.

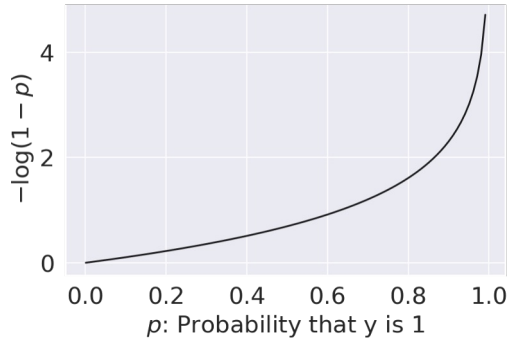
The **cross-entropy loss** is defined as:

$$\text{CE loss} = \begin{cases} -\log(p), & \text{if } y = 1 \\ -\log(1-p), & \text{if } y = 0 \end{cases}$$



For y = 1,

- p → 0: ∞ loss
- p → 1: zero loss



For y = 0,

- p → 0: zero loss
- p → 1: ∞ loss

Cross-Entropy Loss: Two Loss Functions In One!

The piecewise loss function we introduced just then is difficult to optimize – we don't want to check "which" loss to use at each step of optimizing theta.

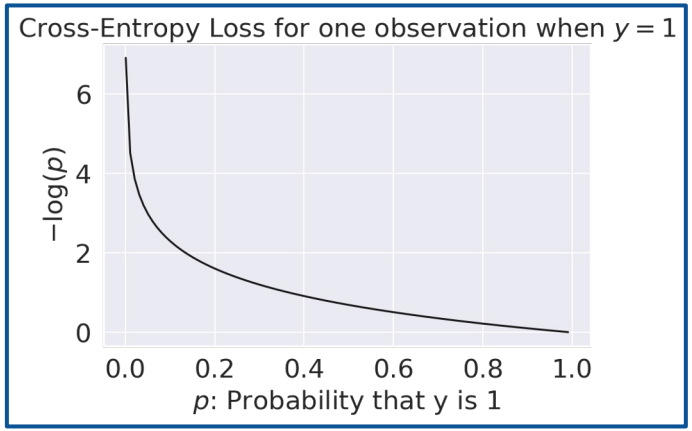
Cross-entropy loss can be equivalently expressed as:

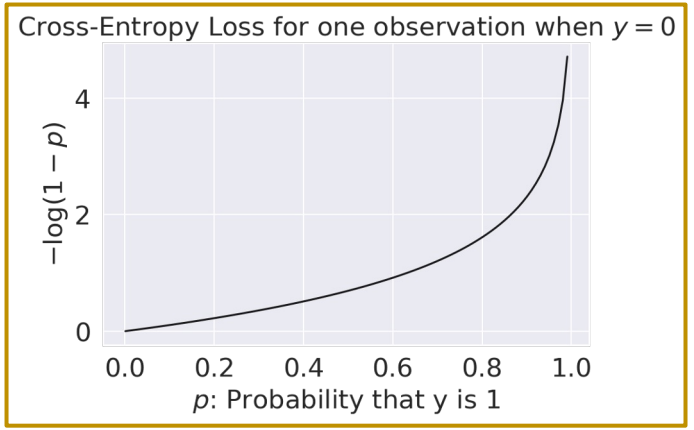
$$\text{CE loss} = \begin{cases} -\log(p), & \text{if } y = 1 \\ -\log(1-p), & \text{if } y = 0 \end{cases} \rightarrow -\left(y \log(p) + (1-y)\log(1-p)\right)$$

makes loss positive

for y = **1**, only this term stays

for y = **0**, only this term stays





Empirical Risk: Average Cross-Entropy Loss

For a single datapoint, the cross-entropy curve is convex. It has a global minimum.

$$-\left(y\log(p)+(1-y)\log(1-p)\right)$$

What about average cross-entropy loss, i.e., empirical risk?

