Logistic Regression, Cross Entropy Loss, Classification (Ch. 17.4-17.7)

UC Berkeley Data 100 Fall 2019
Suraj Rampure

(Slides adapted from Josh Hug, Sam Lau, John DeNero)

Announcements

Hi, I'm Suraj! I'm not Deb or Josh, but I'm super grateful that they let me do this.
There's a feedback form in the live lecture Piazza thread – please fill it out if you can!

- Thursday's lecture will be midterm review by Josh.
- Midterm 2 next Wednesday (11/13), from 8-9:30PM. Logistics post coming soon.
- Next Tuesday's lecture (11/12) will be the final classification lecture.
    - 11/12 will not be in scope for the midterm – **today's lecture is the last in-scope.**
    - We won't take attendance next Tuesday, but it'll be in scope for the final.
- Homework 7 due Saturday 11:59pm. Primarily about gradient descent.

Goals for Today

Part 1: Reformulate logistic regression

- Modeling probabilities instead of real numbers.

Part 2: Motivative and talk about cross entropy loss

- L2 loss isn’t terrible choice for logistic regression, but we can do better.

Part 3: Classification and evaluating classifiers

- Logistic regression in and of itself isn’t a classifier – it just models probabilities.
- How do we actually classify things, then?
- What does it mean for our classifier to be “good”?

Logistic Regression

Linear vs. Logistic Regression

In a **linear regression** model with p features, our goal is to predict a **quantitative** variable (i.e., some real number) from those features.

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

- Our output can be **any real number**.

In a **logistic regression** model with p features, our **goal** is to predict a **categorical** variable from those features.

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

- The output of **logistic regression** is always between 0 and 1, i.e. it is **quantitative**!
  - Gives probability under our model that the category is 1.
- Our goal is to perform **binary classification –** to predict either 0 or 1.
  - How do we actually classify, then? We will find out today!

Remember, $\vec{x}^T \vec{\hat{\beta}} = x_1\hat{\beta}_1 + x_2\hat{\beta}_2 + \ldots + x_p\hat{\beta}_p$

Logistic Regression and the Logistic Function

In logistic regression, we're modelling the **probability** that an observation belongs to class 1 (as opposed to class 0).

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

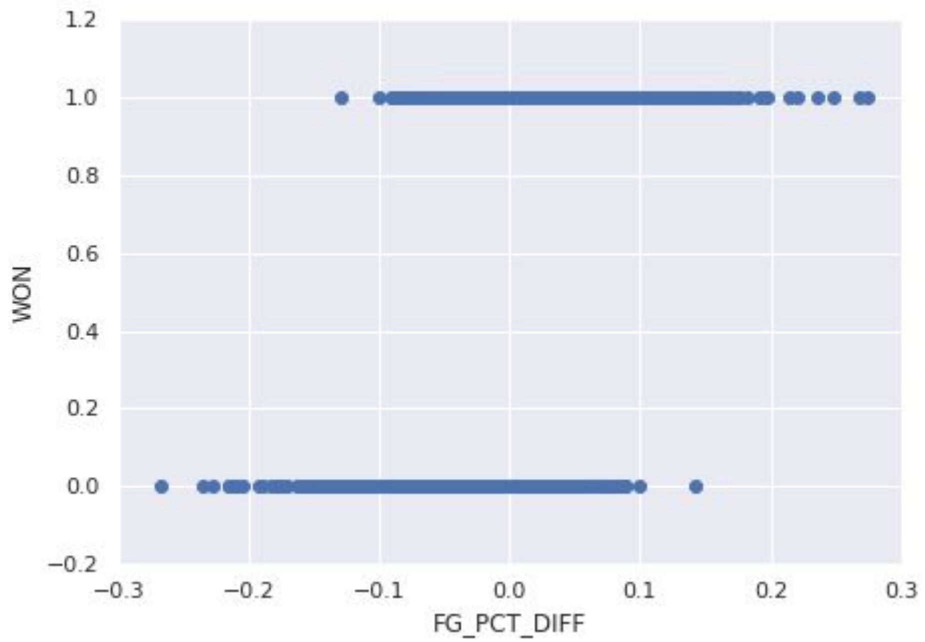
Our model looks like the linear regression model, except with a $\sigma(\cdot)$ around it.

This is called the **logistic function**. It is a type of sigmoid.

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

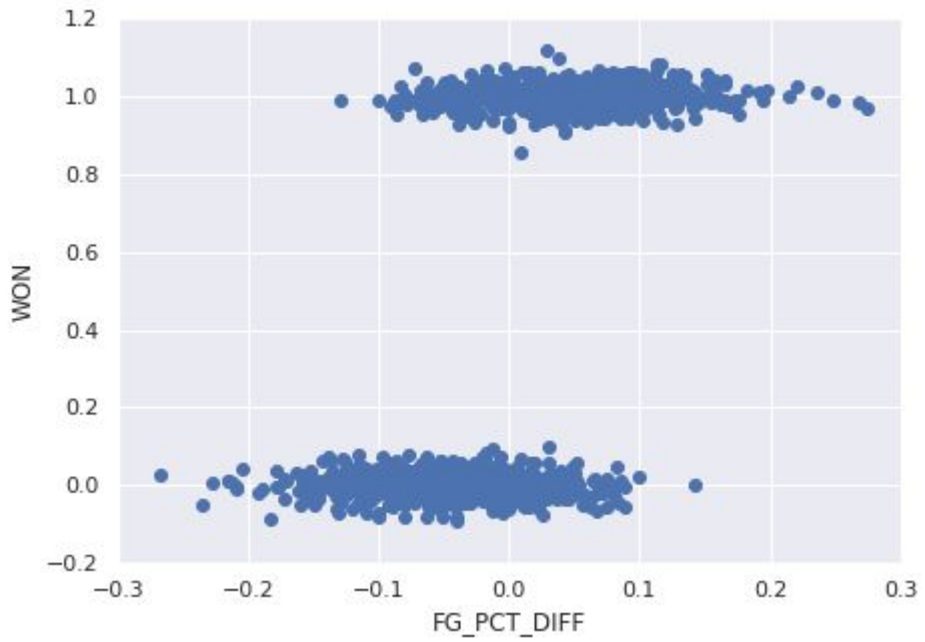
Let's walk through the steps we took last lecture on our way to determining our model.

Logistic Regression (demo from last lecture)



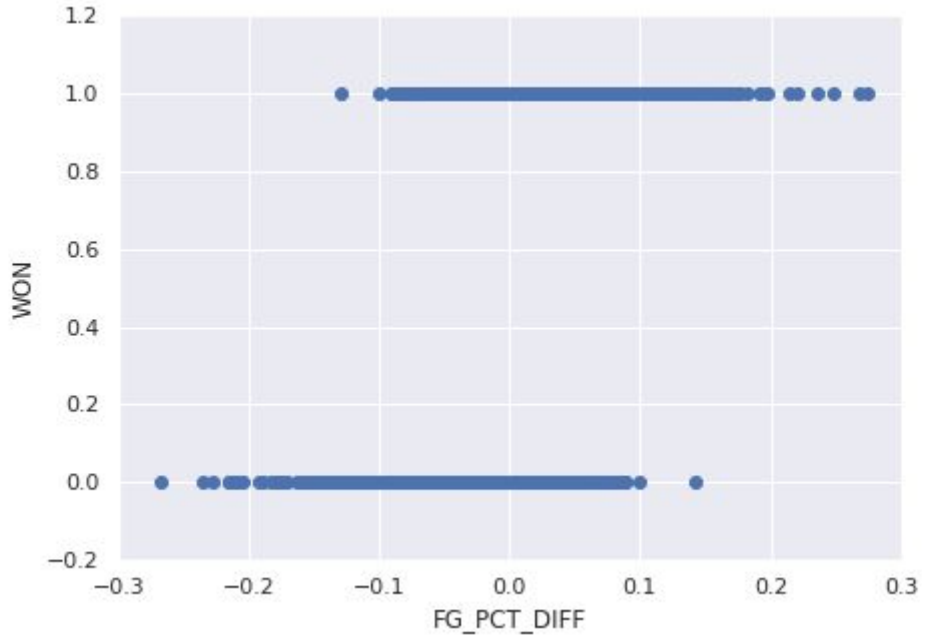
Recall, we looked at whether or not a team won, given their FG_PCT_DIFF.

Logistic Regression (demo from last lecture)



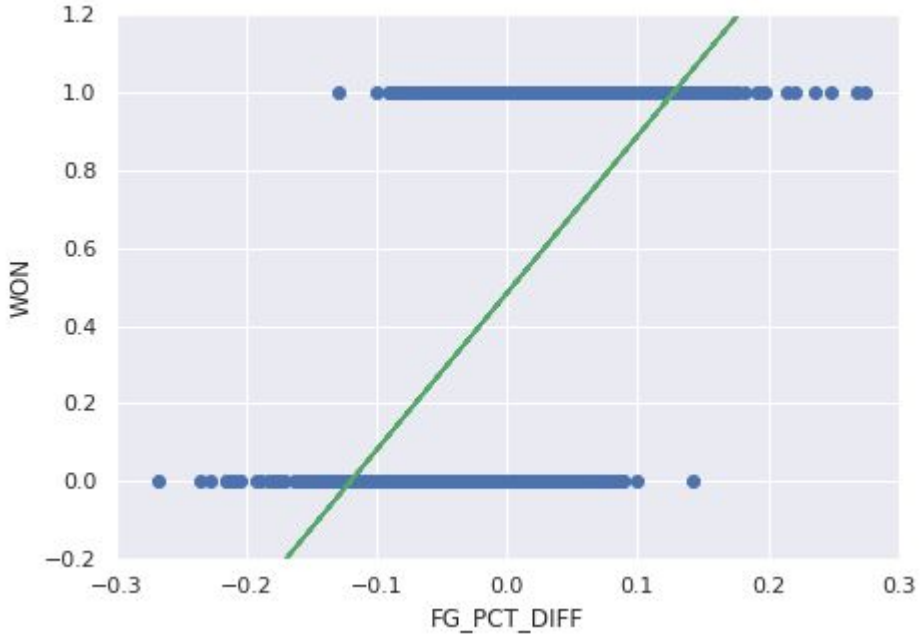
Recall, we looked at whether or not a team won, given their FG_PCT_DIFF.

