LECTURE 19

# Logistic Regression, Classification

Using our model to make classifications, and evaluating the quality of our model.

# Recap: Logistic regression

- In a **logistic regression** model, we 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.

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

- We arrived at this model by assuming that the **log-odds of the probability of belonging to class 1 is linear.**
- To find $\hat{\theta}$, we can choose squared loss or cross-entropy loss.
  - Squared loss works, but is generally not a good idea.
  - Cross-entropy loss is much better (convex, better suited for modeling probabilities).

# Review: Logistic Regression Process

In logistic regression, we model the *probability* that a datapoint belongs to Class 1.

| 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 **sigmoid**

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

**Decision rule**

Output: **class**

Last lecture

Today

# Thresholding

**Thresholding**

Evaluating classifiers

Visual Metrics

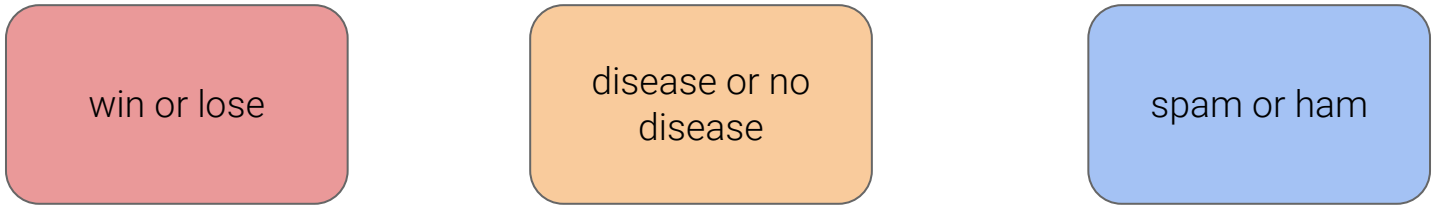
Decision boundaries

Linear separability, regularization

Multiclass 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.




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

In order to **classify** – that is, to predict a 1 or 0 – we pair our logistic model with a **decision rule**, or **threshold**.

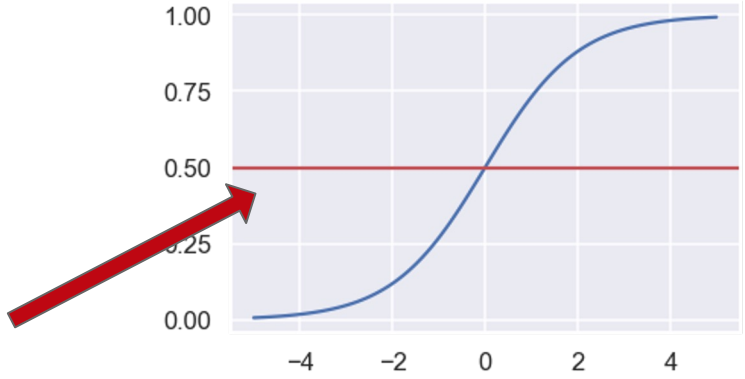
A **decision rule** tells us how to interpret the output of the model to make a decision on how to classify a datapoint.

# Thresholds

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

Example for $T = 0.5$:

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



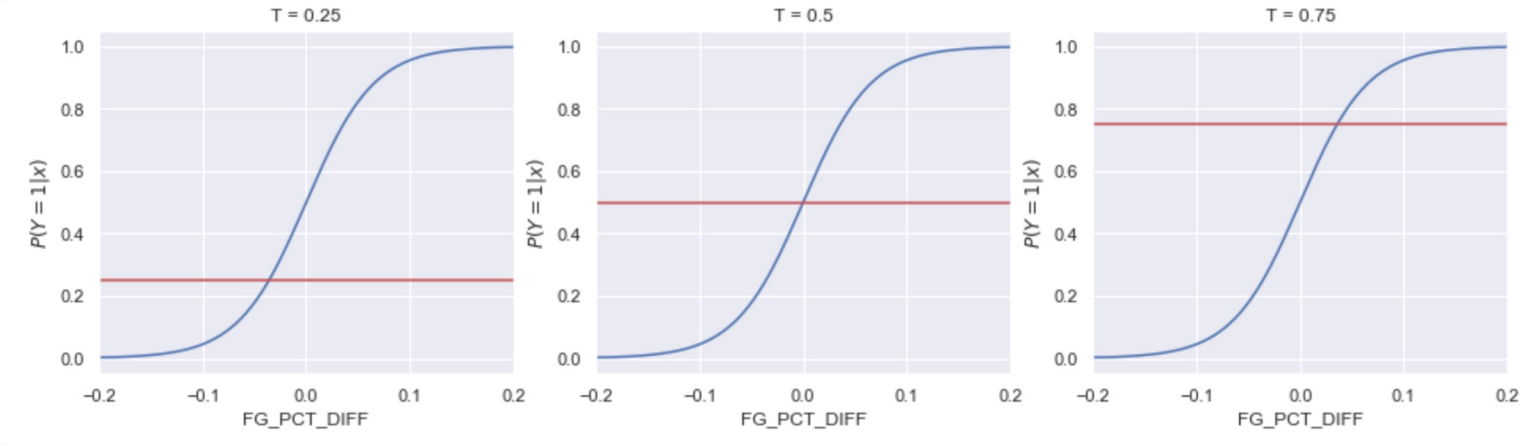

- Note: We **don't need** to set our **threshold** to 0.5. Depending on the type of errors we want to minimize, we can increase or decrease it.
  - 0.5 is the default in `scikit-learn`'s `LogisticRegression`, though.
- Logistic regression paired with a decision rule is a classifier.

# Thresholds

Consider the single-feature logistic regression model from last lecture:

$$P(Y = 1|x) = \sigma(\theta_1 \cdot \text{FG\_PCT\_DIFF})$$

Here, the blue line represents our modeled probabilities, and the red lines represent various thresholds.

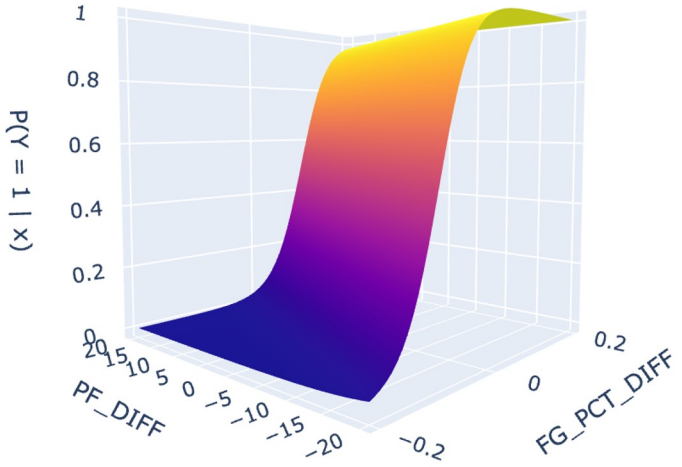


# Thresholds in higher dimensions

Our thresholds work the same way, even if our models have multiple features.

Suppose we fit a model with 2 features – FG_PCT_DIFF and PF_DIFF – along with an intercept term.

$$P(Y = 1|x) = \sigma(\theta_0 + \theta_1 \cdot \mathbf{FG\_PCT\_DIFF} + \theta_2 \cdot \mathbf{PF\_DIFF})$$



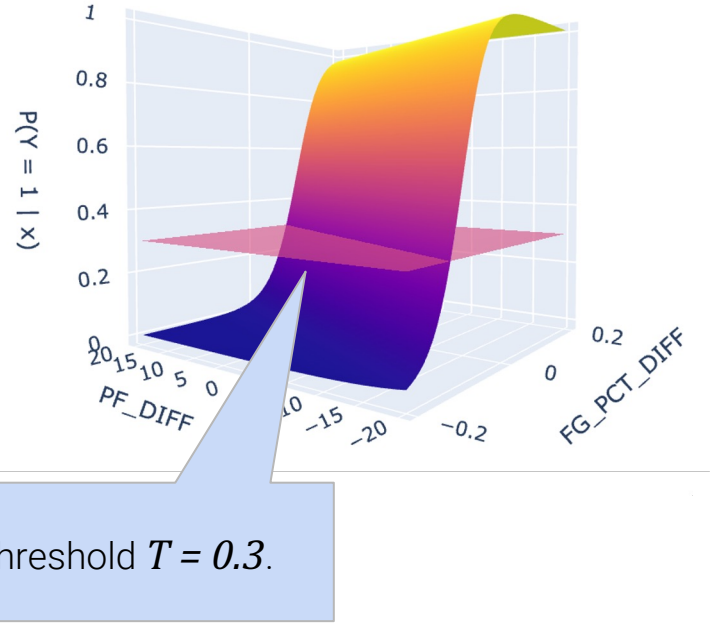
# Thresholds in higher dimensions

Our thresholds work the same way, even if our models have multiple features.