For logistic regression, the empirical risk over a sample of size n is:

$$R(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\log(p_i)+(1-y_i)\log(1-p_i)\right)$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\log\left(\sigma\left(X_i^T\theta\right)\right)-(1-y_i)\log\left(1-\sigma\left(X_i^T\theta\right)\right)\right)$$

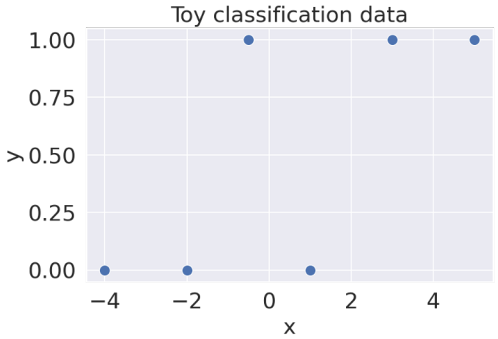
[Recall our model is $\hat{y}_i = p_i = \sigma\left(X_i^T\theta\right)$]

The optimization problem is therefore to find the estimate $\hat{\theta}$ that minimizes R(θ):

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \; -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\log\left(\sigma\left(X_i^T\theta\right)\right)-(1-y_i)\log\left(1-\sigma\left(X_i^T\theta\right)\right)\right)$$

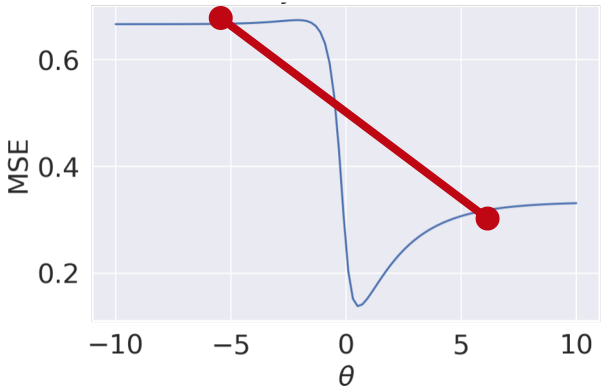
Convexity Proof By Picture

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \; -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log\left(\sigma\left(X_i^T\theta\right)\right) - (1-y_i)\log\left(1-\sigma\left(X_i^T\theta\right)\right)\right)$$



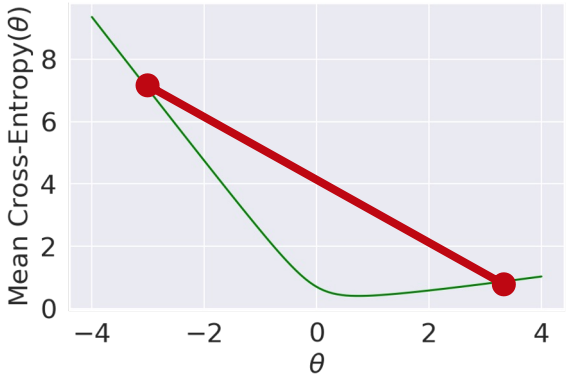
	x	y
0	-4.0	0
1	-2.0	0
2	-0.5	1
3	1.0	0
4	3.0	1
5	5.0	1

Squared Loss Surface



A straight line crosses the curve
Non-convex

Cross-Entropy Loss Surface



Convex!

Logistic Regression with cross entropy

1. Choose a model

Logistic Regression $\hat{y} = f_\theta(x) = P(Y = 1|x) = \sigma(x^T\theta)$

2. Choose a loss function

~~$R(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \sigma(\mathbb{X}_i^T\theta))^2$~~ $\text{loss} = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$

$$R(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\sigma(\mathbb{X}_i^T\theta)) + (1-y_i)\log(1-\sigma(\mathbb{X}_i^T\theta))\right)$$

3. Fit the model

4. Evaluate model performance

Logistic Regression with cross entropy

1. Choose a model

2. Choose a loss function

3. Fit the model

4. Evaluate model performance

Logistic Regression $\hat{y} = f_\theta(x) = P(Y = 1|x) = \sigma(x^T\theta)$

$$R(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\sigma(\mathbb{X}_i^T\theta)) + (1-y_i)\log(1-\sigma(\mathbb{X}_i^T\theta))\right)$$

For logistic regression, we can use squared loss if we want to!