Logistic Regression (demo from last lecture)



Recall, we looked at whether or not a team won, given their FG_PCT_DIFF.

Logistic Regression (demo from last lecture)

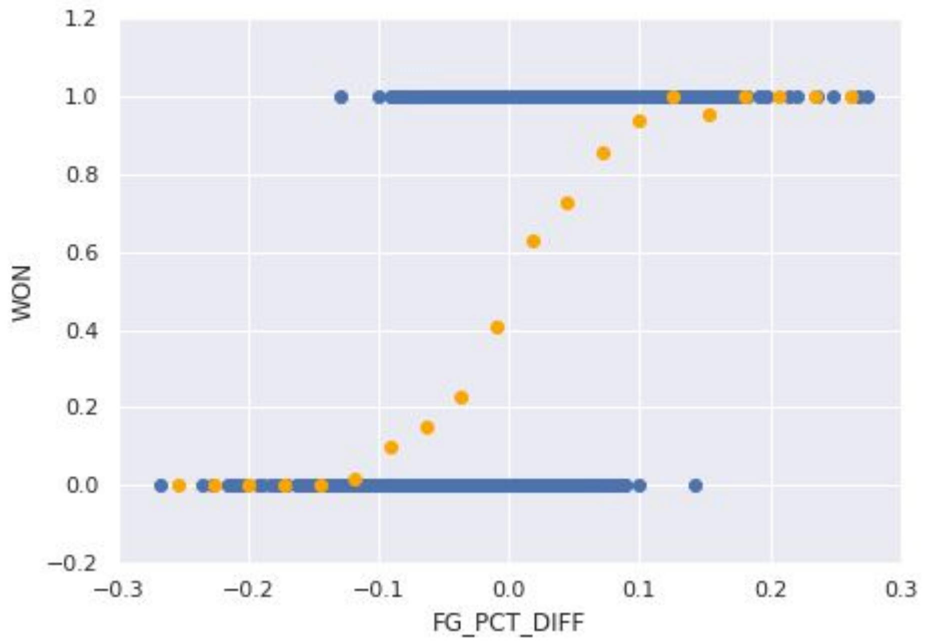


Recall, we looked at whether or not a team won, given their FG_PCT_DIFF.

In **green**, we used ordinary least squares to try and compute the probability of our point having WON = 1 from FG_PCT_DIFF.

There are multiple problems with this. Primarily, it gives us outputs that are not probabilities (less than 0, greater than 1).

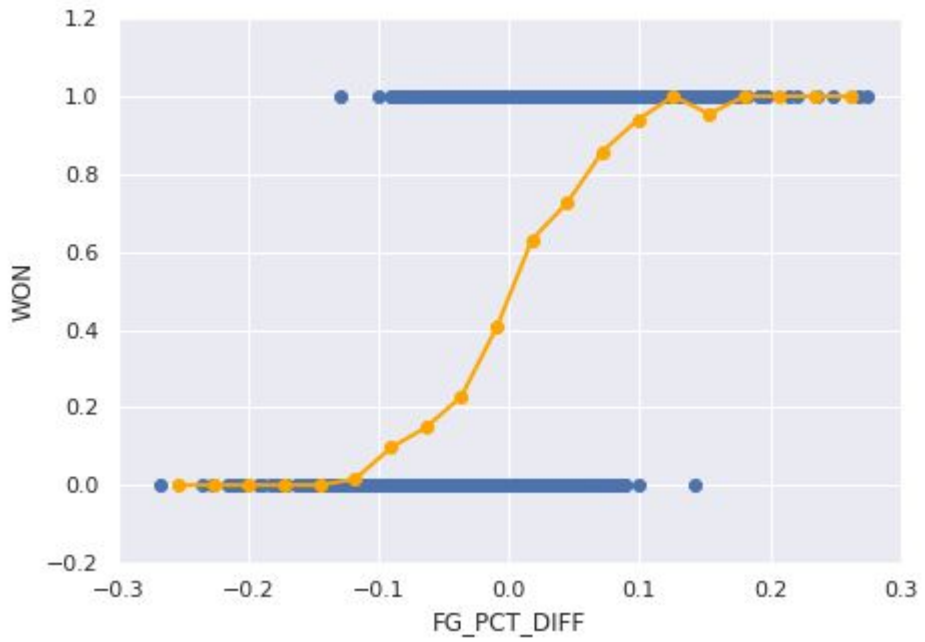
Logistic Regression (demo from last lecture)



Recall, we looked at whether or not a team won, given their FG_PCT_DIFF.

In **orange**, we split our x-axis (FG_PCT_DIFF) into 20 bins (intervals) of equal width, and computed the **average WON value** for each bin.

Logistic Regression (demo from last lecture)

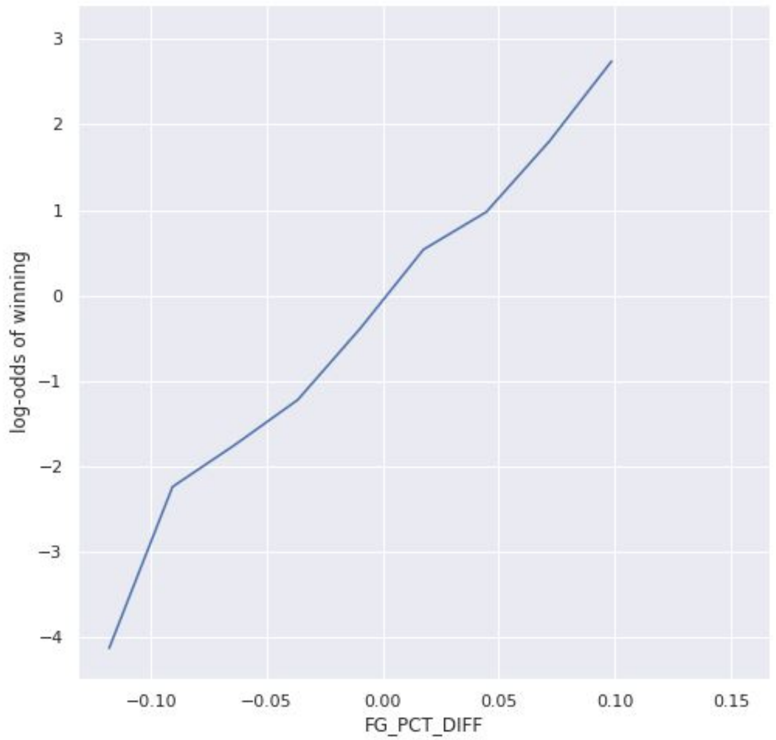


Recall, we looked at whether or not a team won, given their FG_PCT_DIFF.

In **orange**, we split our x-axis (FG_PCT_DIFF) into 20 bins (intervals) of equal width, and computed the **average WON value** for each bin.

Our goal was to **model this orange curve.**

Logistic Regression (demo from last lecture)



On the left is a plot of the "log-odds" of $p = P(Y = 1|\vec{x})$
(Think of p as being the orange dots on the previous slide.)

$$\text{odds}(p) = \frac{p}{1-p}$$

$$\text{log-odds}(p) = \log\left(\frac{p}{1-p}\right) = \log\left(\frac{P(Y=1|\vec{x})}{P(Y=0|\vec{x})}\right)$$

We noticed that the plot of log-odds is roughly linear.

$$\log\left(\frac{P(Y=1|\vec{x})}{P(Y=0|\vec{x})}\right) = \vec{x}^T\hat{\vec{\beta}}$$

Through this **assumption**, we derived the following:

$$\Longrightarrow \boxed{P(Y=1|\vec{x}) = \sigma(\vec{x}^T\hat{\vec{\beta}})}$$

The derivation of this is on the following slide.
We'll skip it during lecture, but you can refer to it later.

