Classification: Logistic Regression

Introduction to Data Science
Spring 1404

Yadollah Yaghoobzadeh

From:
DS100 UC Berkeley

Goals for this Lecture

Moving away from linear regression – it's time for a new type of model

- Introduce a new task: classification
- Deriving a new model to handle this task

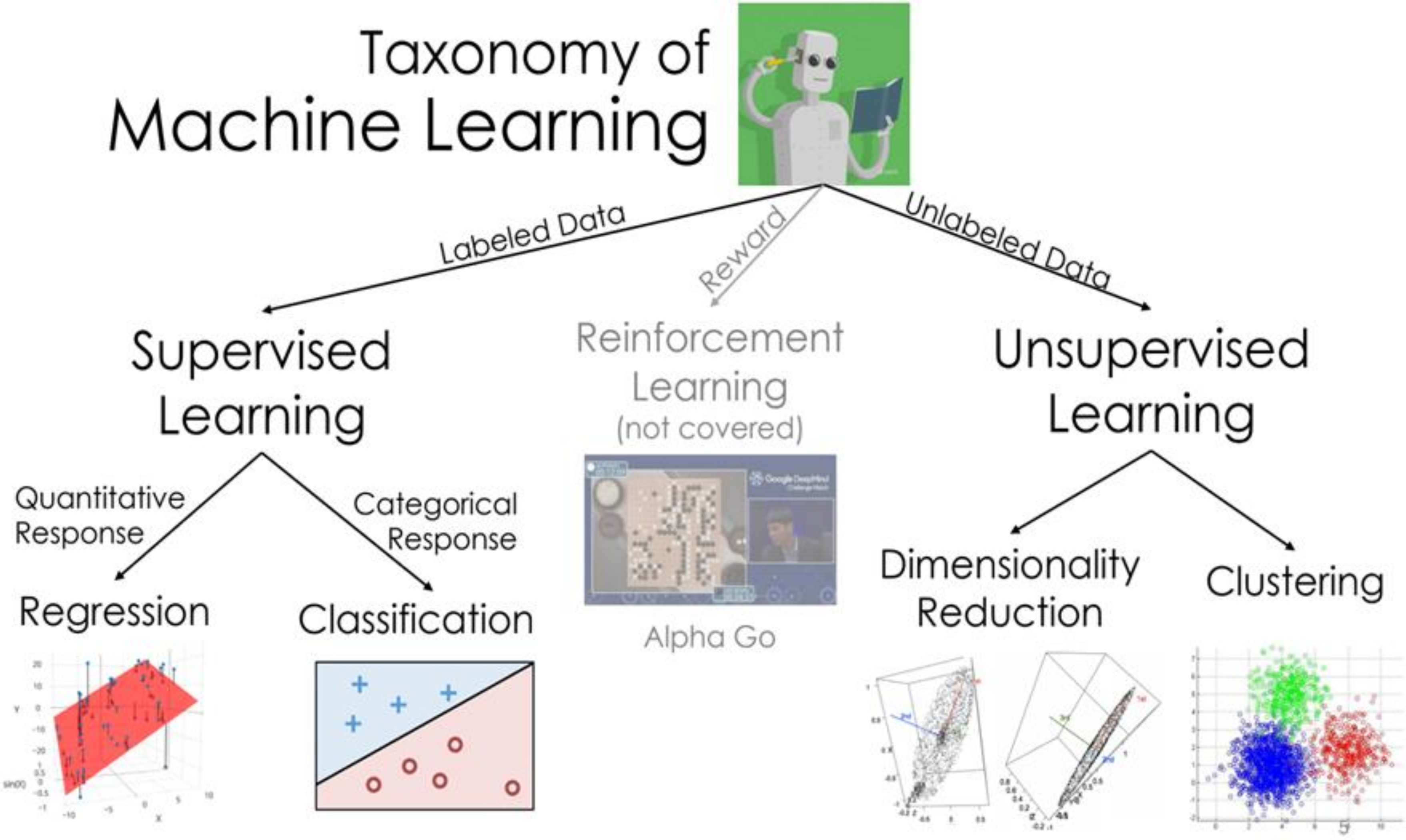
Agenda

- Regression vs. Classification
- The Logistic Regression Model
- Cross-Entropy Loss

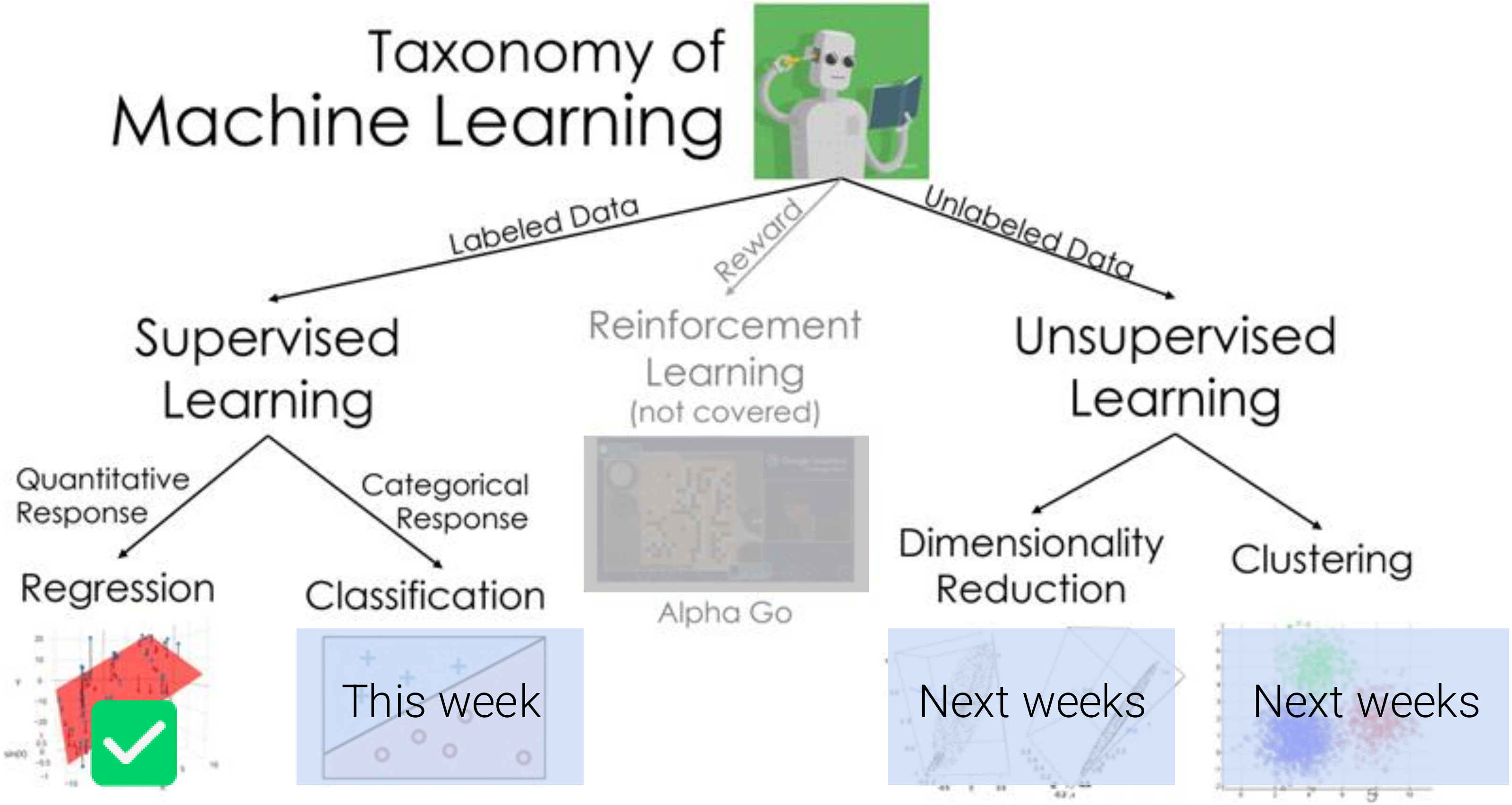
Agenda

- **Regression vs. Classification**
- The Logistic Regression Model
- Cross-Entropy Loss

Beyond Regression



Beyond Regression



So Far: Regression

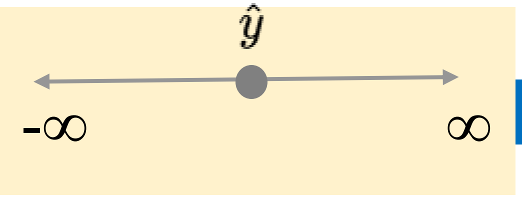
In regression, we use unbounded numeric features to predict an *unbounded numeric output.*

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

Input: numeric features

$$x^\top \theta$$

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

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

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)

Agenda

- Regression vs. Classification
- **The Logistic Regression Model**
- Cross-Entropy Loss

The games Dataset

The games dataset describes the win/loss results of basketball teams.