- Using squared loss and using cross-entropy loss will usually result in different $\hat{\theta}$
  - **Different optimization problems, different solutions**.
- **Cross-entropy loss is strictly better than squared loss for logistic regression.**
  - Convex, so easier to minimize using numerical techniques.
  - Better suited for modeling probabilities.

Maximum Likelihood Estimation

Regression vs Classification

Logistic regression model derivation

- logistic function (sigmoid)
- Parameter interpretation

Loss function

- Pitfalls of Squared Loss
- Cross Entropy

Maximum Likelihood Estimation

PMF of the Bernoulli distribution

If Y is the result of one toss of a coin that lands heads with chance p,

$$P(Y = y) = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{if } y = 0 \end{cases}$$
$$= p^y (1 - p)^{1-y}$$

Then, if Y_1 and Y_2 are the results of tosses of two coins:

$$P(Y_1 = y_1, Y_2 = y_2) = \left(p_1^{y_1}(1 - p_1)^{1-y_1}\right)\left(p_2^{y_2}(1 - p_2)^{1-y_2}\right)$$

Estimating the two probabilities

- Suppose we want to estimate the values of p_1 and p_2.
- We know what the likelihood is.

$$P(Y_1 = y_1, Y_2 = y_2) = \left(p_1^{y_1}(1-p_1)^{1-y_1}\right)\left(p_2^{y_2}(1-p_2)^{1-y_2}\right)$$

- Our goal is to find the p_1 and p_2 that **maximize** the above function, over all p_1 and p_2.
  - Maximize, because we are looking for the p_1 and p_2 that are “most likely” to have generated the data that we observed.
- As before, this involves differentiating, setting equal to 0, and solving.

Log likelihoods

- Maximizing $P(Y_1 = y_1, Y_2 = y_2) = \left(p_1^{y_1}(1-p_1)^{1-y_1}\right)\left(p_2^{y_2}(1-p_2)^{1-y_2}\right)$ is annoying.
  - Products -> chain rule.
- log(x) is a **strictly increasing** function.
  - If a > b, then log(a) > log(b).
- This means, the values of p_1 and p_2 that maximize $P(Y_1 = y_1, Y_2 = y_2)$ are the same values that maximize

$$\log\left(\left(p_1^{y_1}(1-p_1)^{1-y_1}\right)\left(p_2^{y_2}(1-p_2)^{1-y_2}\right)\right)$$
$$= y_1\log(p_1) + (1-y_1)\log(1-p_1) + y_2\log(p_2) + (1-y_2)\log(1-p_2)$$
$$= \sum_{i=1}^{2}\left(y_i\log(p_i) + (1-y_i)\log(1-p_i)\right)$$

Starting to look familiar!

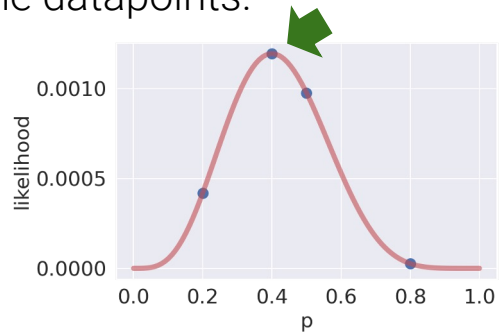
Likelihood of Data; Definition of Probability

A Bernoulli random variable Y with parameter p has distribution: $P(Y = y) = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{if } y = 0 \end{cases}$

Given that all flips are IID from the same coin (probability of heads = p), the **likelihood** of our data is **proportional to** the probability of observing the datapoints.

Training data: [0, 0, 1, 1, 1, 1, 0, 0, 0, 0]

Data likelihood: $p^4(1-p)^6$.