Derivation of Sigmoid (taken from Josh's notebook)

Abbreviate $P(Y=1|\vec{x})$ as p and $\vec{x}^T\hat{\vec{\beta}}$ as t, then:

$$\log\left(\frac{P(Y=1|\vec{x})}{P(Y=0|\vec{x})}\right) = \vec{x}^T\hat{\vec{\beta}}$$

$$\log\left(\frac{p}{1-p}\right) = t \qquad \text{; Abbreviate and complement rule}$$

$$\frac{p}{1-p} = \exp(t) \qquad \text{; exponentiate both sides}$$

$$p = \exp(t) - p\exp(t) \qquad \text{; multiply by } 1-p$$

$$p(1+\exp(t)) = \exp(t) \qquad \text{; add } p\exp(t) \text{ and factor out } p$$

$$p = \frac{\exp(t)}{1+\exp(t)} \qquad \text{; divide by } 1+\exp(t)$$

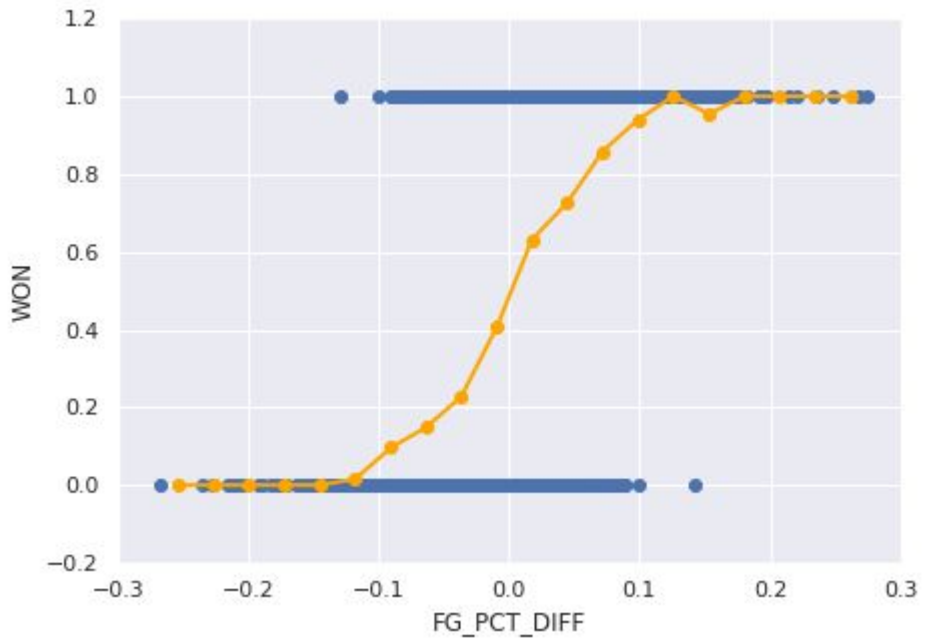
$$p = \frac{1}{1+\exp(-t)} \qquad \text{; multiply by } \frac{\exp(-t)}{\exp(-t)}$$

$$P(Y=1|\vec{x}) = \frac{1}{1+\exp(-\vec{x}^T\hat{\vec{\beta}})} \qquad \text{; Unabbreviate}$$

This transformation is called the logistic function, traditionally denoted σ (sigma).

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

Logistic Regression (demo from last lecture)



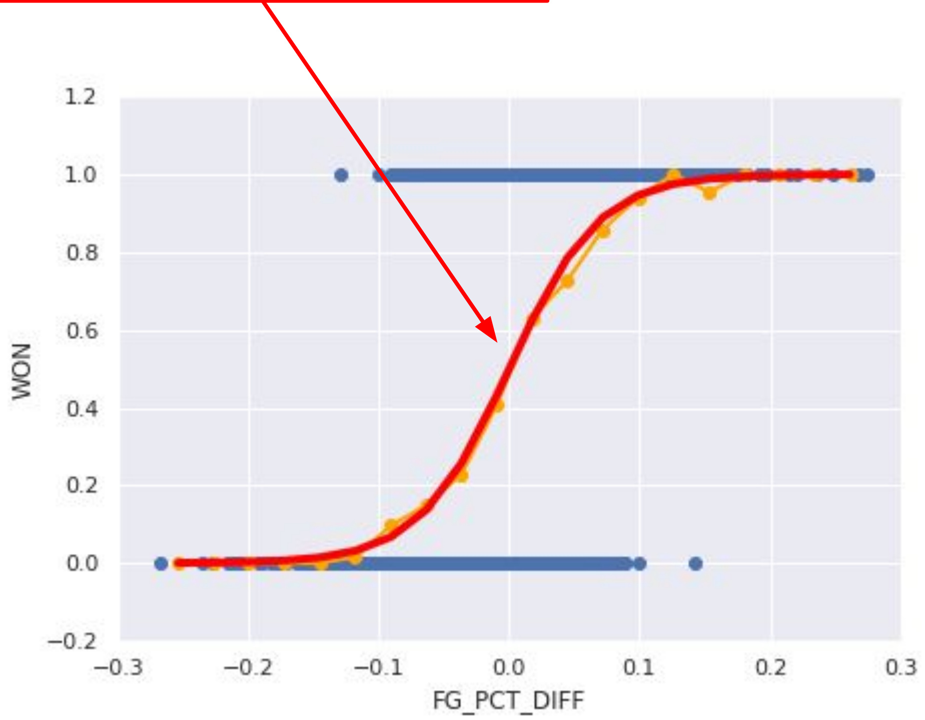
Recall, we looked at whether or not a team won, given their FG_PCT_DIFF.

In **orange**, we split our x-axis (FG_PCT_DIFF) into 20 bins (intervals) of equal width, and computed the **average WON value** for each bin.

Our goal was to **model this orange curve.**

Logistic Regression (demo from last lecture)

$$\boxed{\hat{y} = \sigma(30 \cdot \text{FG PCT DIFF})}$$



Recall, we looked at whether or not a team won, given their FG_PCT_DIFF.

In **orange**, we split our x-axis (FG_PCT_DIFF) into 20 bins (intervals) of equal width, and computed the **average WON value** for each bin.

Our goal was to **model this orange curve.**

Under the **assumption** that our log-odds is linear, we derived the **red curve**, which serves as our **logistic model**.

Logistic Regression

In logistic regression, we're modelling the **probability** that an observation belongs to class 1 (as opposed to class 0).

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

Our model looks like the linear regression model, except with a $\sigma(\cdot)$ around it.

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

- $\sigma(t)$ came from the assumption that the **log-odds of p is roughly linear.**
- Has a nice "s" shape, and has outputs bounded between 0 and 1.
    - *Fun Fact: Our choice of* $\sigma(t)$ *is called the "logistic function." There is actually a whole family of sigmoid functions we could use (arctan, tanh, erf…).*

Algebraic Tricks for Computing Probabilities

The following identities may be useful in manually computing $P(Y=0|\vec{x})$.

(This slide was skipped during live lecture, but it's here for your reference.)

$$\begin{aligned} P(Y=1|\vec{x}) &= \sigma(\vec{x}^T\hat{\vec{\beta}}) \\ &= \frac{1}{1+e^{-\vec{x}^T\hat{\vec{\beta}}}} \\ &= \frac{e^{\vec{x}^T\hat{\vec{\beta}}}}{e^{\vec{x}^T\hat{\vec{\beta}}}+1} \end{aligned}$$

Multiply by $\frac{e^{\vec{x}^T\hat{\vec{\beta}}}}{e^{\vec{x}^T\hat{\vec{\beta}}}}$

$$\begin{aligned} P(Y=0|\vec{x}) = 1 - P(Y=1|\vec{x}) &= 1 - \frac{e^{\vec{x}^T\hat{\vec{\beta}}}}{e^{\vec{x}^T\hat{\vec{\beta}}}+1} \\ &= \frac{1}{e^{\vec{x}^T\hat{\vec{\beta}}}+1} \\ &= \sigma(-\vec{x}^T\hat{\vec{\beta}}) \end{aligned}$$

Logistic Regression – Your Turn

Suppose I want to predict the probability that LeBron's shot goes in, given **shot distance** (first feature) and # of **seconds left** on the shot clock (second feature).

I fit a model using my training data, and somehow compute

$$\vec{\hat{\beta}}^T = \begin{bmatrix} 0.1 & -0.5 \end{bmatrix}$$

Under the logistic model, compute the **probability his shot goes in**, given that

- He shoots it from **15** feet.
- There is **1** second left on the shot clock.

Logistic Regression – Your Turn (Solution)

$$\vec{x}^T = \begin{bmatrix} 15 & 1 \end{bmatrix} \qquad \vec{\hat{\beta}}^T = \begin{bmatrix} 0.1 & -0.5 \end{bmatrix}$$

$$\begin{aligned} P(Y = 1|\vec{x}) &= \sigma(\vec{x}^T \vec{\hat{\beta}}) \\ &= \sigma(\hat{\beta}_1 \cdot \text{SHOT DISTANCE} + \hat{\beta}_2 \cdot \text{SECONDS LEFT}) \\ &= \sigma(0.1 \cdot 15 + (-0.5) \cdot 1) \\ &= \sigma(1) \\ &= \frac{1}{1 + e^{-1}} \\ &\approx 0.7311 \end{aligned}$$

An explicit expression representing our model.

Motivating Cross Entropy Loss

Choosing a Loss Function

To find $\hat{\vec{\beta}}$, we need to minimize empirical risk.

In order to do that, we need to **choose a loss function**.

In the last lecture and in Lab 11, we chose to use L2 loss (i.e. our empirical risk was MSE).

- That is, given our observations / training set $\{(\vec{x_1}, y_1), (\vec{x_2}, y_2), ..., (\vec{x_n}, y_n)\}$, our empirical risk was

$$R(\vec{\beta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \sigma(\vec{x_i}^T\vec{\beta}))^2$$

- *Sometimes*, it works just fine! Other times...

Pitfalls of L2 Loss with Logistic Regression

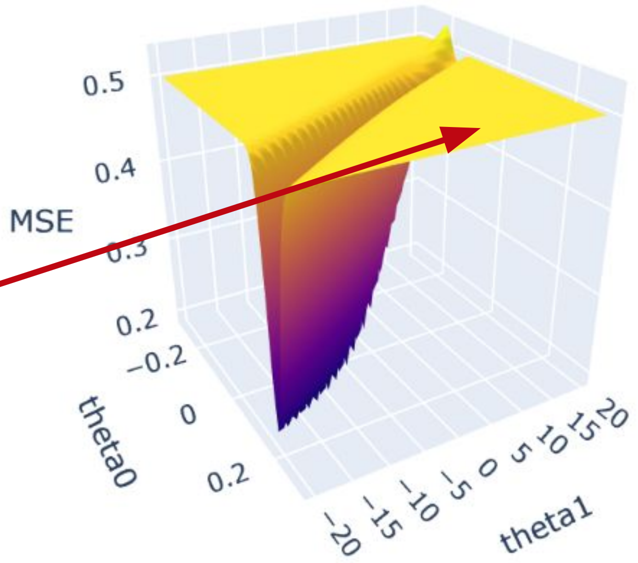
In Lab 11, we saw the loss surface to the right. (A loss surface is a plot of empirical risk vs. parameter.)