Suppose we fit a model with 2 features – FG_PCT_DIFF and PF_DIFF – along with an intercept term.

$$P(Y = 1|x) = \sigma(\theta_0 + \theta_1 \cdot \text{FG\_PCT\_DIFF} + \theta_2 \cdot \text{PF\_DIFF})$$

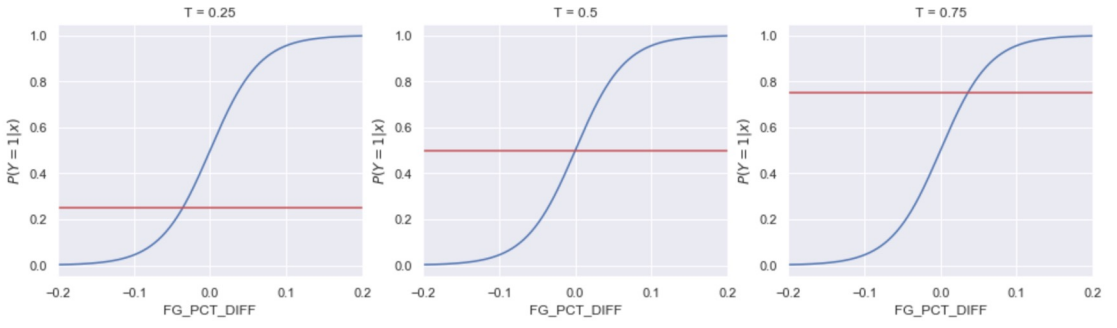
Any data point whose predicted probability is greater than 0.3 (above the plane) is classified as 1.



# From probabilities to labels

With different thresholds, we get different predictions.

- Everything above the red line is classified as 1, and everything below is classified as 0.
- The larger we make $T$, our threshold, the fewer observations are classified as 1.
  - The “standard” is higher.




| P(Y = 1 \| x) | T = 0.25 | T = 0.5 | T = 0.75 |
|---|---|---|---|
| 0.182665 | 0 | 0 | 0 |
| 0.834894 | 1 | 1 | 1 |
| 0.285491 | 1 | 0 | 0 |
| 0.777950 | 1 | 1 | 1 |
| 0.783187 | 1 | 1 | 1 |
| ... | ... | ... | ... |
| 0.996919 | 1 | 1 | 1 |
| 0.891870 | 1 | 1 | 1 |
| 0.627113 | 1 | 1 | 0 |
| 0.059965 | 0 | 0 | 0 |
| 0.048976 | 0 | 0 | 0 |

# Evaluating Classifiers

Thresholding

**Evaluating classifiers**

Visual Metrics

Decision boundaries

Linear separability, regularization

Multiclass classification

# 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

Let's do it

# 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.

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

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

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

accuracy(X, Y) # 0.794
```

```
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.

- 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**.

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!!!

Your other friend ("Friend 2"):

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

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

**Low** ⚠ accuracy…
…but we detected **all** of the spam!!!

# Pitfalls of Accuracy: Class Imbalance

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).

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!!!

Your other friend:

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

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

**Low** ⚠ accuracy…
…but we detected **all** of the spam!!!

# Types of classification errors

There are four different classifications that our model might make:

Moving forward, “positive” refers to 1 and “negative” refers to 0.

- **True positives and true negatives** are when we correctly classify an observation as being positive or negative.
- **False positives** are like “false alarms.” **(Type I Error)**
- **False negatives** are when we “fail to detect” things. **(Type II Error)**

| | | Prediction | |
|---|---|---|---|
| | | 0 | 1 |
| Actual | 0 | True negative (TN) | False positive (FP) |
| | 1 | False negative (FN) | True positive (TP) |

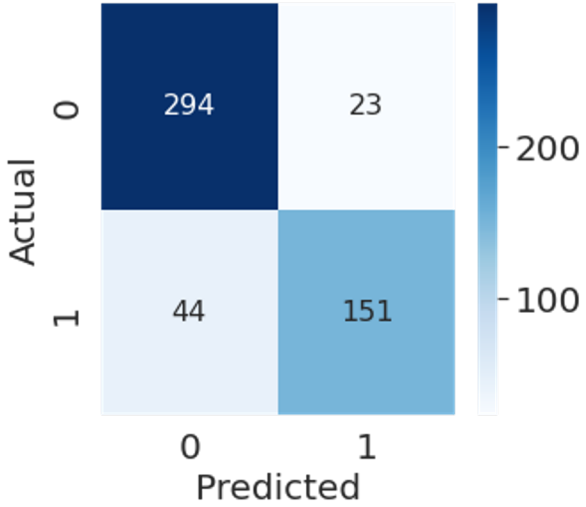
Sometimes this table is presented with predictions on the y-axis and actual values on the x-axis.

# 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

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

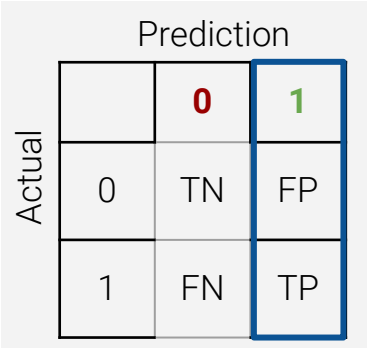
What proportion of points did our classifier classify correctly?

| Actual \ Prediction | **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.