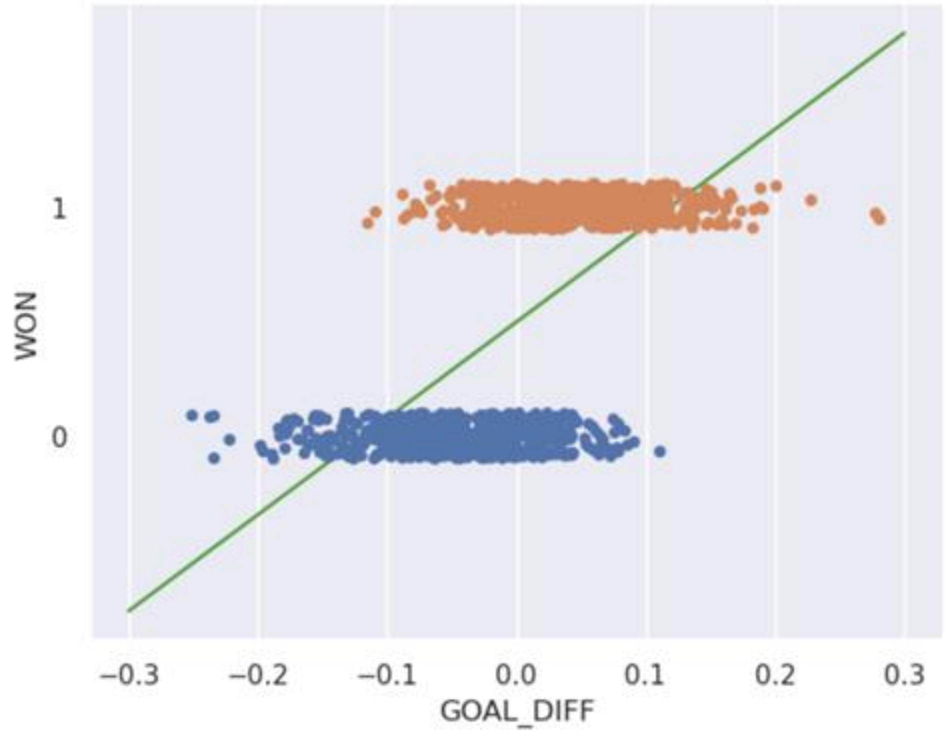
GAME_ID	TEAM_NAME	MATCHUP	WON	GOAL_DIFF
21700001	Boston Celtics	BOS @ CLE	0	-0.049
21700002	Golden State Warriors	GSW vs. HOU	0	0.053
21700003	Charlotte Hornets	CHA @ DET	0	-0.030
21700004	Indiana Pacers	IND vs. BKN	1	0.041
21700005	Orlando Magic	ORL vs. MIA	1	0.042

Difference in field goal success rate between teams

If a team won their game, we say they are in “Class 1”

Why Not Least Squares Linear Regression?

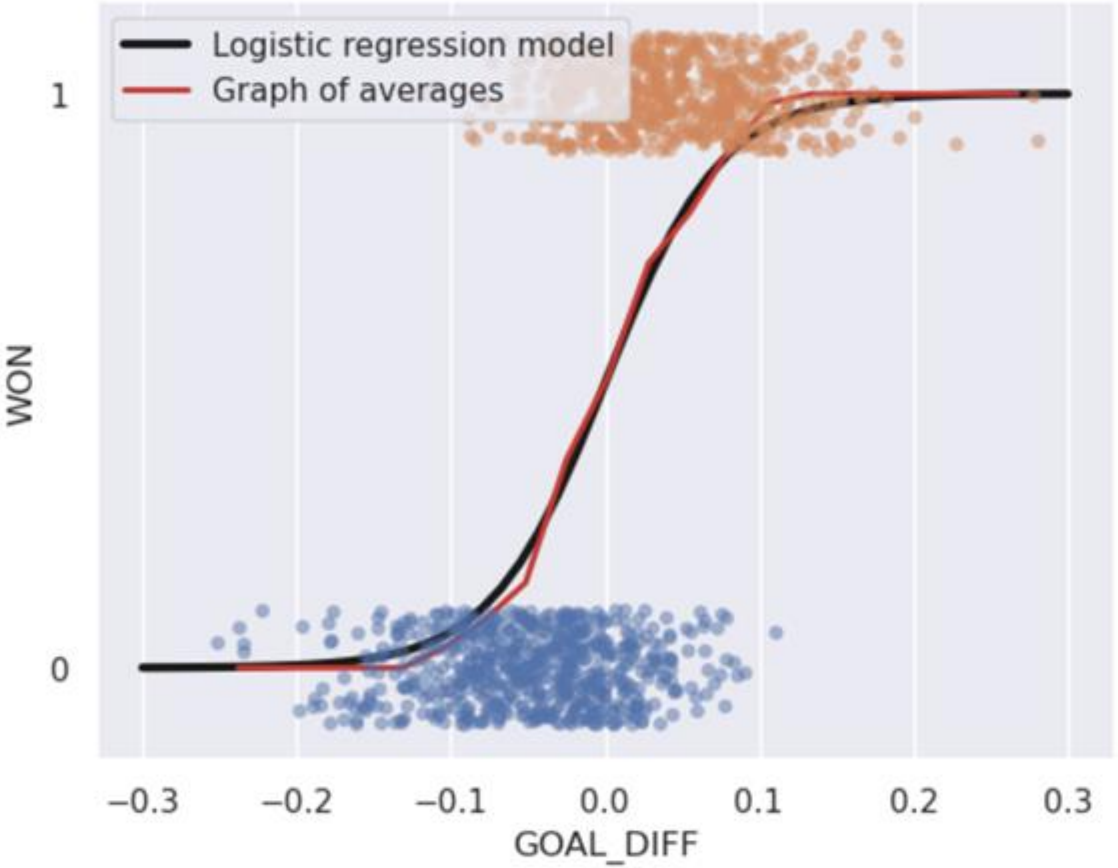
I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail. – Abraham Maslow, *The Psychology of Science*



Problems:

- The output $\hat{y}$ can be outside the label range {0, 1}.
- Some outputs can't be interpreted: what does a class of "-2.3" mean?

Logistic Regression Model



$$P(Y = 1 \mid x) = \frac{1}{1+e^{-x^\top \theta}}$$

$$= \frac{1}{1+e^{-(\theta_0+\theta_1 x_1+\ldots+\theta_p x_p)}}$$

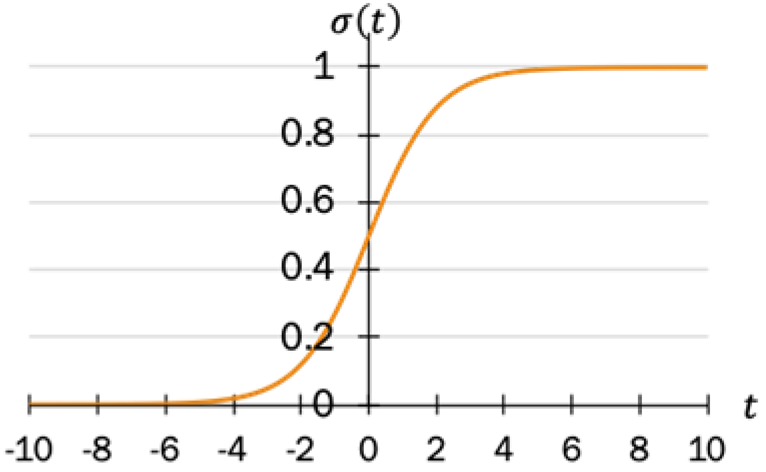
To predict a probability:

- Compute a linear combination of the features, $x^\top \theta$
- Apply the **sigmoid function** $\sigma(x^\top \theta)$

The Sigmoid Function

The S-shaped curve is formally known as the **sigmoid function**.

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



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

Domain

$$-\infty < t < \infty$$

Range

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

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

Derivative

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

The Sigmoid Converts Numerical Features to Probabilities

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

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

Model: linear combination transformed by **activation function**

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

Decision rule

Output: **class**

In logistic regression, the sigmoid transforms a linear combination of numerical features into a **probability.**

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

Formalizing the Logistic Regression Model

Our main takeaways of this section:

- Fit the "S" curve as best as possible.
- The curve models probability: P(Y = 1 | x).
- Assume log-odds is a linear combination of x and ϴ.