If your initial guess for $\hat{\vec{\beta}}$ is way out in the flat yellow region, your numerical optimization routine (e.g. gradient descent) can get stuck.

Gradient descent update equation:

$$\vec{\beta}^{(t+1)} = \vec{\beta}^{(t)} - \alpha \boxed{\nabla_{\vec{\beta}} L(\vec{\beta}, X, \vec{y})}$$

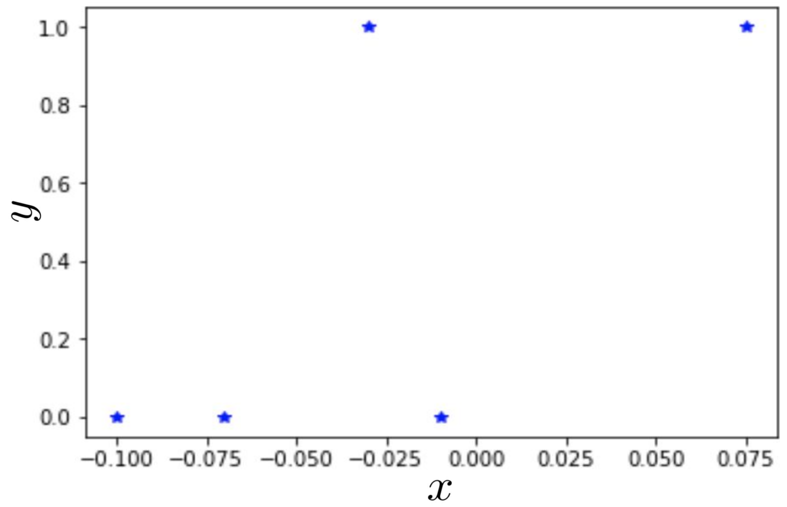
If our gradient is near 0, our updates won't change very much!

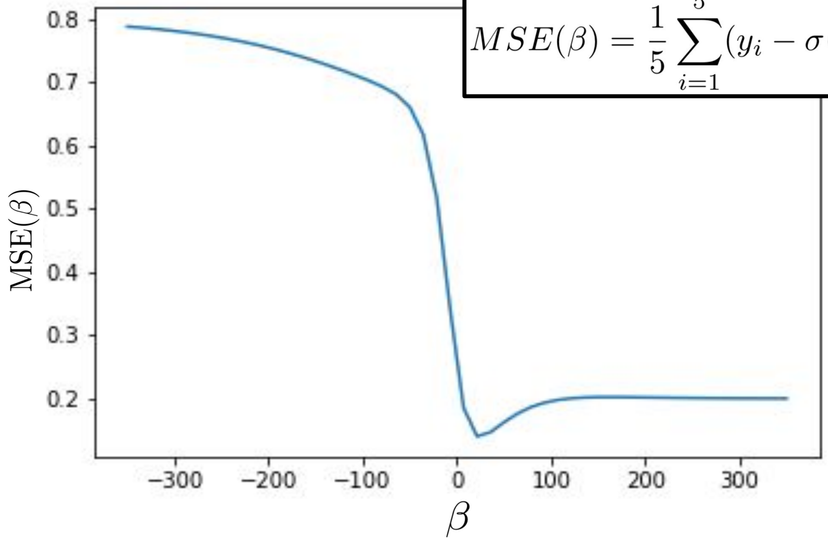


Pitfalls of L2 Loss with Logistic Regression

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 logistic regression model, for different values of β (i.e. the loss surface).

Note: Here, our model has a single feature, so x and beta are scalars. We wouldn't be able to effectively visualize this issue if we had any more features, but it still holds true.





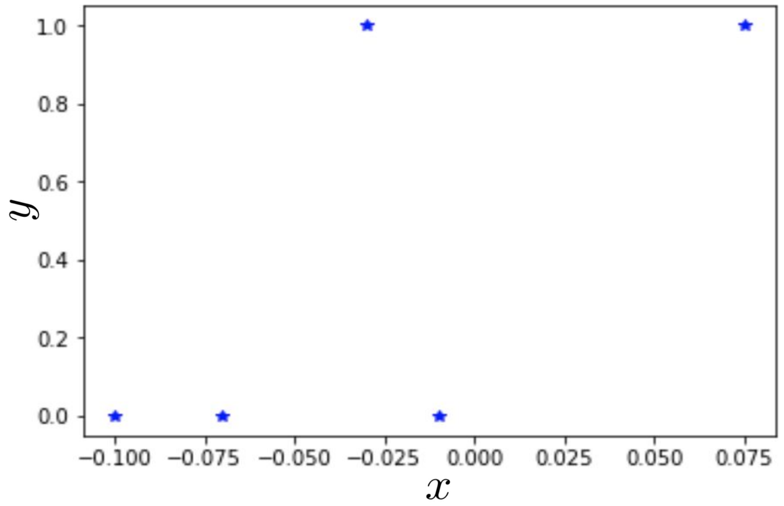
$$\boxed{MSE(\beta) = \frac{1}{5}\sum_{i=1}^{5}(y_i - \sigma(x_i\beta))^2}$$

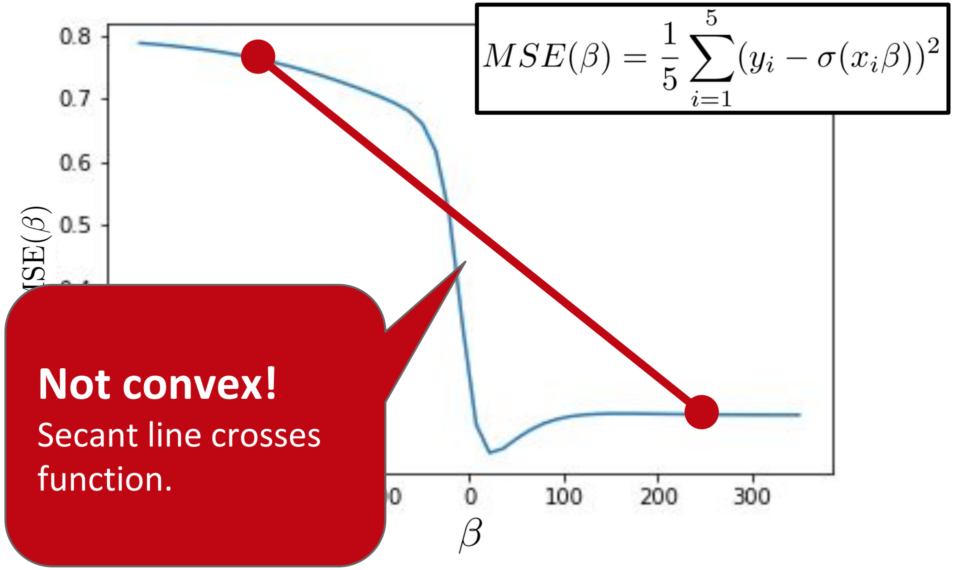
What do you notice?

Pitfalls of L2 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 logistic regression model, for different values of β (i.e. the loss surface).

Note: Here, our model has a single feature, so x and beta are scalars. We wouldn't be able to effectively visualize this issue if we had any more features, but it still holds true.





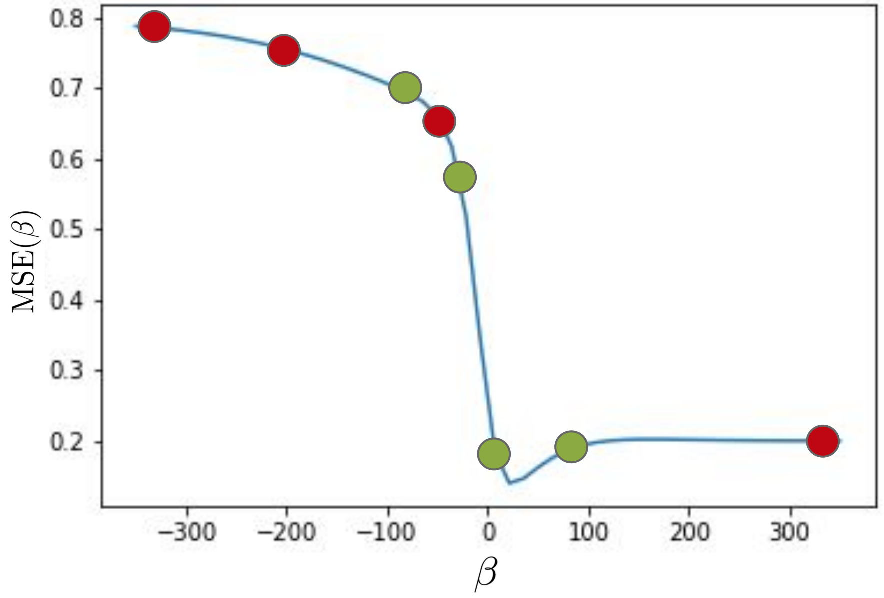
$$MSE(\beta) = \frac{1}{5}\sum_{i=1}^{5}(y_i - \sigma(x_i\beta))^2$$

Not convex!
Secant line crosses function.