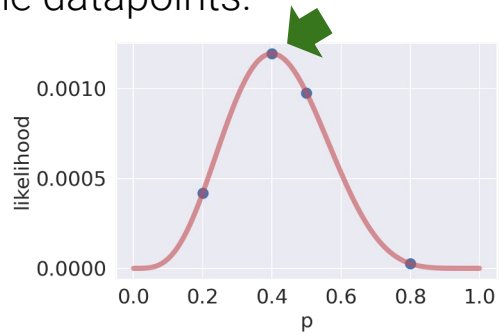
Likelihood of Data

A Bernoulli random variable Y with parameter p has distribution: $P(Y = y) = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{if } y = 0 \end{cases}$

Given that all flips are IID from the same coin (probability of heads = p), the **likelihood** of our data is **proportional to** the probability of observing the datapoints.

Training data: [0, 0, 1, 1, 1, 1, 0, 0, 0, 0]

Data likelihood: $p^4(1-p)^6$.



An example of a bad estimate is parameter $p = 0.1$ since the likelihood of observing the training data is going to be :

$$(0.1)^4(0.9)^6 = 0.000053$$

An example of a bad estimate is parameter $p = 0.4$ since the likelihood of observing the training data is going to be :

$$(0.4)^4(0.6)^6 = 0.001194$$

Generalization of the Coin Demo

For training data: {0, 0, 1, 1, 1, 1, 0, 0, 0, 0}

0.4 is the most "intuitive" θ for two reasons:

1. Frequency of heads in our data
2. Maximizes the **likelihood** of our data:

$$\hat{\theta} = \underset{\theta}{\text{argmax}} \left(\theta^4 (1-\theta)^6\right)$$

Parameter θ:
Probability that
IID flip == 1 (Heads)

Prediction:
1 or 0

How can we generalize this notion of likelihood to **any** random binary sample?

$$\{y_1, y_2, \ldots, y_n\} \Rightarrow \hat{\theta} = \underset{\theta}{\text{argmax}} \left(???\right)$$

data (1's and 0's)

likelihood

A Compact Representation of the Bernoulli Probability Distribution

How can we generalize this notion of likelihood to **any** random binary sample?

$$\{y_1, y_2, \ldots, y_n\} \Rightarrow \hat{\theta} = \underset{\theta}{\operatorname{argmax}} \, (???)$$

data (1's and 0's) — likelihood

Let Y be Bernoulli(p). The probability distribution can be written compactly:

$$P(Y = y) = p^y (1 - p)^{1-y}$$

For P(Y = **1**), only this term stays

For P(Y = **0**), only this term stays

(long, non-compact form):

$$P(Y = y) = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{if } y = 0 \end{cases}$$

Generalized Likelihood of Binary Data

How can we generalize this notion of likelihood to **any** random binary sample?

$$\{y_1, y_2, \ldots, y_n\} \Rightarrow \hat{\theta} = \underset{\theta}{\operatorname{argmax}} \, (???)$$

data (1's and 0's) — likelihood

Let Y be Bernoulli(p). The probability distribution can be written compactly:

$$P(Y = y) = p^y (1-p)^{1-y}$$

For P(Y = **1**), only this term stays

For P(Y = **0**), only this term stays

If binary data are **IID with same** probability p, then the likelihood of the data is:

$$\prod_{i=1}^{n} p^{y_i} (1-p)^{(1-y_i)}$$

Ex: {0, 0, 1, 1, 1, 1, 0, 0, 0, 0} → $p^4(1-p)^6$

Generalized Likelihood of Binary Data

How can we generalize this notion of likelihood to any random binary sample?

$$\{y_1, y_2, \ldots, y_n\} \Rightarrow \hat{\theta} = \underset{\theta}{\operatorname{argmax}}\,(???)$$

data (1's and 0's) — likelihood

Let Y be Bernoulli(p). The probability distribution can be written compactly:

$$P(Y = y) = p^y (1-p)^{1-y}$$

For P(Y = **1**), only this term stays

For P(Y = **0**), only this term stays

If binary data are **IID with same** probability p, then the likelihood of the data is:

$$\prod_{i=1}^{n} p^{y_i} (1-p)^{(1-y_i)}$$

If data are independent with **different** probability p_i, then the likelihood of the data is:

$$\prod_{i=1}^{n} p_i^{y_i} (1-p_i)^{(1-y_i)}$$

Maximum Likelihood Estimation (MLE)

Our **maximum likelihood estimation** problem:

- For i = 1, 2, …, n, let Y_i be independent Bernoulli(p_i). Observe data $\{y_1, y_2, \ldots, y_n\}$.
- We'd like to estimate $p_1, p_2, \ldots, p_n$.