Putting it all together:

$$\underbrace{\hat{P}_{\theta}(Y = 1|x)}_{\text{Estimated probability that given the features x, the response is 1}} = \underbrace{\frac{1}{1 + e^{-x^T\theta}}}_{\text{Logistic function } \sigma(\;) \text{ at the value } x^T\theta}$$

The logistic regression model is most commonly written as follows:

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

Looks like linear regression. Now wrapped with $\boldsymbol{\sigma}(\;)$!

Properties of the Logistic Model

Consider a logistic regression model with one feature and an intercept term:

$$p = P(Y = 1 \mid x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

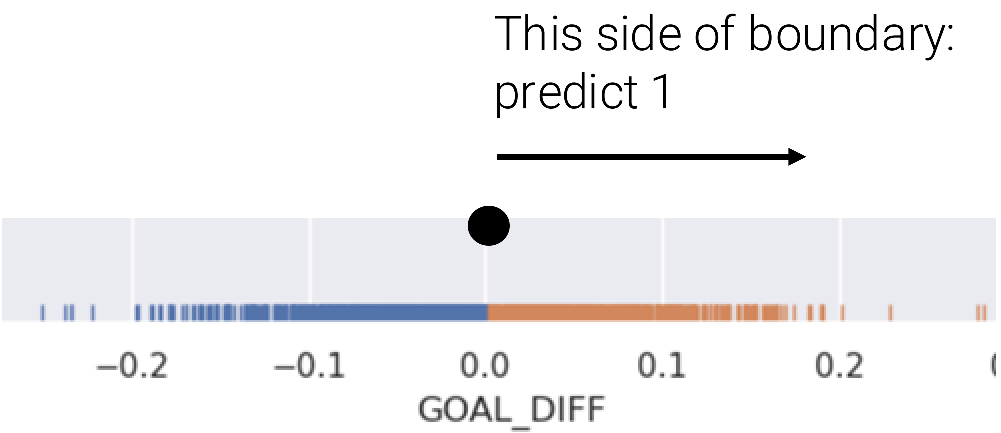
Properties:

- θ_0 controls the position of the curve along the horizontal axis.
- The magnitude of θ_1 controls the "steepness" of the sigmoid.
- The sign of θ_1 controls the orientation of the curve.

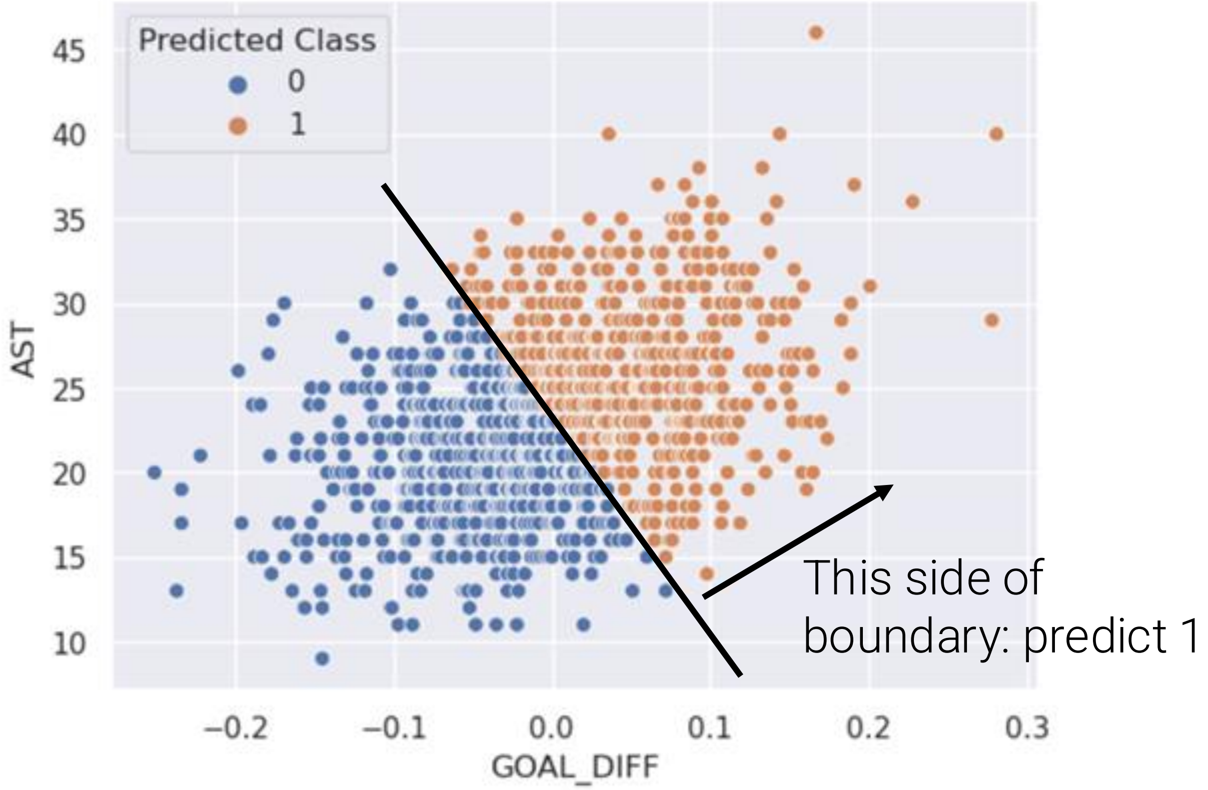
Decision Boundaries

A **decision boundary** describes the "line" the splits the data into classes based on its **features**.

- For logistic regression, the decision boundary is a **hyperplane**: a linear combination of the features in p-dimensions.



1 feature: decision boundary is a 1D point



2 features: decision boundary is a 2D line

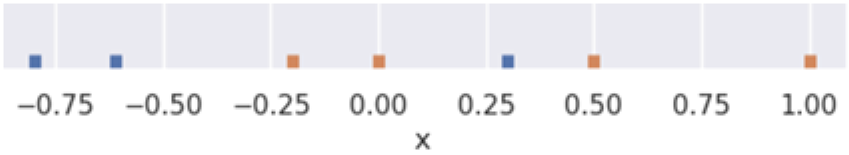
Linear Separability

A classification dataset is said to be **linearly separable** if there exists a hyperplane **among input features x** that separates the two classes y.

If there is one feature, look for a point that separates the classes.

separable

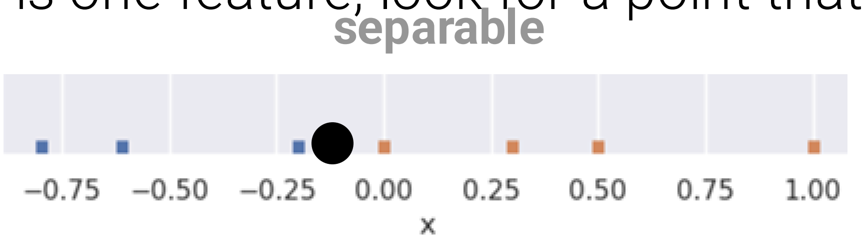
not separable

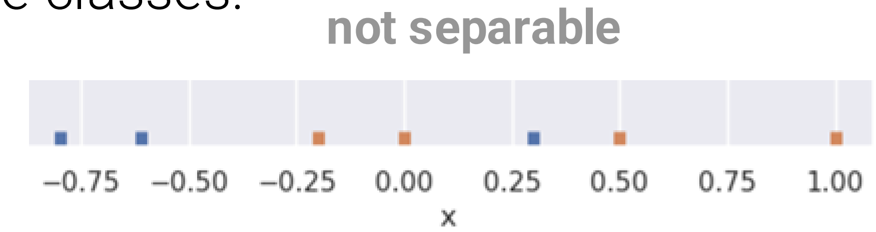


Linear Separability

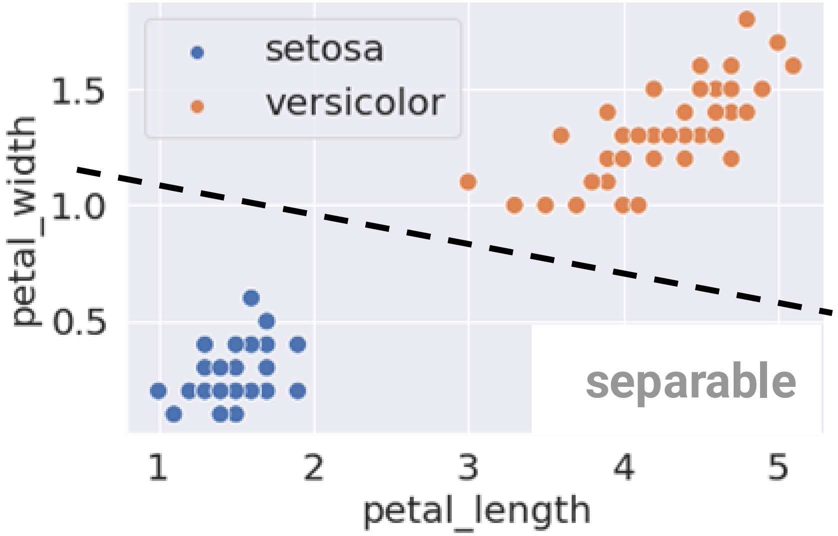
A classification dataset is said to be **linearly separable** if there exists a hyperplane **among input features x** that separates the two classes y.