Pitfalls of L2 Loss with Logistic Regression

Let's look at the optimal value that scipy.minimize finds, for various initial guesses of β :

Initial guess	Optimal value found via scipy.minimize
-350	-350
-200	317
-90	**~24**
-75	342
-50	**~24**
0	**~24**
90	**~24**
350	350



Even in the neighborhood of the minimizing beta, numerical optimization doesn't always work.

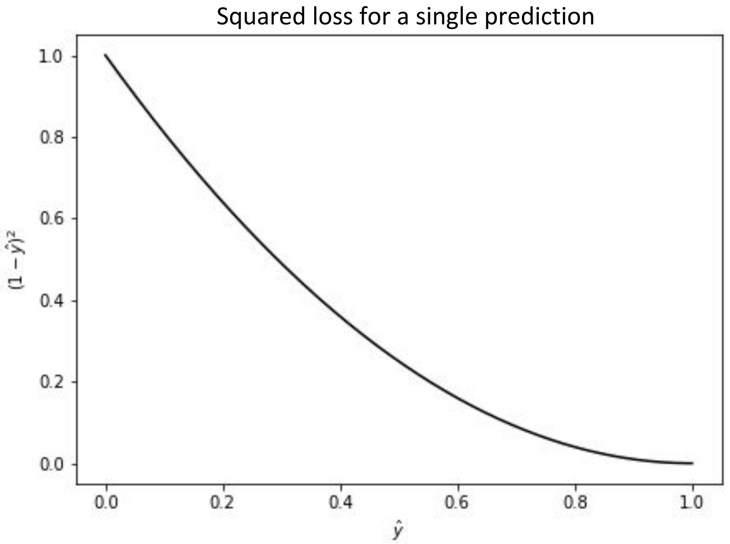
Pitfalls of L2 Loss with Logistic Regression

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**.

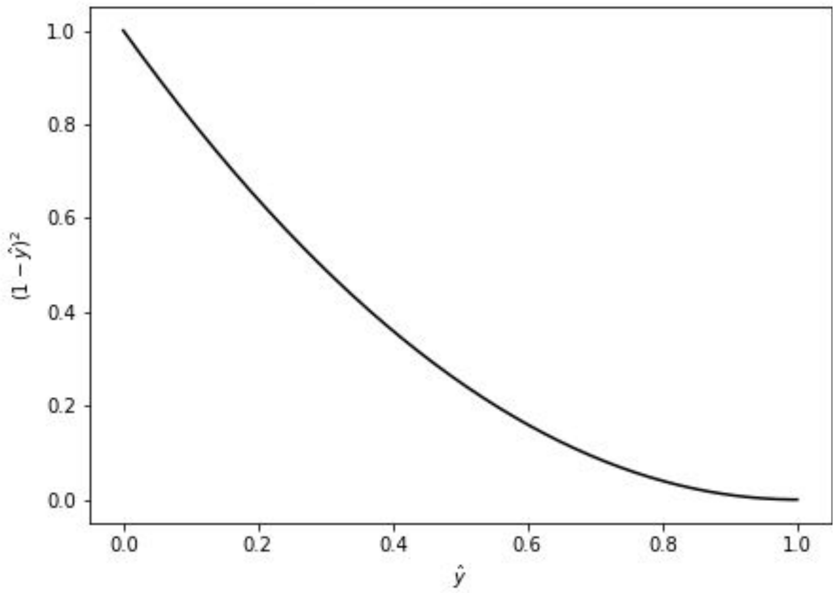


L2 Loss vs. Log Loss

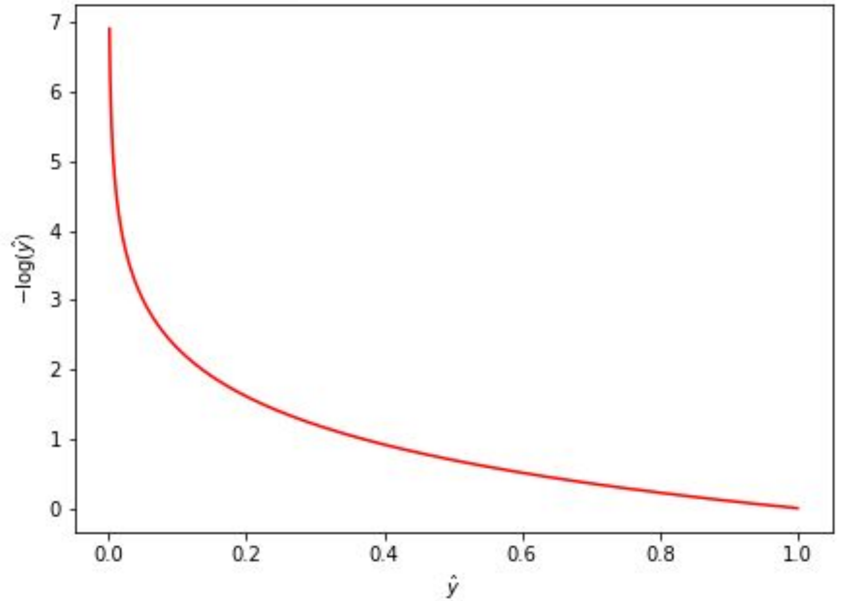
On the left is the graph of squared loss when our correct class is 1.

On the right is the graph of a new loss – **log loss** – also when our correct class is 1.

$(1-\hat{y})^2$ vs $\hat{y}$



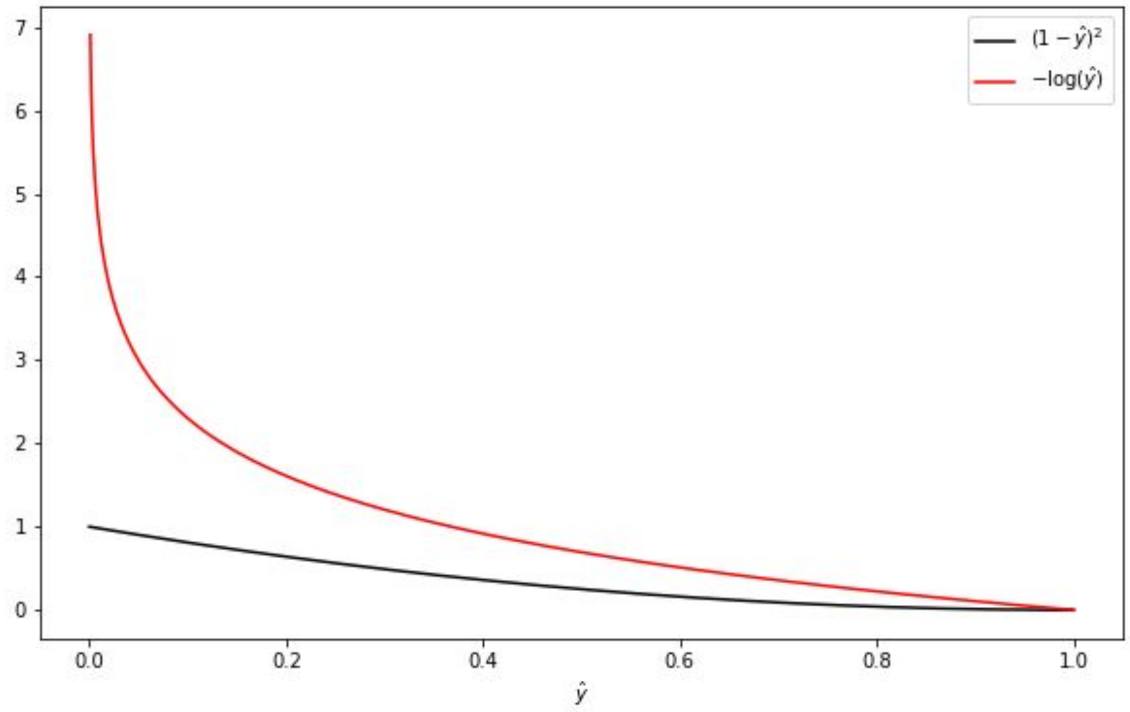
$-\log(\hat{y})$ vs $\hat{y}$



Both of these plots are for a **single prediction.** **What do you notice about log loss?**

L2 Loss vs. Log Loss

- The stark difference between the losses becomes more evident when plotting them on the same axes.
- When our predictions are close, both losses are roughly the same.
- However, the more wrong our prediction is, the larger the output of $-\log(\hat{y})$ is, meaning we're **penalizing wrong answers significantly more** than with $(1-\hat{y})^2$.