Find $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n$ that **maximize** $\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}$

Maximum Likelihood Estimation (MLE)

Our **maximum likelihood estimation** problem:

- For i = 1, 2, ..., n, let Y_i be independent Bernoulli(p_i). Observe data $\{y_1, y_2, \ldots, y_n\}$.
- We'd like to estimate $p_1, p_2, \ldots, p_n$.

Find $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n$ that **maximize** $\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}$

Equivalent, simplifying optimization problems (since we need to take the first derivative):

$$\underset{p_1, p_2, \ldots, p_n}{\textbf{maximize}} \quad \log\left(\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}\right)$$

(log is an increasing function. If a > b, then log(a) > log(b).)

$$= \sum_{i=1}^{n} \log\left(p_i^{y_i}(1-p_i)^{(1-y_i)}\right) = \sum_{i=1}^{n} \left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$$

Maximum Likelihood Estimation (MLE)

Our **maximum likelihood estimation** problem:

- For i = 1, 2, …, n, let Y_i be independent Bernoulli(p_i). Observe data $\{y_1, y_2, \ldots, y_n\}$.
- We'd like to estimate $p_1, p_2, \ldots, p_n$.

Find $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n$ that **maximize** $\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}$

Equivalent, simplifying optimization problems (since we need to take the first derivative):

maximize $p_1, p_2, \ldots, p_n$

$$\log\left(\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}\right)$$

(log is an increasing function. If a > b, then log(a) > log(b).)

$$= \sum_{i=1}^{n} \log\left(p_i^{y_i}(1-p_i)^{(1-y_i)}\right) = \sum_{i=1}^{n} \left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$$

minimize $p_1, p_2, \ldots, p_n$

$$-\frac{1}{n}\sum_{i=1}^{n} \left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$$

Maximizing Likelihood == Minimizing Average Cross-Entropy

maximize $\underset{p_1, p_2, \ldots, p_n}{}$ $\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}$

Log is increasing;
max/min properties

minimize $\underset{p_1, p_2, \ldots, p_n}{}$ $-\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$

For logistic regression,
let $p_i = \sigma(X_i^T\theta)$

minimize $\underset{\theta}{}$ $\underbrace{-\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\sigma(X_i^T\theta) + (1-y_i)\log(1-\sigma(X_i^T\theta))\right)}$

Cross-Entropy Loss!!

Minimizing cross-entropy loss is equivalent to **maximizing the likelihood of the training data**.

- We are choosing the model parameters that are "most likely", given this data.

Assumption: all data drawn **independently** from the same logistic regression model with parameter θ

- It turns out that many of the model + loss combinations we've seen can be motivated using MLE (OLS, Ridge Regression, etc.)
- You will study MLE further in probability and ML classes. But now you know it exists.

Solving Maximum Likelihood Estimation (Not required)

$$R(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\sigma(\mathbb{X}_i^T\theta)) + (1-y_i)\log(1-\sigma(\mathbb{X}_i^T\theta))\right)$$

- Approach 1: Gradient Descent (take larger – more certain – steps opposite the gradient)
- Approach 2: Stochastic Gradient Descent (SGD) (take many small steps opposite the gradient)
- Approach 3: Newton's Method (use second derivatives to better follow curvature)

Logistic Regression

1. Choose a model

Logistic Regression $\hat{y} = f_\theta(x) = P(Y = 1|x) = \sigma(x^T\theta)$

2. Choose a loss function

$$R(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\sigma(\mathbb{X}_i^T\theta)) + (1 - y_i)\log(1 - \sigma(\mathbb{X}_i^T\theta))\right)$$

3. Fit the model

✅

4. Evaluate model performance

Next time!