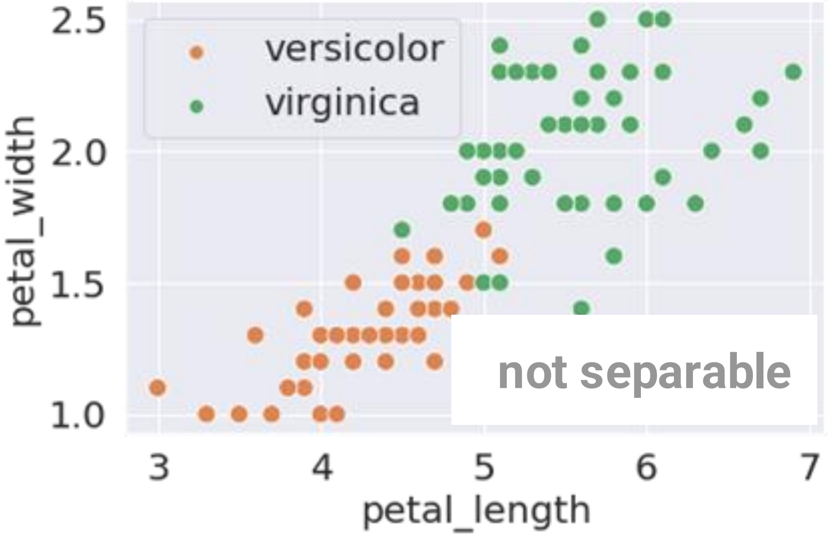
If there is one feature, look for a point that separates the classes.





If there are two features, look for a line that separates the classes





Cross-Entropy Loss

- Regression vs. Classification
- The Logistic Regression Model
- **Cross-Entropy Loss**

The Modeling Process

	Regression ($y \in \mathbb{R}$)	Classification ($y \in \{0, 1\}$)
1. Choose a model	Linear Regression $\hat{y} = f_\theta(x) = x^T\theta$	Logistic Regression $\hat{P}_\theta(Y = 1\|x) = \sigma(x^T\theta)$
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 time)

The Modeling Process

1600133

	Regression ($y \in \mathbb{R}$)	Classification ($y \in \{0, 1\}$)
1. Choose a model	Linear Regression $\hat{y} = f_\theta(x) = x^T\theta$	Logistic Regression $\hat{P}_\theta(Y = 1\|x) = \sigma(x^T\theta)$
2. Choose a loss function	Squared Loss or Absolute Loss	Can squared loss still work?
3. Fit the model	Regularization Sklearn/Gradient descent	Regularization Sklearn/Gradient descent
4. Evaluate model performance	R^2, Residuals, etc.	?? (next time)

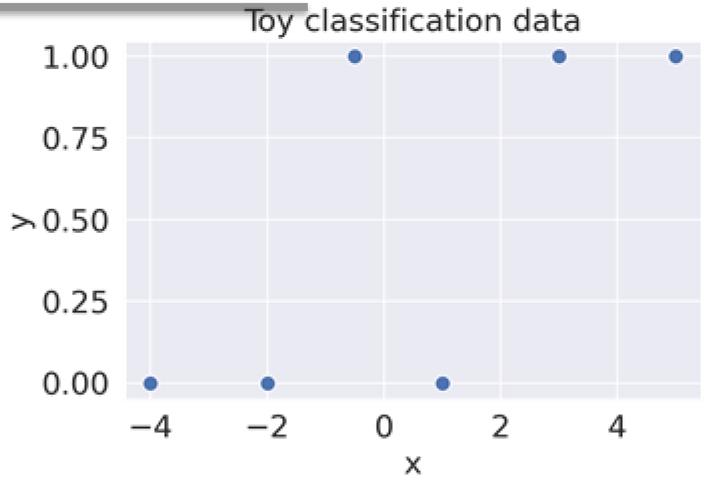
Toy Dataset: L2 Loss

Logistic Regression model:

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

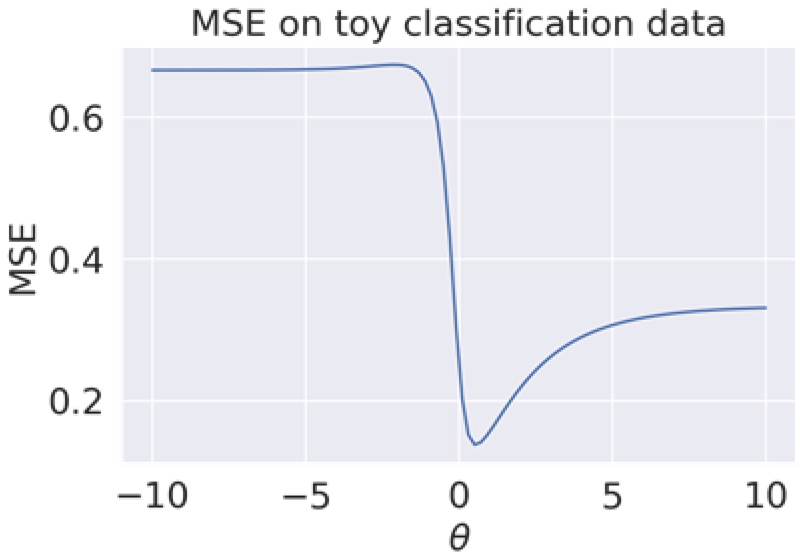
Assume no intercept.
So x, θ both scalars.

	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



Mean Squared Error:

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



The MSE loss surface for logistic regression has many issues!

Choosing a Different Loss Function

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

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.

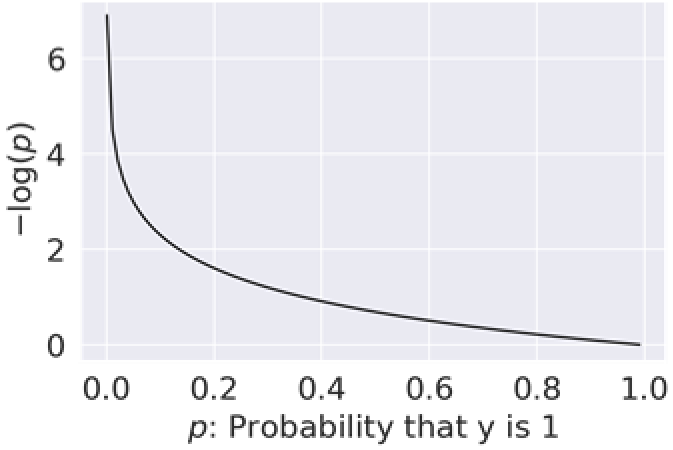
In other words, the behavior we need from our loss function depends on the value of the true class, y.

Cross-Entropy Loss

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

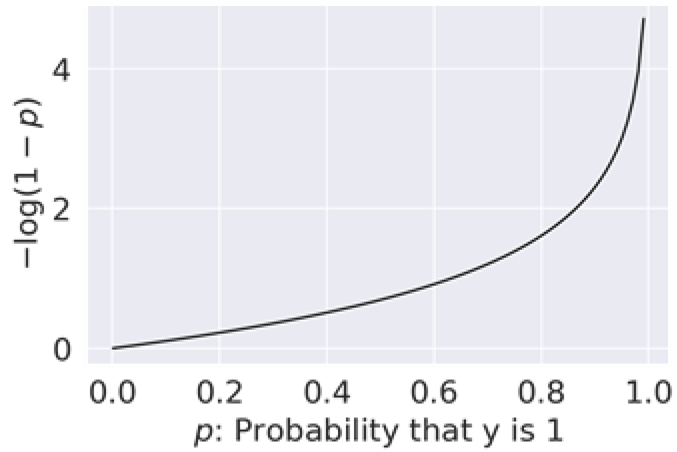
The **cross-entropy loss** is defined as:

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



For y = 1,

- p → 0: ∞ loss
- p → 1: zero loss



For y = 0,

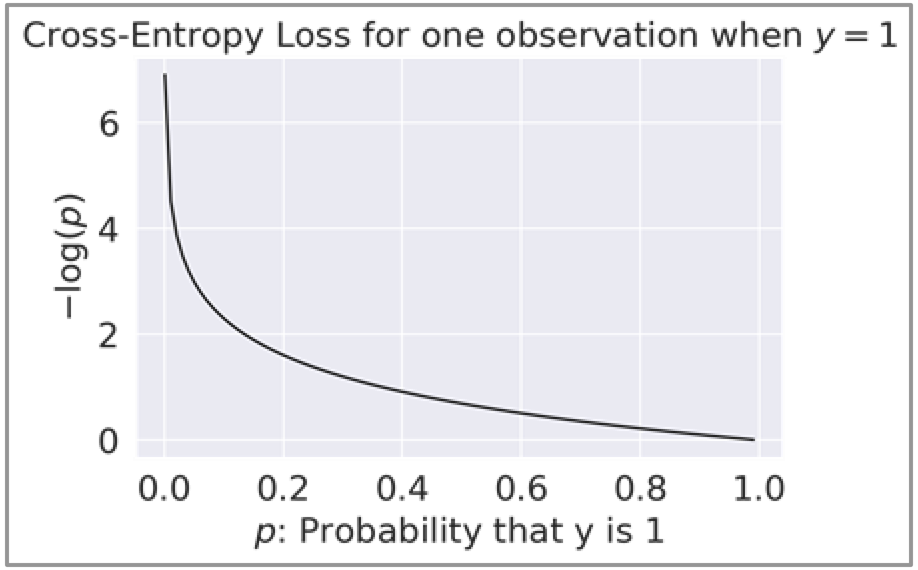
- p → 0: zero loss
- p → 1: ∞ loss

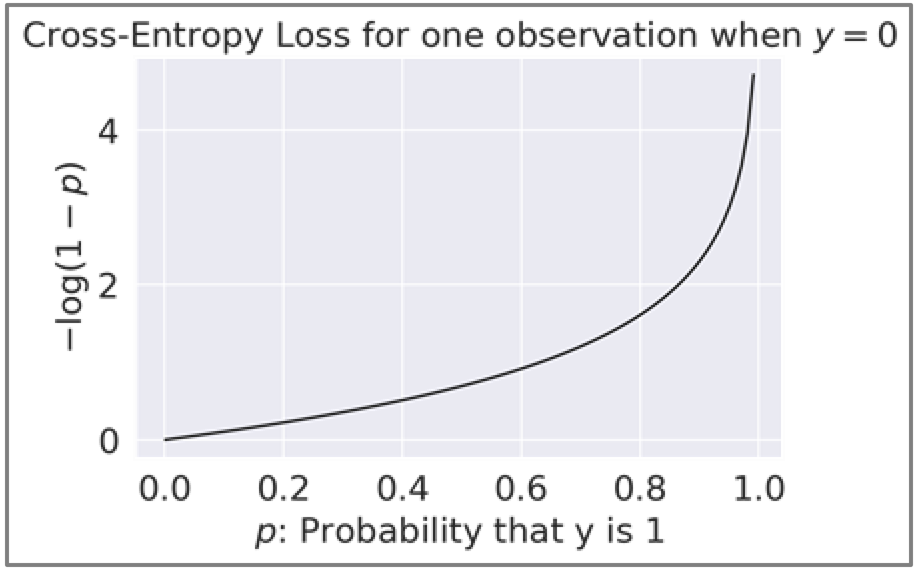
Cross-Entropy Loss: Two Loss Functions In One!

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:

$$\text{CE loss} = \begin{cases} -\log(p), & \text{if } \; y = 1 \\ -\log(1-p), & \text{if } \; y = 0 \end{cases} \;\Longrightarrow\; \underbrace{\boxed{-}}_{\text{makes loss positive}} (\underbrace{y\log(p)}_{\text{for y = 1, only this term stays}} + \underbrace{(1-y)\log(1-p)}_{\text{for y = 0, only this term stays}})$$





Empirical Risk: Average Cross-Entropy Loss

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

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

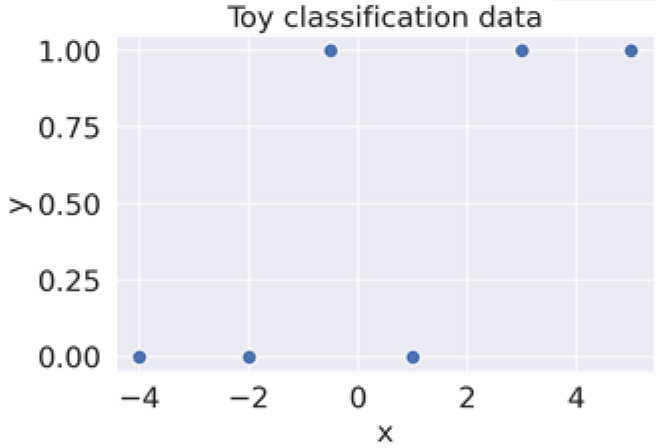
[Recall our model is $\hat{y}_i = p_i = \sigma\left(X_i^T\theta\right)$]

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

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

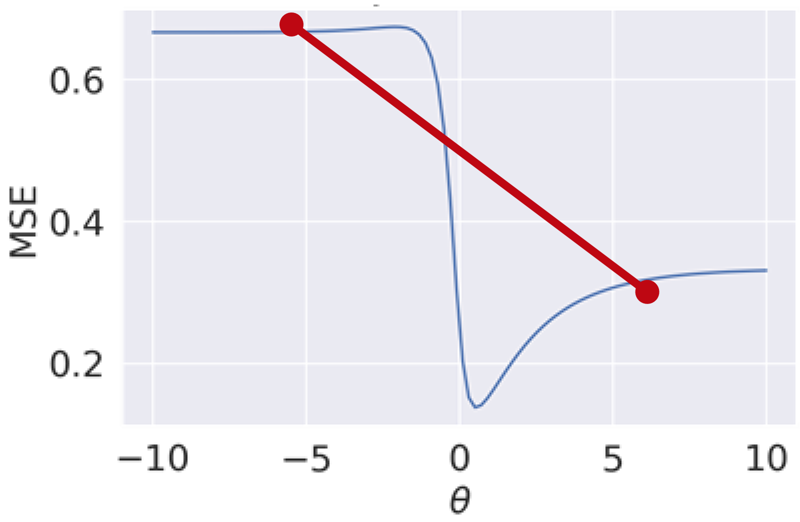
Convexity Proof By Picture

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



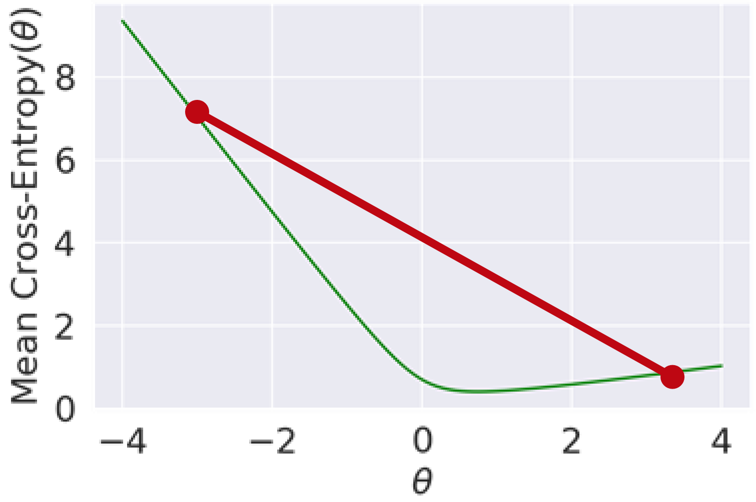
	x	y
0	-4.0	0
1	-2.0	0
2	-0.5	1
3	1.0	0
4	3.0	1
5	5.0	1

Squared Loss Surface



A straight line crosses the curve
Non-convex

Cross-Entropy Loss Surface



Convex!

Maximum Likelihood Estimation

It may have seemed like we just pulled cross-entropy loss out of thin air.

CE loss is justified by a probability analysis technique called maximum likelihood estimation. Read on if you would like to learn more.

Recorded walkthrough: link.

We Did it!

	Regression ($y \in \mathbb{R}$)	Classification ($y \in \{0, 1\}$)
1. Choose a model	Linear Regression $\hat{y} = f_\theta(x) = x^T\theta$	Logistic Regression $\hat{P}_\theta(Y = 1 \mid x) = \sigma(x^T\theta)$
2. Choose a loss function	Squared Loss or Absolute Loss	Average Cross-Entropy Loss $-\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\sigma(X_i^T\theta) + (1 - y_i)\log(1 - \sigma(X_i^T\theta))\right)$
3. Fit the model	Regularization Sklearn/Gradient descent	Regularization Sklearn/Gradient descent
4. Evaluate model performance	R^2, Residuals, etc.	?? (next time)

Why More Performance Metrics?