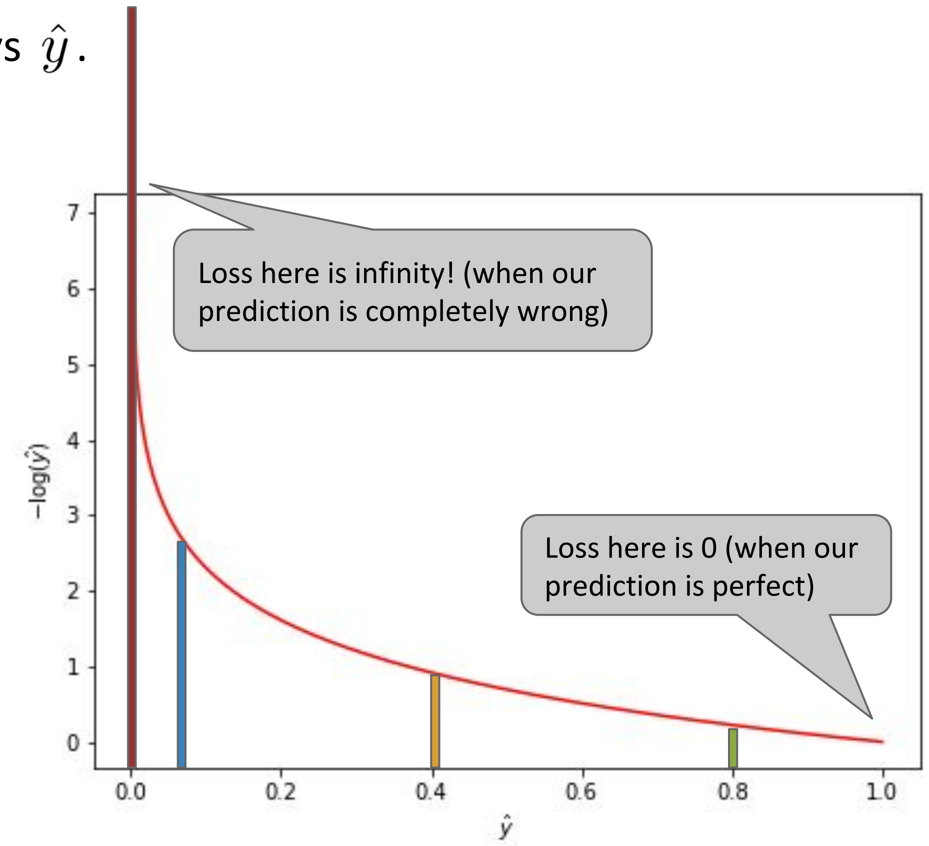
Log Loss

Let's zoom in on the graph of $-\log(\hat{y})$ vs $\hat{y}$.

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

$\hat{y}$	$-\log(\hat{y})$
1	0
0.8	0.25
0.4	1
0.05	3
0	infinity!

Note: sigma never outputs 0 or 1 exactly, so there's never actually 0 loss or infinite loss.

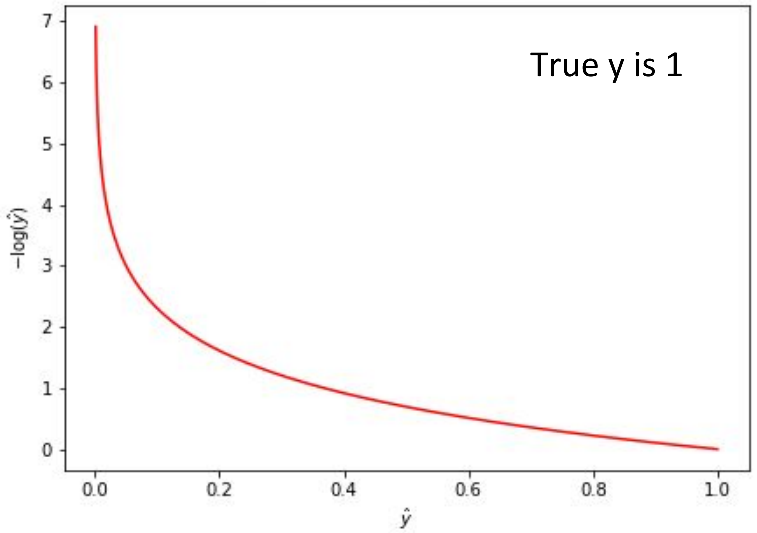


Log Loss

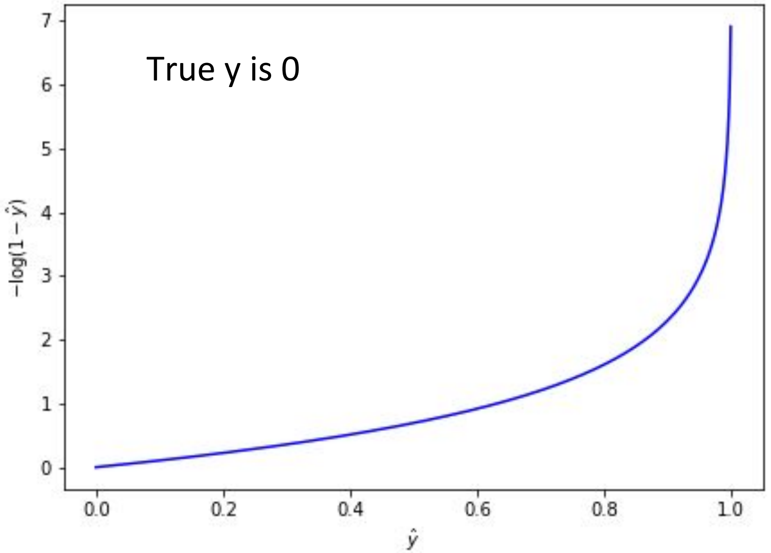
So far, we've only looked at plots when our correct class was 1.

What if our correct class is 0?

$-\log(\hat{y})$ vs $\hat{y}$



$-\log(1-\hat{y})$ vs $\hat{y}$



If the correct class is 0, we want to have **low** loss for values of $\hat{y}$ **close to 0**, and **high** loss for values of $\hat{y}$ **close to 1**. **This is achieved by just "flipping" the plot on the left!**

Cross Entropy Loss

We can combine the two cases from the previous slide into a single loss function:

$$\text{loss} = \begin{cases} -\log(1-\hat{y}) & y = 0 \\ -\log(\hat{y}) & y = 1 \end{cases}$$

This is often written unconditionally as:

$$\boxed{\text{loss} = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})}$$

Note: *Since y = 0 or 1, one of these two terms is always equal to 0, which reduces this equation to the piecewise one above.*

We call this loss function **cross entropy loss** (or “log loss”).

Cross Entropy Loss

$$\text{loss} = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$

or, in terms of our observation $(\vec{x_i}, y_i)$:

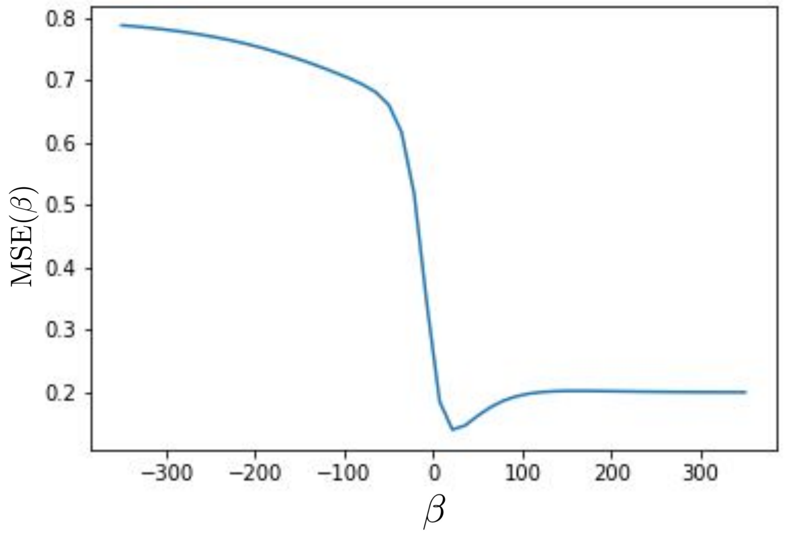
$$-y_i \log(\sigma(\vec{x_i}^T\vec{\beta})) - (1-y_i)\log(1-\sigma(\vec{x_i}^T\vec{\beta}))$$

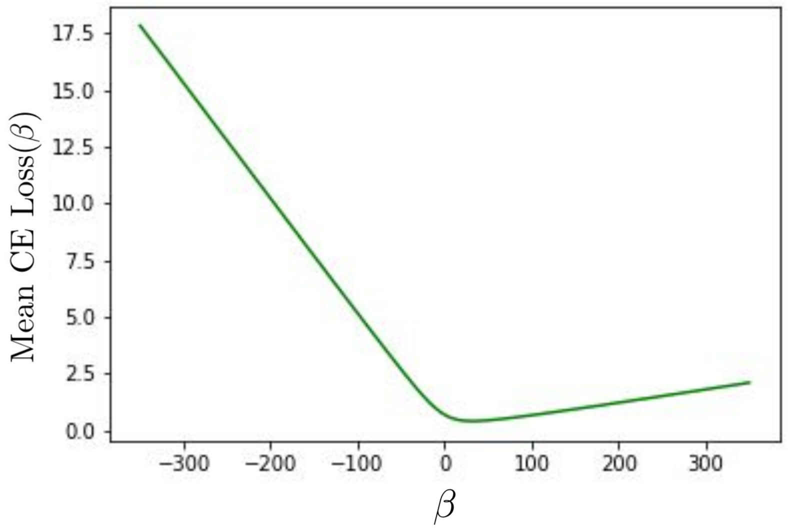
Benefits of cross entropy loss over L2 loss, for logistic regression:

1. The average loss (empirical risk) over all the data is guaranteed to be convex when used with logistic regression, and thus easier to optimize (we won't prove this).
2. It more strongly penalizes very bad predictions.
3. It has roots in probability and information theory (maximum likelihood estimation, KL divergence). *The TL;DR is that it's well-suited for comparing probability distributions. Take CS 189, Data 102, or EE 229A if interested!*

Cross Entropy Loss Surface vs. L2 Loss Surface (Toy Data)

On the left, we have a plot of the MSE loss surface on our toy dataset from before.
On the right, we have a plot of the mean cross entropy loss surface on the same dataset.