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

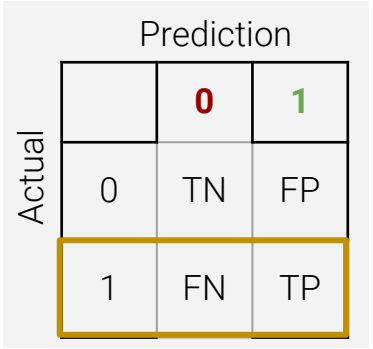
Of all observations that were predicted to be 1, what proportion were actually 1?

- 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?




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.

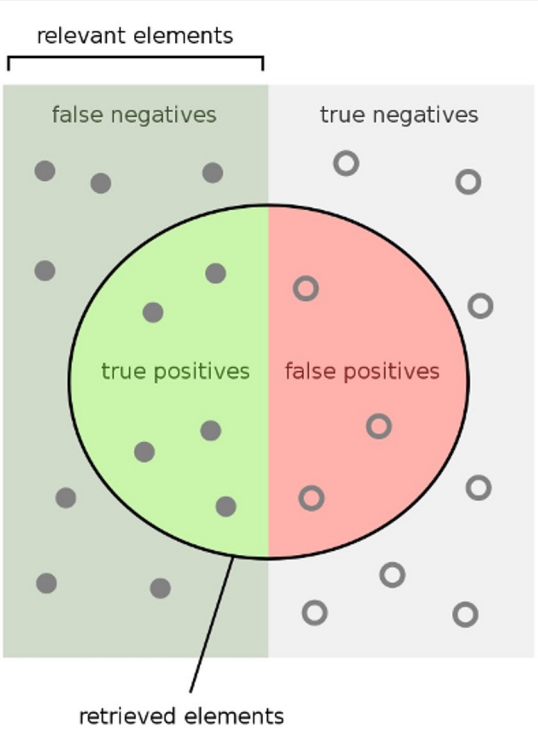
$$\text{recall} = \frac{TP}{TP + FN}$$

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., positive;
predicted class is 1)




How many retrieved items are relevant?

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

How many relevant items are retrieved?

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

(*i.e., true 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).

| | 0 | 1 |
|---|---|---|
| 0 | TN: 95 | FP: 0 |
| 1 | FN: 5 | TP: 0 |

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

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

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

# 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).

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

Never positive!

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

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

Your other friend ("Friend 2"):

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

| | 0 | 1 |
|---|---|---|
| 0 | TN: 0 | FP: 95 |
| 1 | FN: 0 | TP: 5 |

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

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

Many false positives!

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

No false negatives!

# Precision vs. Recall

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

Precision penalizes false positives, and

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

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.

# Examples

In each of the following cases, what would we want to maximize: precision, recall, or accuracy?

- Predicting whether or not a patient has some disease.
- Determining whether or not someone should be sentenced to life in prison.
- Determining if an email is spam or ham.

# Examples

In each of the following cases, what would we want to maximize: precision, recall, or accuracy?

- Predicting whether or not a patient has some disease.
  - Maximize **recall**.
  - Presumably if we say someone has the disease, they will go through further testing.
  - If we say they don't, the condition may be left untreated, which is dangerous.
- Determining whether or not someone should be sentenced to life in prison.
  - Maximize **precision**.
  - We don't want to sentence innocent people, so we want to be very sure that this is a true positive.
- Determining if an email is spam or ham.
  - Maximize **accuracy**, though this one is subjective.
  - Depends what you think is worse – having a bunch of spam emails ending up in your inbox, or a bunch of non-spam emails being filtered out.

# True and False Positive Rates

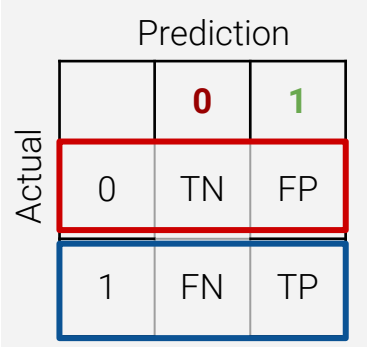
Keeping things interesting – two more performance metrics.