We’ve already introduced cross-entropy loss – why do we need additional ways of assessing how well our models perform?

- In linear regression, we made numerical predictions and used a loss function to determine how “good” these predictions were.
- In logistic regression, our ultimate goal is to *classify* data – we are more concerned with whether or not each datapoint was assigned the correct class using the decision rule.

The performance metrics we are about to introduce will assess the quality of classifications, not the predicted probabilities.

Classifier Accuracy

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

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

The most basic evaluation metric for a classifier is **accuracy**.

```python
def accuracy(X, Y):
    return np.mean(model.predict(X) == Y)

accuracy(X, Y) # 0.794
```

```python
model.score(X, Y) # 0.794
```

(sklearn documentation)

While widely used, the accuracy metric is **not so meaningful** when dealing with **class imbalance**.

Pitfalls of Accuracy: A Case Study

Suppose we're trying to build a classifier to filter spam emails (Project B!).

- Each email is **spam** (1) or **ham** (0).

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

Your friend ("Friend 1"):

Classify *every* email as **ham** (0).

$$\hat{y} = \text{classify}_{\text{friend}}(x) = 0$$

1. What is the accuracy of your friend's classifier?
2. Is accuracy a good metric of this classifier's performance?

Pitfalls of Accuracy: A Case Study

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

- Each email is **spam** (1) or **ham** (0).

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

Your friend ("Friend 1"):

Classify every email as **ham** (0).

$$\text{accuracy}_1 = \frac{95}{100} = 0.95$$

High accuracy...
...but we detected **none** ⚠️ of the spam!!!

Pitfalls of Accuracy: A Case Study

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

- Each email is **spam** (1) or **ham** (0).

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

Accuracy is not always a good metric for classification, particularly when your data have
class imbalance (e.g., very few 1's compared to 0's).

$$\text{accuracy}_1 = \frac{95}{100} = 0.95$$

High accuracy...
...but we detected **none** ⚠️ of the spam!!!

Types of Classifications

- There are four different classifications that our model might make:
  - True positive: correctly classify a positive point as being positive ($y = 1,\ \hat{y} = 1$)
  - False positive: incorrectly classify a negative point as being positive ($y = 0,\ \hat{y} = 1$)
  - False negative: incorrectly classify a positive point as being negative ($y = 1,\ \hat{y} = 0$)
  - True negative: correctly classify a negative point as being negative ($y = 0,\ \hat{y} = 0$)

"**positive**" means a prediction of **1**.
"**negative**" means a prediction of **0**.

A confusion matrix plots these quantities for a particular classifier and dataset.

Prediction $\hat{y}$

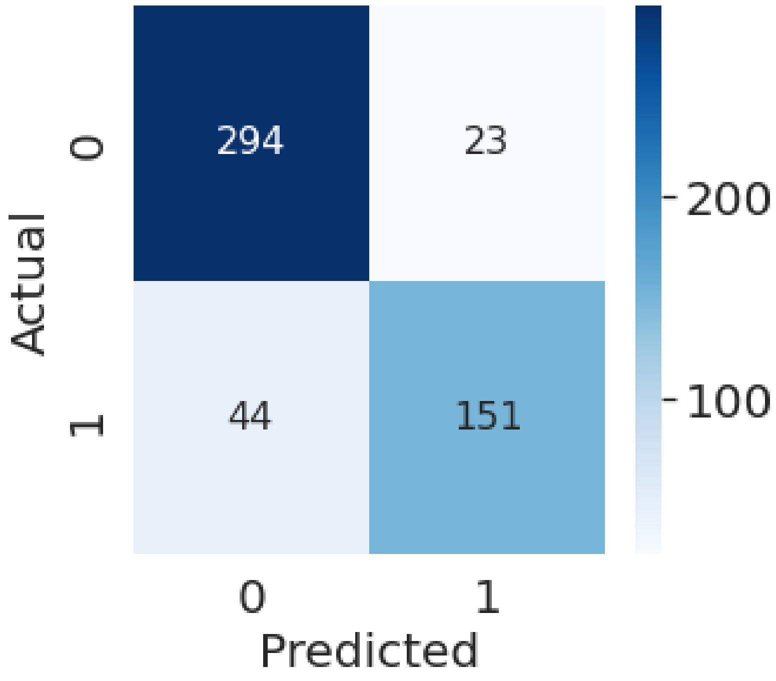
Actual y	**0**	**1**
0	True **negative** (TN)	False **positive** (FP)
1	False **negative** (FN)	True **positive** (TP)

Types of Classifications

A confusion matrix plots these quantities for a particular classifier and dataset.