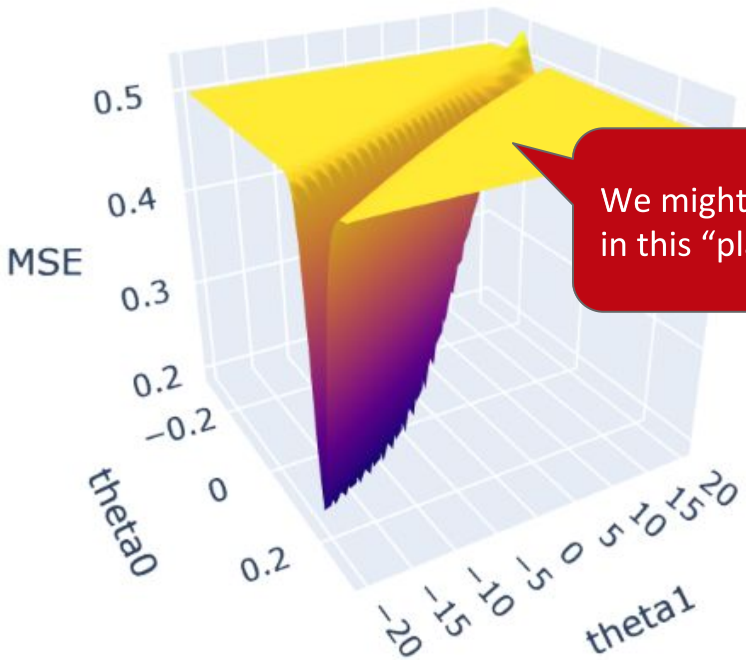


As claimed, mean cross entropy loss is convex **when used with logistic regression.**

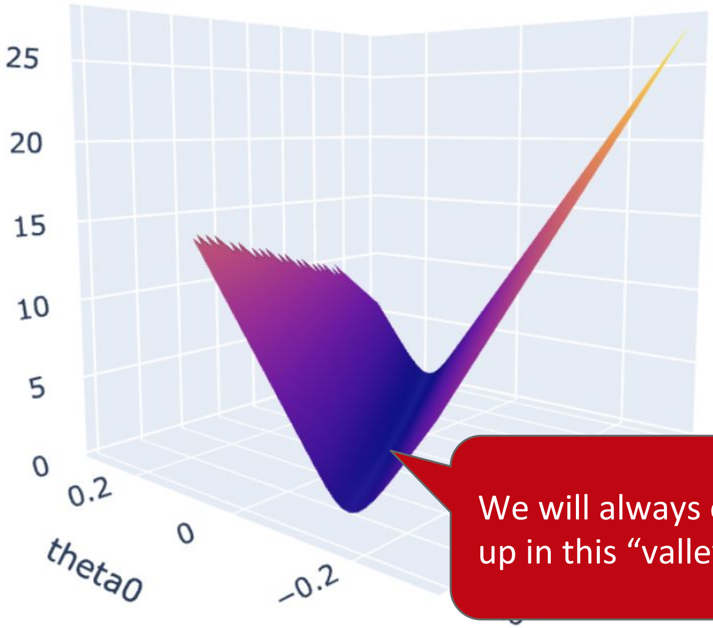
Note: The scales of both plots are vastly different. Why is that?

Cross Entropy Loss Surface vs. L2 Loss Surface (Lab Data)

We can also compare the loss surfaces from the NBA data in Lab 11 yesterday.



We might get stuck in this "plateau".



We will always end up in this "valley".

Due to the convexity of cross entropy loss **when used with logistic regression**, numerical techniques will always be able to find the optimal $\hat{\vec{\beta}}$.

Cross Entropy Loss – Your Turn

$$-y_i \log(\sigma(\vec{x_i}^T \vec{\beta})) - (1 - y_i) \log(1 - \sigma(\vec{x_i}^T \vec{\beta}))$$

Compute the cross entropy loss for both of the following observations (individually).

$$\vec{\hat{\beta}}^T = \begin{bmatrix} 1 & 2 \end{bmatrix}$$

Observation 1

$$\vec{x_1}^T = \begin{bmatrix} -2 & 3 \end{bmatrix}, y_1 = 1$$

yellkey.com/dark

Observation 2

$$\vec{x_2}^T = \begin{bmatrix} 4 & -1 \end{bmatrix}, y_2 = 0$$

Cross Entropy Loss – Your Turn (Solution)

$$-y_i \log(\sigma(\vec{x_i}^T \vec{\beta})) - (1 - y_i) \log(1 - \sigma(\vec{x_i}^T \vec{\beta}))$$

Compute the cross entropy loss for both of the following observations (individually).

$$\vec{\hat{\beta}}^T = \begin{bmatrix} 1 & 2 \end{bmatrix}$$

Observation 1

$$\vec{x_1}^T = \begin{bmatrix} -2 & 3 \end{bmatrix}, y_1 = 1$$

$$\begin{aligned} \sigma(\vec{x_1}^T \vec{\hat{\beta}}) &= \sigma(1 \cdot (-2) + 2 \cdot 3) \\ &= \sigma(4) \\ \text{loss}_1 &= -1 \cdot \log(\sigma(4)) - 0 \cdot \log(1 - \sigma(4)) \\ &= \boxed{-\log(\sigma(4))} \end{aligned}$$

Observation 2

$$\vec{x_2}^T = \begin{bmatrix} 4 & -1 \end{bmatrix}, y_2 = 0$$

$$\begin{aligned} \sigma(\vec{x_2}^T \vec{\hat{\beta}}) &= \sigma(1 \cdot 4 + 2 \cdot (-1)) \\ &= \sigma(2) \\ \text{loss}_2 &= -0 \cdot \log(\sigma(2)) - 1 \cdot \log(1 - \sigma(2)) \\ &= \boxed{-\log(1 - \sigma(2))} \end{aligned}$$

Minimizing Empirical Risk using Cross Entropy Loss

Our empirical risk over our entire dataset, then, is:

$$R(\vec{\beta}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\sigma(\vec{x_i}^T\vec{\beta})) + (1-y_i)\log(1-\sigma(\vec{x_i}^T\vec{\beta}))\right]$$

Let's go back to the **NBA dataset from last lecture**.

- Previously, we found the $\hat{\vec{\beta}}$ that minimized mean squared loss.
- Let's now find the $\hat{\vec{\beta}}$ that minimizes mean cross entropy loss. The process is very similar.

(demo)

Choices for Modeling with Logistic Regression

As per usual, when we want to do some sort of prediction, we need to

1. Pick a model
2. Pick a loss
3. Minimize empirical risk for the given model and loss

It's important to note that, for the **logistic regression model** $\hat{y} = P(Y = 1|\vec{x}) = \sigma(\vec{x}^T \hat{\vec{\beta}})$ we can **choose** to use L2 loss or cross entropy loss. These are different optimization problems, and hence will lead to different $\hat{\vec{\beta}}$. **In practice, though, we use cross entropy.**

This is similar to earlier in the course, when our goal was to find a single **summary statistic** for our data, i.e. $\hat{y}_i = c$. Both L1 and L2 loss yielded different empirical risks, and hence, different solutions.

$$R(c) = \frac{1}{n}\sum_{i=1}^{n} |y_i - c| \implies \boxed{\hat{c} = \text{median}(y)} \qquad R(c) = \frac{1}{n}\sum_{i=1}^{n} (y_i - c)^2 \implies \boxed{\hat{c} = \bar{y}}$$

Choices for Modeling with Logistic Regression

As per usual, when we want to do some sort of prediction, we need to

1. Pick a model
2. Pick a loss
3. Minimize empirical risk for the given model and loss

It's important to note that, for the **logistic regression model** $\hat{y} = P(Y = 1|\vec{x}) = \sigma(\vec{x}^T \hat{\vec{\beta}})$ we can **choose** to use L2 loss or cross entropy loss. These are different optimization problems, and hence will lead to different $\hat{\vec{\beta}}$. **In practice, though, we use cross entropy.**

This is similar to earlier in the course, when our goal was to find a single **summary statistic** for our data, i.e. $\hat{y}_i = c$. Both L1 and L2 loss yielded different empirical risks, and hence, different solutions.

However, unlike with summary statistics where there are reasons to use either L1 or L2, **cross entropy loss is strictly better than MSE for logistic regression** (due to its motivations in probability and information theory).

Classification

Classification

Our motivation for performing logistic regression was to predict **categorical labels.** Specifically, we were looking to perform **binary classification**, i.e. classification where our outputs are **1 or 0**.

win or lose

disease or no disease

spam or ham

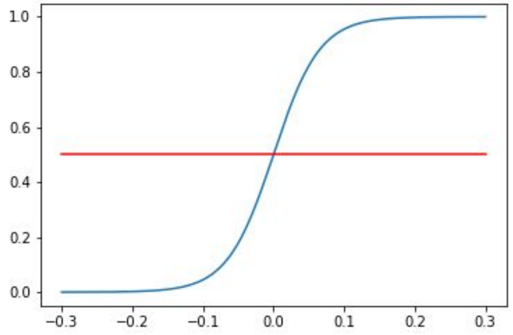
However, **the output of logistic regression is a continuous value** in the range [0, 1], which we interpret as a probability – specifically, $\hat{y} = P(Y = 1|\vec{x})$.

In order to predict a 1 or 0, we pair our logistic model with a **decision rule**, or **classification rule**.