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

**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.




**F1 Score**(F1):

$$\frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

# Visual Metrics

Thresholding

Evaluating classifiers

**Visual Metrics**

Decision boundaries

Linear separability, regularization

Multiclass classification

# The effect of thresholds

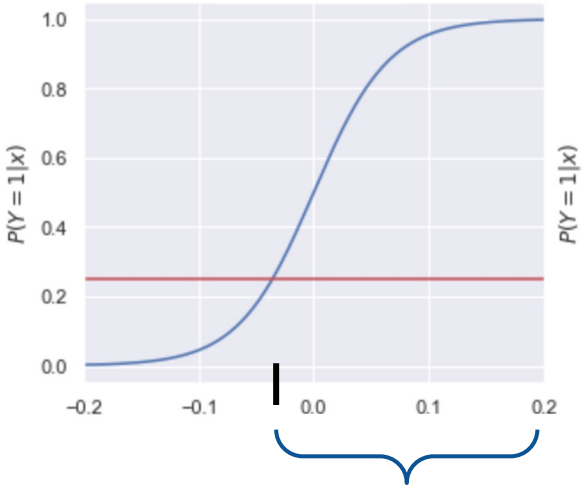
- Our choice of threshold impacts our model's:
  - Accuracy.
  - Precision.
  - Recall.
- Let's explore!

# Changing the Threshold

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

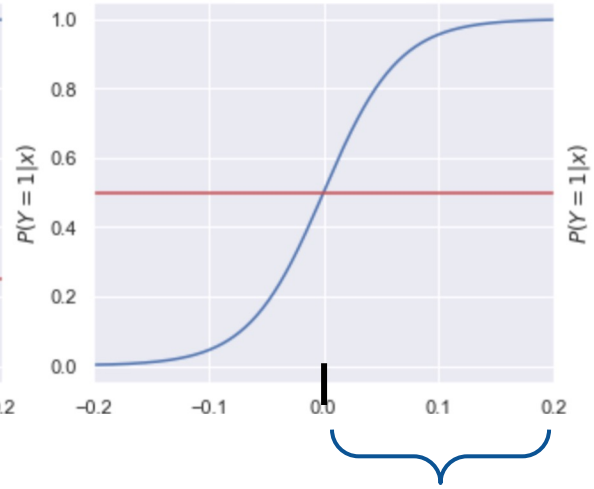
As we increase the threshold T, we “raise the standard” of how confident our classifier needs to be to predict 1 (i.e., “positive”).

T = 0.25

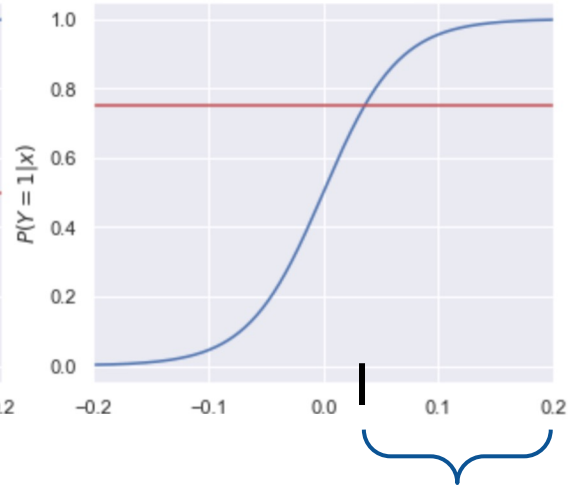



These x will all predict 1

T = 0.50



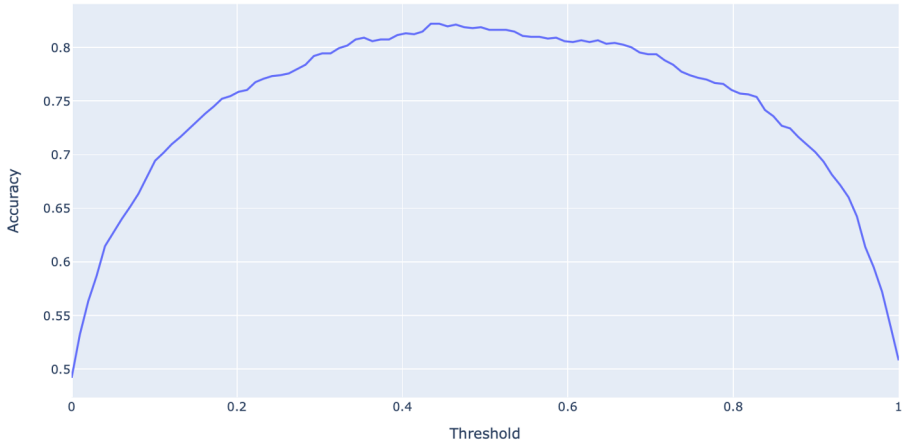

T = 0.75




Fewer positives

# Accuracy vs. threshold

- If our threshold is too high, we will have many false negatives.
- If our threshold is too low, we will have many false positives.
- Thus, we'd expect our accuracy to be maximized when our threshold is near 0.5 in a typical setting.
    - Not always the case.