In `sklearn`:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_true, Y_pred)
```



Accuracy, Precision, and Recall

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

What proportion of points did our classifier classify correctly?

Prediction

Actual	0	1
0	TN	FP
1	FN	TP

Accuracy, Precision, and Recall

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

What proportion of points did our classifier classify correctly?

Precision and recall are two commonly used metrics that measure performance even in the presence of class imbalance.

		Prediction	
		0	1
Actual	0	TN	FP
	1	FN	TP

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

Of all observations that were predicted to be 1, what proportion were actually 1?

- How **precise** is our classifier **when it is positive**?
- Penalizes false positives.

Accuracy, Precision, and Recall

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

What proportion of points did our classifier classify correctly?

	Prediction 0	Prediction 1
Actual 0	TN	FP
Actual 1	FN	TP

Precision and **recall** are two commonly used metrics that measure performance even in the presence of class imbalance.

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

Of all observations that were predicted to be 1, what proportion were actually 1?

- How accurate is our classifier when it is positive?
- Penalizes false positives.

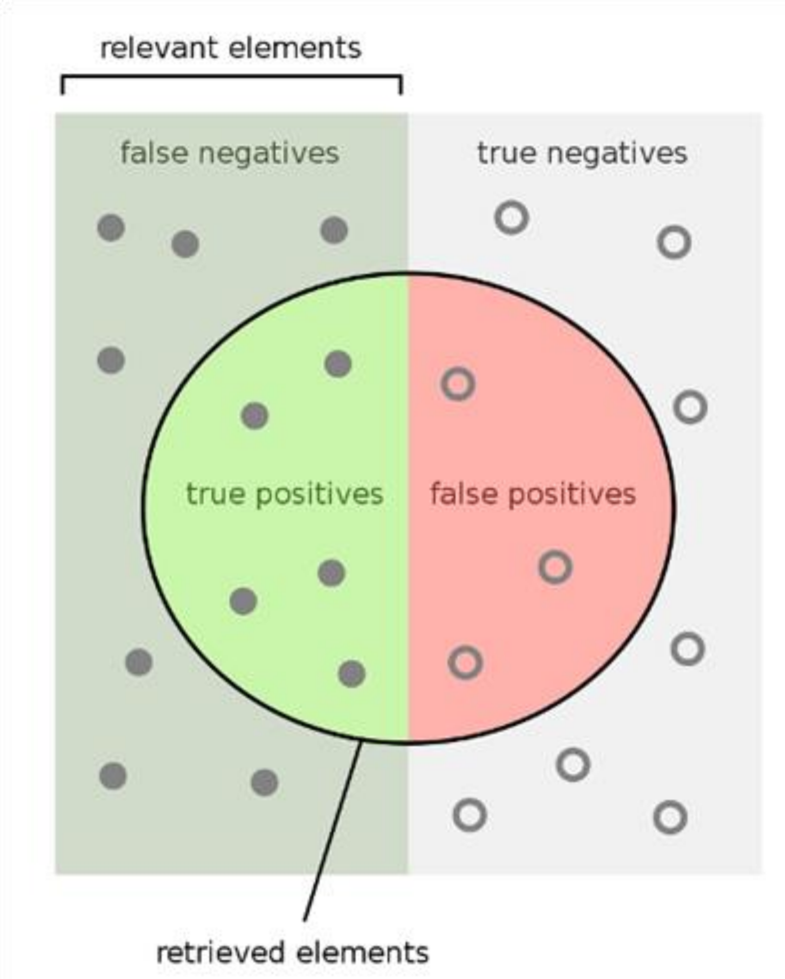
$$\text{recall} = \frac{TP}{TP + FN}$$

Of all observations that were actually 1, what proportion did we predict to be 1? (Also known as sensitivity.)

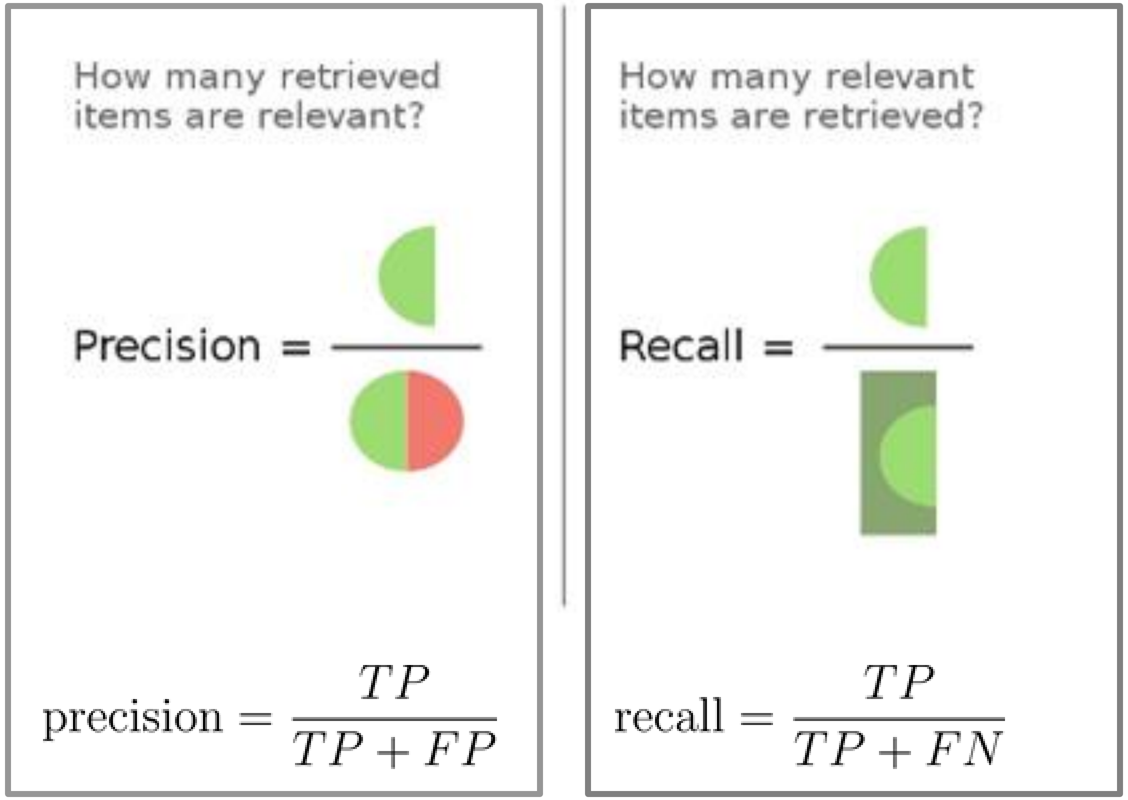
- How **sensitive** is our classifier to **positives**?
- Penalizes false negatives.

One of the Most Valuable Graphics on Wikipedia

(*i.e., true class is 1)



(i.e., positive; predicted class is 1)



[adapted from Wikipedia]

Back to the Spam

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

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

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

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

- Each email is **spam** (1) or **ham** (0).

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

Your friend:

Classify every email as **ham** (0).

$$\text{accuracy}_1 = \frac{95}{100} = 0.95$$

$$\text{precision}_1 = \frac{0}{0+0} = \text{undefined}$$

$$\text{recall}_1 = \frac{0}{0+5} = 0$$

	0	**1**
0	TN: 95	FP: 0
1	FN: 5	TP: 0

Back to the Spam

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

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

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

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

- Each email is **spam** (1) or **ham** (0).

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

	0	**1**
0	TN: 0	FP: 95
1	FN: 0	TP: 5

Your friend:

Classify every email as **ham** (0).

$$\text{accuracy}_1 = \frac{95}{100} = 0.95$$

$$\text{precision}_1 = \frac{0}{0 + 0} = \text{undefined}$$

$$\text{recall}_1 = \frac{0}{0 + 5} = 0$$

Your other friend ("Friend 2"):

Classify every email as **spam** (1).

$$\text{accuracy}_2 = \frac{5}{100} = 0.05$$

$$\text{precision}_2 = \frac{5}{5 + 95} = 0.05$$

Many false positives!

$$\text{recall}_2 = \frac{5}{5 + 0} = 1.0$$

No false negatives!

Precision vs. Recall

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

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

Precision penalizes false positives, and Recall penalizes false negatives.

This suggests that there is a **tradeoff** between precision and recall; they are often inversely related.