Decision Rules

Given an observation $\vec{x}$, the following **decision rule** outputs 1 or 0, depending on the probability that our model assigns to $\vec{x}$ belonging to class 1:

$$\text{classify}(\vec{x}) = \begin{cases} 1, & P(Y = 1|\vec{x}) \geq 0.5 \\ 0, & P(Y = 1|\vec{x}) < 0.5 \end{cases}$$

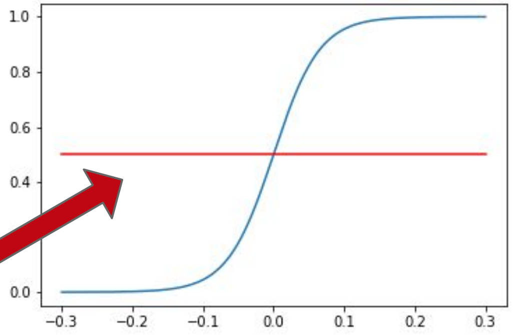


- Note: We **don't need** to set our **threshold** to 0.5. Depending on the type of errors we want to minimize, we can increase or decrease it.
- Furthermore, depending on the domain, we may want to choose not to provide a classification at all, if our probability of being in class 1 is too close to 0.5.

Decision Rules

Given an observation $\vec{x}$, the following **decision rule** outputs 1 or 0, depending on the probability that our model assigns to $\vec{x}$ belonging to class 1:

$$\text{classify}(\vec{x}) = \begin{cases} 1, & P(Y=1|\vec{x}) \geq 0.5 \\ 0, & P(Y=1|\vec{x}) < 0.5 \end{cases}$$

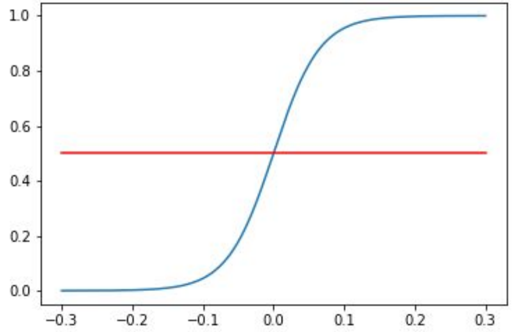


- Note: We **don't need** to set our **threshold** to 0.5. Depending on the type of errors we want to minimize, we can increase or decrease it.
- Furthermore, depending on the domain, we may want to choose not to provide a classification at all, if our probability of being in class 1 is too close to 0.5.

Decision Rules

Given an observation $\vec{x}$, the following **decision rule** outputs 1 or 0, depending on the probability that our model assigns to $\vec{x}$ belonging to class 1:

$$\text{classify}(\vec{x}) = \begin{cases} 1, & P(Y = 1|\vec{x}) \geq 0.5 \\ 0, & P(Y = 1|\vec{x}) < 0.5 \end{cases}$$



In the case where we do set our threshold to be 0.5, the following classifier is equivalent to the above:

$$\text{classify}(\vec{x}) = \begin{cases} 1, & \vec{x}^T \vec{\hat{\beta}} \geq 0 \\ 0, & \vec{x}^T \vec{\hat{\beta}} < 0 \end{cases}$$

since $\sigma(0) = \frac{1}{1 + e^{-0}} = 0.5$

Accuracy and Error

Now that we actually have our classifier, let's try and quantify how well it performs.

The two most basic evaluation metrics are **accuracy** (proportion of points classified correctly) and **error** (proportion of points classified incorrectly).

$$\text{accuracy} = \frac{\#\text{ of points classified correctly}}{\#\text{ points total}}$$

$$\text{error} = 1 - \text{accuracy} = \frac{\#\text{ of points classified incorrectly}}{\#\text{ points total}}$$

Error is just the complement of accuracy – it doesn't give us any new information.

Pitfalls of Accuracy – Your Turn

Suppose we're trying to build a classifier to filter **spam emails**.

- Specifically, each email is either **spam** (1) or **ham** (0).
    - "Ham" just means "not spam".

Let's say we have 100 emails, of which **5** are truly **spam**, and the remaining **95** are **ham**.

- Your friend, who hasn't taken Data 100, suggests you **classify every email as ham**.
- What is the accuracy of your friend's classifier?
- Is accuracy a good metric of this classifier's performance?

Pitfalls of Accuracy

Suppose we're trying to build a classifier to filter **spam emails**.

- Specifically, each email is either **spam** (1) or **ham** (0).
  - "Ham" just means "not spam".

Let's say we have 100 emails, of which **5** are truly **spam**, and the remaining **95** are **ham**.

- Your friend, who hasn't taken Data 100, suggests you **classify every email as ham**.
- What is the accuracy of your friend's classifier? **95%.**
- Is accuracy a good metric of this classifier's performance?
  - No! While our accuracy was high, **we didn't actually catch any spam emails**. There were truly 5 spam emails, but we said they were all non-spam. This would be a terrible classifier in practice.
  - **Alternatively,** we could classify every email as spam. That *would* catch every spam email, but it would also classify 95 non-spam emails as spam. Not good!
  - This suggests that we need to be looking at not just accuracy, but the **number of spam emails we actually catch**, and **types of misclassifications** we make.

True Positives, True Negatives, False Positives, False Negatives

Our observed truth is either 1 or 0, and our prediction for each observation is either 1 or 0. This means there are **4 cases** to consider, each of which has its own name:

		Truth	
Prediction		**1**	**0**
	1	**TP**: True Positive	**FP**: False Positive
	0	**FN**: False Negative	**TN**: True Negative

We can define accuracy in terms of the above quantities:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{n} \qquad n = \#\text{ points total}$$

But, as we mentioned on the previous slide, we want to look at more than just accuracy.

Precision and Recall

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{n}$$

What proportion of points did our classifier classify correctly?
Doesn’t tell the full story, especially in cases with high class imbalance.

		Truth	
		1	0
Prediction	1	TP	FP
	0	FN	TN

$$\text{precision} = \frac{TP}{TP + FP}$$

Of all observations that were predicted to be 1, what proportion were actually 1?
How precise is our classifier? **Penalizes false positives.**

$$\text{recall} = \frac{TP}{TP + FN}$$

Of all observations that were actually 1, what proportion did we predict to be 1?
How good is our classifier at detecting points belonging to class 1? **Penalizes false negatives.**

Precision and Recall

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{n}$$

What proportion of points did our classifier classify correctly?
Doesn’t tell the full story, especially in cases with high class imbalance.

		Truth	
		1	**0**
Prediction	**1**	**TP**	**FP**
	0	**FN**	**TN**

$$\text{precision} = \frac{TP}{TP + FP}$$

Of all observations that were predicted to be 1, what proportion were actually 1?
How precise is our classifier? **Penalizes false positives.**

$$\text{recall} = \frac{TP}{TP + FN}$$

Of all observations that were actually 1, what proportion did we predict to be 1?
How good is our classifier at detecting points belonging to class 1? **Penalizes false negatives.**

Precision and Recall

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{n}$$

What proportion of points did our classifier classify correctly?
Doesn’t tell the full story, especially in cases with high class imbalance.

		Truth	
		1	0
Prediction	1	TP	FP
	0	FN	TN

$$\text{precision} = \frac{TP}{TP + FP}$$

Of all observations that were predicted to be 1, what proportion were actually 1?
How precise is our classifier? **Penalizes false positives.**

$$\text{recall} = \frac{TP}{TP + FN}$$

Of all observations that were actually 1, what proportion did we predict to be 1?
How good is our classifier at detecting points belonging to class 1? **Penalizes false negatives.**

Precision and Recall

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{n}$$

What proportion of points did our classifier classify correctly?
Doesn't tell the full story, especially in cases with high class imbalance.

		Truth	
		1	0
Prediction	1	TP	FP
	0	FN	TN

$$\text{precision} = \frac{TP}{TP + FP}$$

Of all observations that were predicted to be 1, what proportion were actually 1?
How precise is our classifier? **Penalizes false positives.**

Our friend's **"call-all-ham"** classifier from before had an accuracy of 95%, but an **undefined precision** (0/0) and **0% recall** (0/5). Not looking so good now...

$$\text{recall} = \frac{TP}{TP + FN}$$

Of all observations that were actually 1, what proportion did we predict to be 1?
How good is our classifier at detecting points belonging to class 1? **Penalizes false negatives.**

Trade-off Between Precision and Recall

- Precision penalizes false positives, and recall penalizes false negatives.
- We can **achieve 100% recall** by making our classifier output “1”, regardless of the input.
  - For example, predicting all emails as spam.
  - We would have no false negatives, but **many false positives**, and so our **precision would be low**.
- This suggests that there is a **tradeoff between precision and recall – they are inversely related**. Ideally, both would be near 100%, but that’s unlikely to happen.
- We can adjust our classification threshold to suit our needs, depending on the domain.
  - **Higher threshold** – fewer false positives. Increases **precision**.
  - **Lower threshold** – fewer false negatives. Increases **recall**.

Prediction \ Truth	1	0
1	**TP**	**FP**
0	**FN**	**TN**

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

If we’re trying to classify whether or not people have a **disease**, which of these metrics do we care more about?

Summary

- **Logistic regression** is used to **model** the **probability** that an observation belongs to class 1, i.e. $\hat{y} = P(Y = 1|\vec{x})$.
  - We do so by applying the logistic function to a linear function of our observation: $\hat{y} = P(Y = 1|\vec{x}) = \sigma(\vec{x}^T \hat{\vec{\beta}})$.
  - In order to find $\hat{\vec{\beta}}$, we choose a loss function. L2 can be used, but it has a few problems (non-convex, doesn't penalize heavily enough).
  - Most typically used for logistic regression is cross entropy loss.
- In order to **actually classify** things using logistic regression, we need to pair it with a **decision boundary**.
  - Typically, we apply a threshold on the probability that our observation belongs to a certain class.
- There are several metrics to evaluate the **effectiveness** of our classifier.
  - Accuracy and error work decently, but don't always tell the full story.
  - Precision and recall tell us more about the types of predictions our classifier makes. There's a tradeoff between the two.