Accuracy vs. threshold for our two feature NBA model, $P(Y = 1|x) = \sigma(\theta_0 + \theta_1 \cdot \text{FG\_PCT\_DIFF} + \theta_2 \cdot \text{PF\_DIFF})$

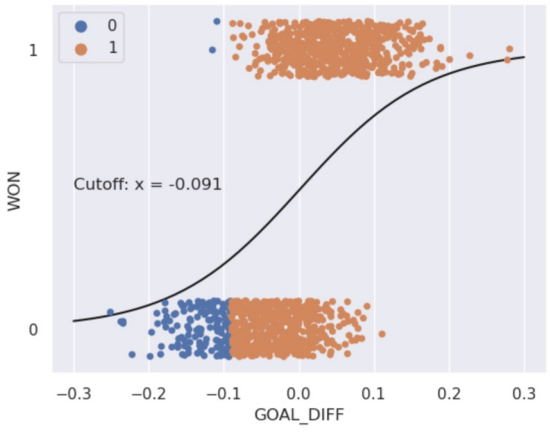
.

# Changing the Threshold

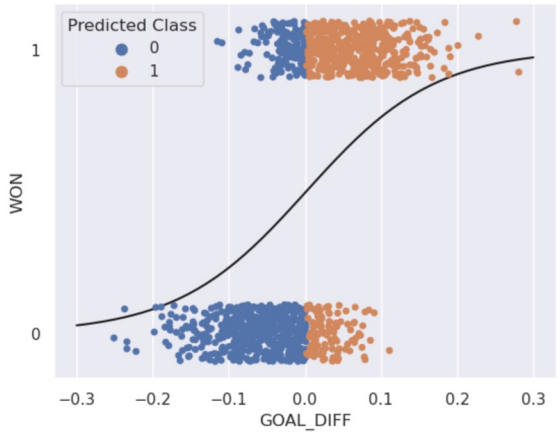
$$\hat{y} = \text{classify}(x) = \begin{cases} \text{Class 1} & p \geq T \\ \text{Class 0} & p < T \end{cases}$$