- Ideally, both would be near 100%, but that's unlikely to happen.

Which Performance Metric?

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

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

Precision penalizes false positives, and Recall penalizes false negatives.

In many settings, there might be a higher “cost” to missing positive or negative cases.

Some examples:
Detecting if someone tests positive (1) or negative (0) for a disease.
Determining if someone should be sentenced to prison (1) or not (0).
Filtering an email as spam (1) or ham (0).

True and False Positive Rates

Keeping things interesting – two more performance metrics.

$$\text{FPR} = \frac{FP}{FP + TN}$$

False Positive Rate (FPR): out of all datapoints that had Y=0, how many did we classify **incorrectly**?

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

True Positive Rate (TPR): out of all datapoints that had Y=1, how many did we classify correctly? Same as recall.

Actual \ Prediction	0	1
0	TN	FP
1	FN	TP

Adjusting the Classification Threshold

Thresholds

We commonly make decision rules by specifying a **threshold**, T. If the predicted probability is greater than T , predict Class 1. Otherwise, predict Class 0.

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

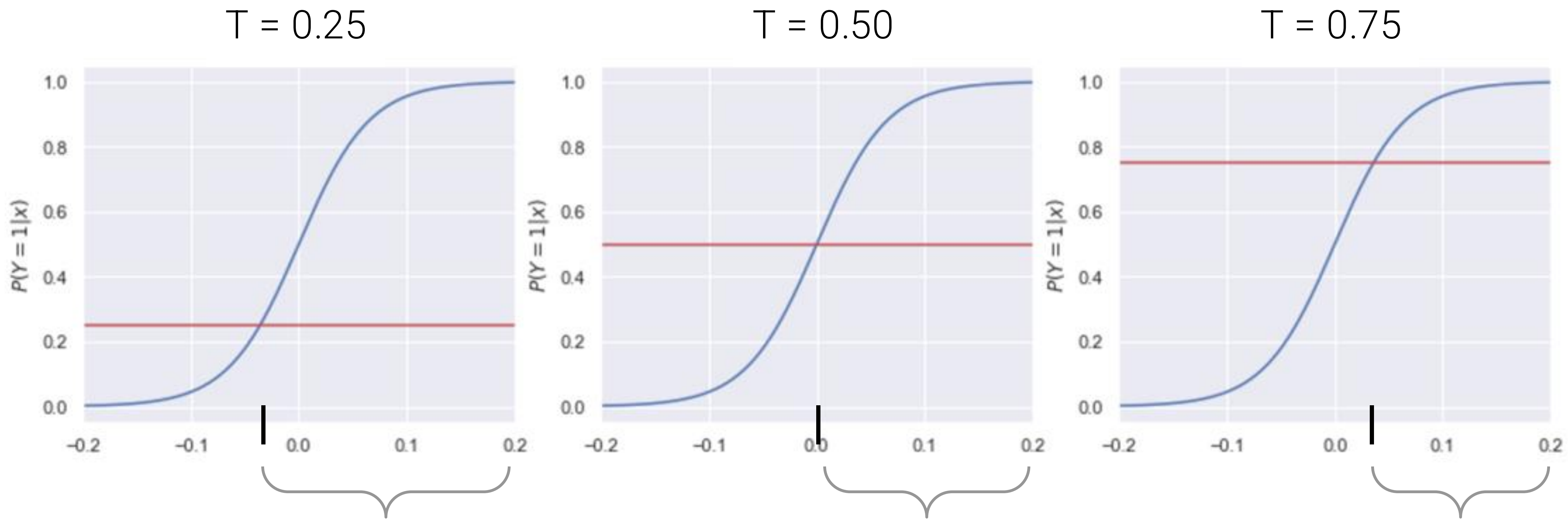
$$\hat{y} = \text{classify}(x) = \begin{cases} \text{Class 1} & p \geq T \\ \text{Class 0} & p < T \end{cases}$$

What happens if we set T to something other than 0.5?

Changing the Threshold

$$\hat{y} = \text{classify}(x) = \begin{cases} \text{Class } 1 & p \geq T \\ \text{Class } 0 & p < T \end{cases}$$

As we increase the threshold T, we “raise the standard” of how confident our classifier needs to be to predict 1 (i.e., “positive”).



These x will all predict 1

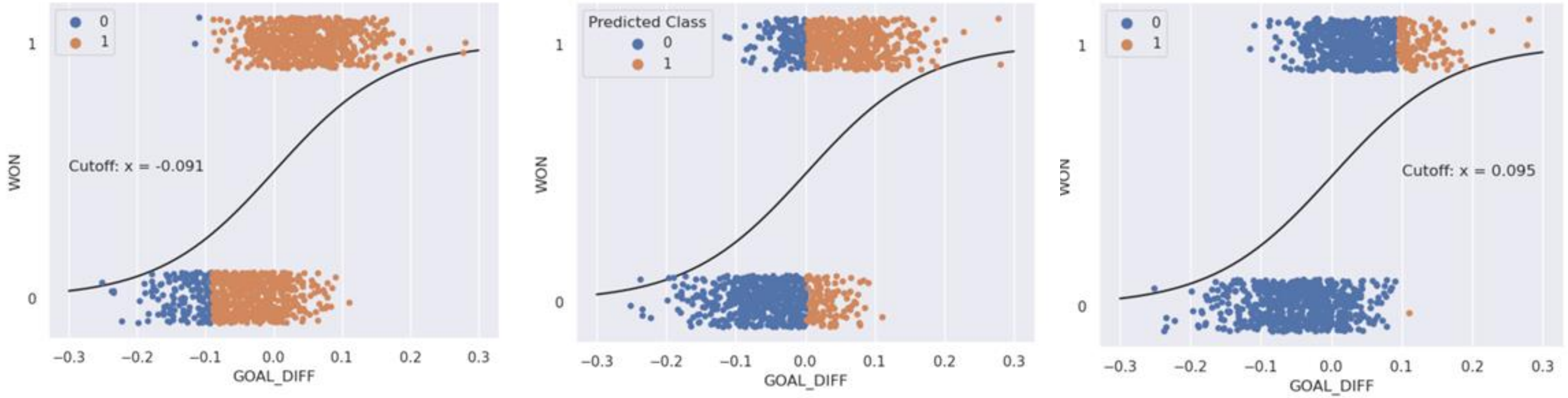
Fewer positives

Changing the Threshold

$$\hat{y} = \text{classify}(x) = \begin{cases} \text{Class } 1 & p \geq T \\ \text{Class } 0 & p < T \end{cases}$$

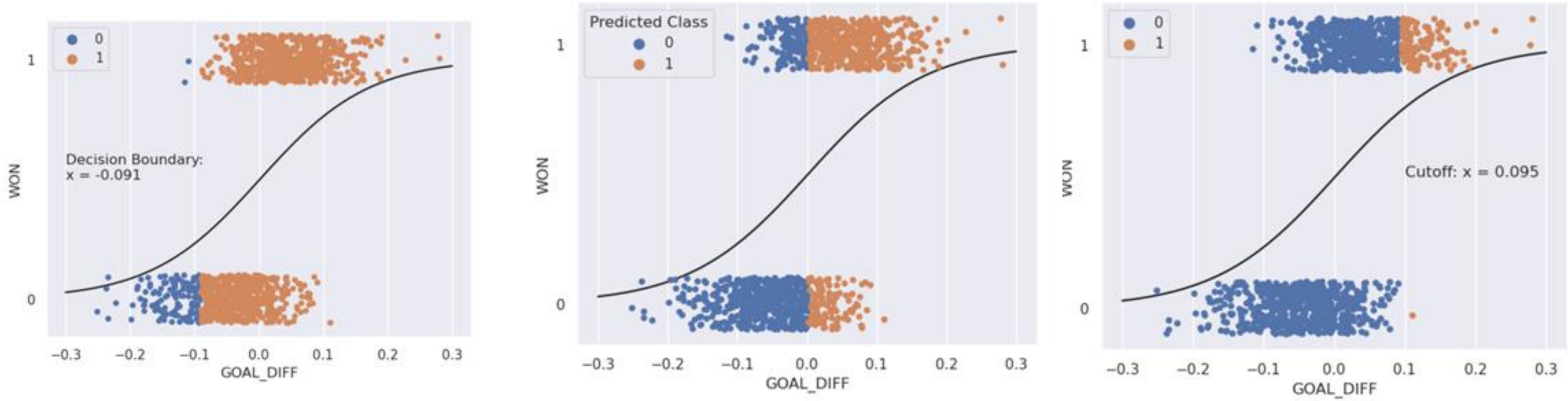
As we increase the threshold T, we "raise the standard" of how confident our classifier needs to be to predict 1 (i.e., "positive").

T = 0.25 T = 0.50 T = 0.75



Changing the Threshold

T = 0.25 T = 0.50 T = 0.75



How to interpret a higher classification threshold: the model needs to predict a higher probability of a point belonging to Class 1 before it can confidently classify it as Class 1

- As T increases, we predict fewer positives!

Changing the threshold allows us to finetune how “confident” we want our model to be before making a positive prediction

Precision-Recall Curves

Earlier, we noticed that there is a tradeoff between precision (penalizes false positives) and recall (penalizes false negatives).

We should choose a threshold that keeps both precision and recall high.

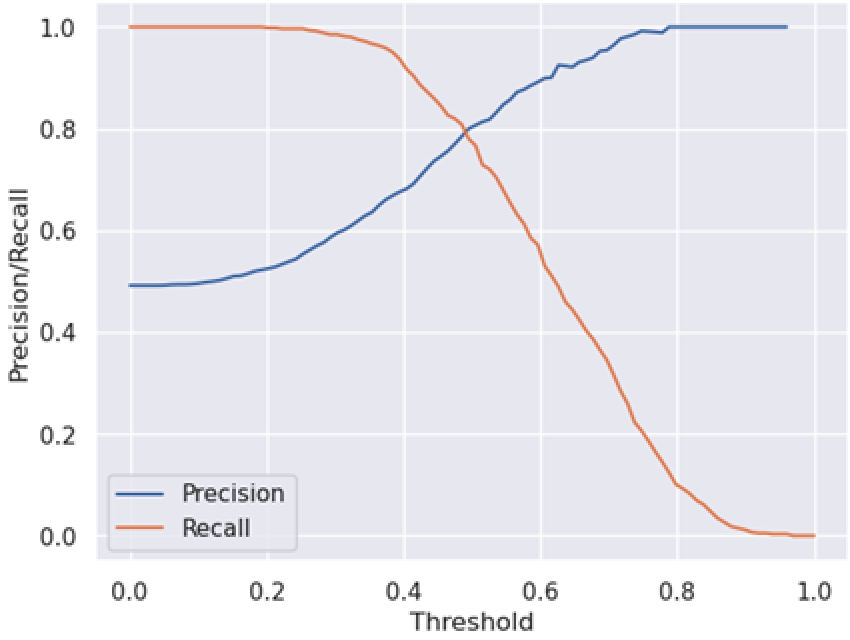
In a precision-recall curve, we:

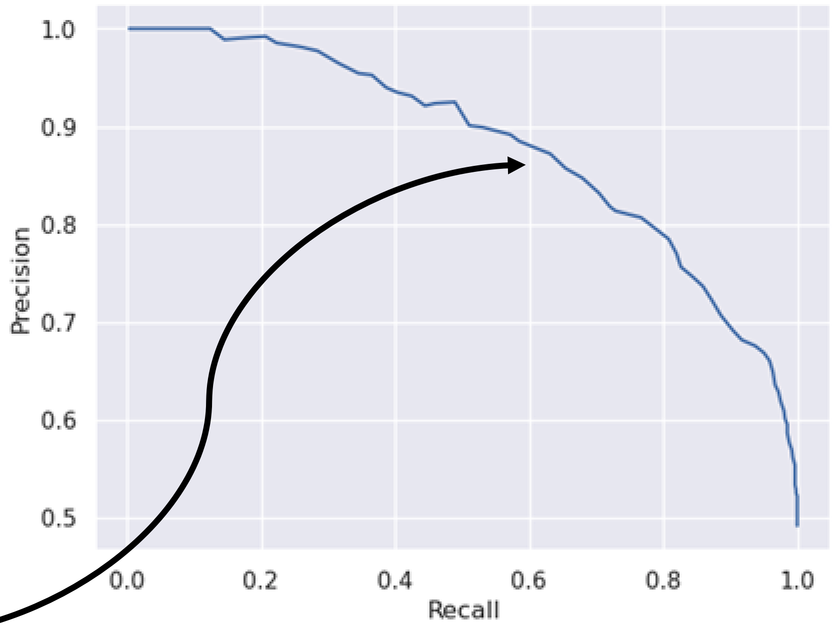
1) Test out many different possible thresholds
2) For each threshold, compute the precision and recall of the classifier
3) Choose the threshold at the “corner” of the curve

Precision-Recall Curves

In a precision-recall curve, we:

1) Test out many different possible thresholds
2) For each threshold, compute the precision and recall of the classifier





We should choose a threshold that keeps both precision and recall high.

F measure

- Combines precision and recall using harmonic mean, the traditional F-measure

$$\mathrm{F} = 2.\frac{precision.recall}{precision + recall}$$