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

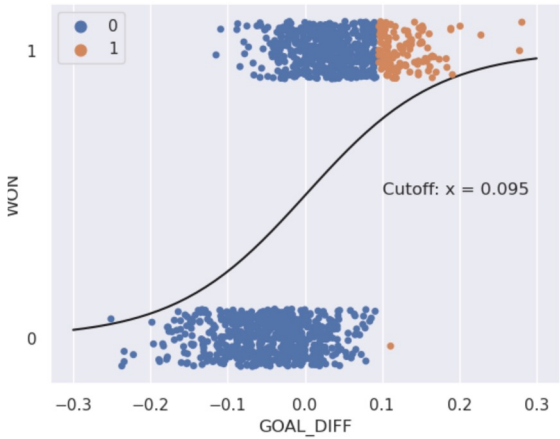
T = 0.25




T = 0.50

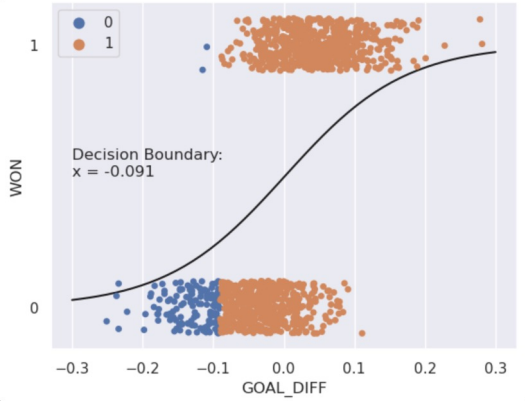



T = 0.75

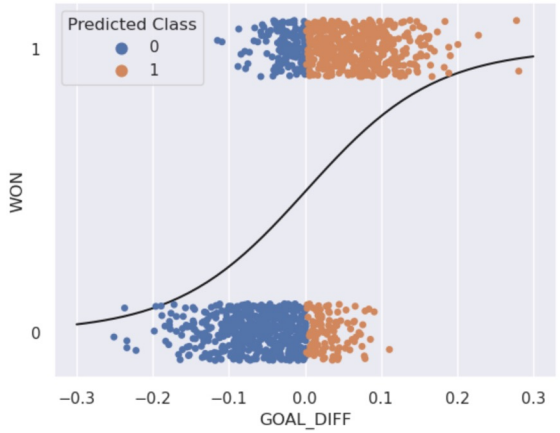


# Changing the Threshold

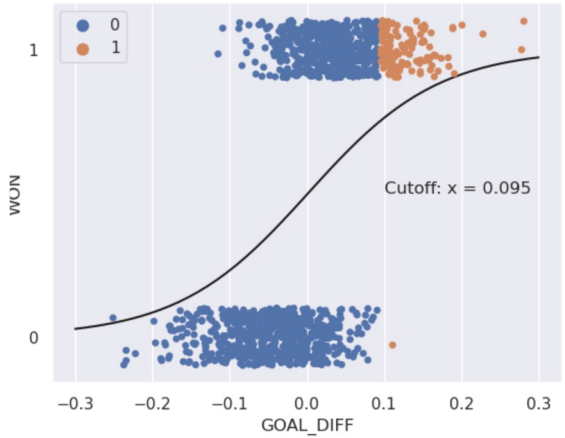
T = 0.25




T = 0.50




T = 0.75



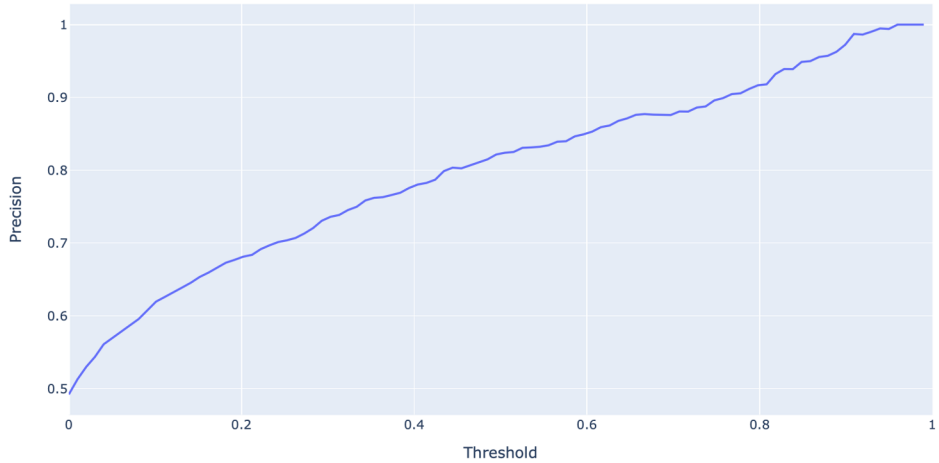

How to interpret a higher classification threshold: the model needs to predict a higher probability of a point belonging to Class 1 before it can confidently classify it as Class 1

- 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 vs. threshold

- As we increase our threshold, we have fewer and fewer false positives.
- Thus, precision tends to increase.




$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} = \frac{\text{True Positives}}{\text{Predicted True}}$$

It is *possible* for precision to decrease slightly with an increased threshold. Why?

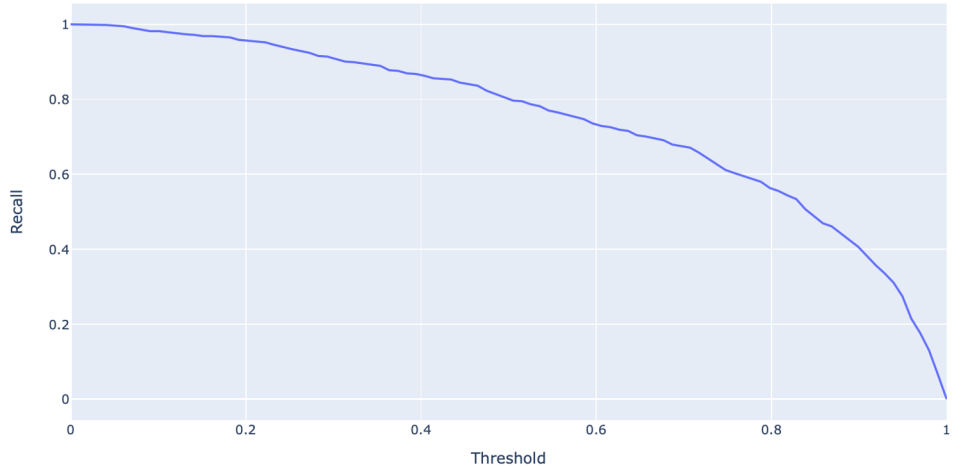
# Recall vs. threshold

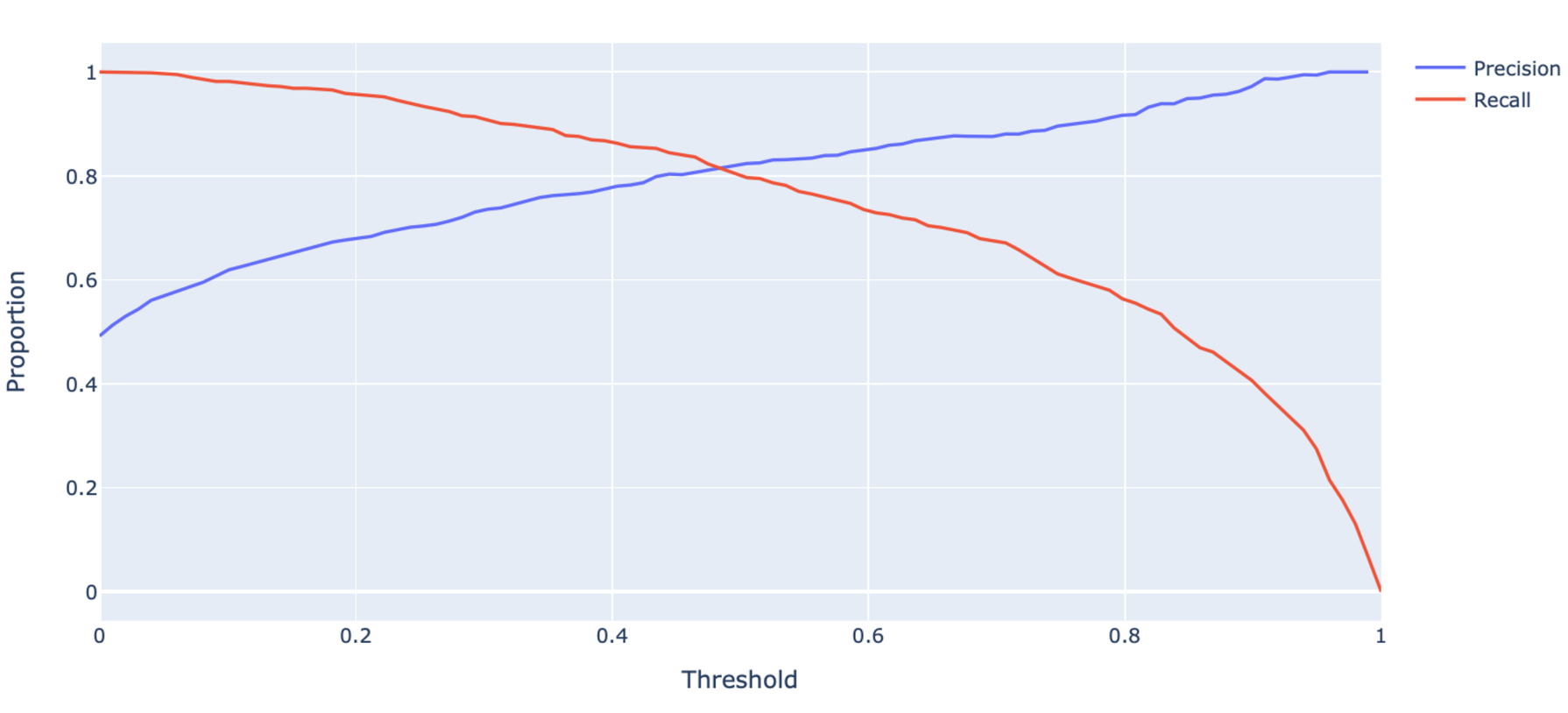
- As we increase our threshold, we have more and more false negatives.
- Thus, recall tends to decrease.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$= \frac{\text{True Positives}}{\text{Actually True}}$$

Recall strictly decreases as we increase our threshold. Why?



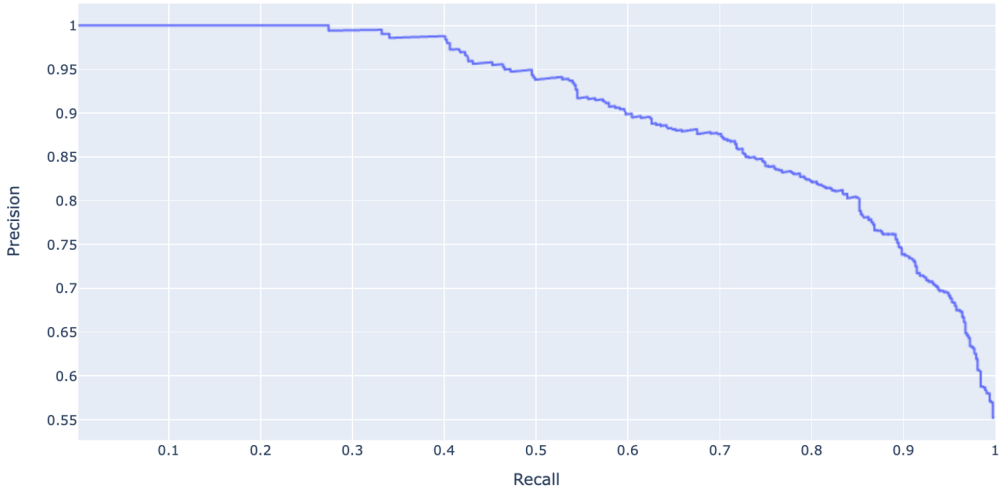


# Precision-recall curves

We can also plot precision vs. recall, for all possible thresholds.

**Question**:

- Which part of this curve corresponds to T = 0.9?
- Which part of this curve corresponds to T = 0.1?

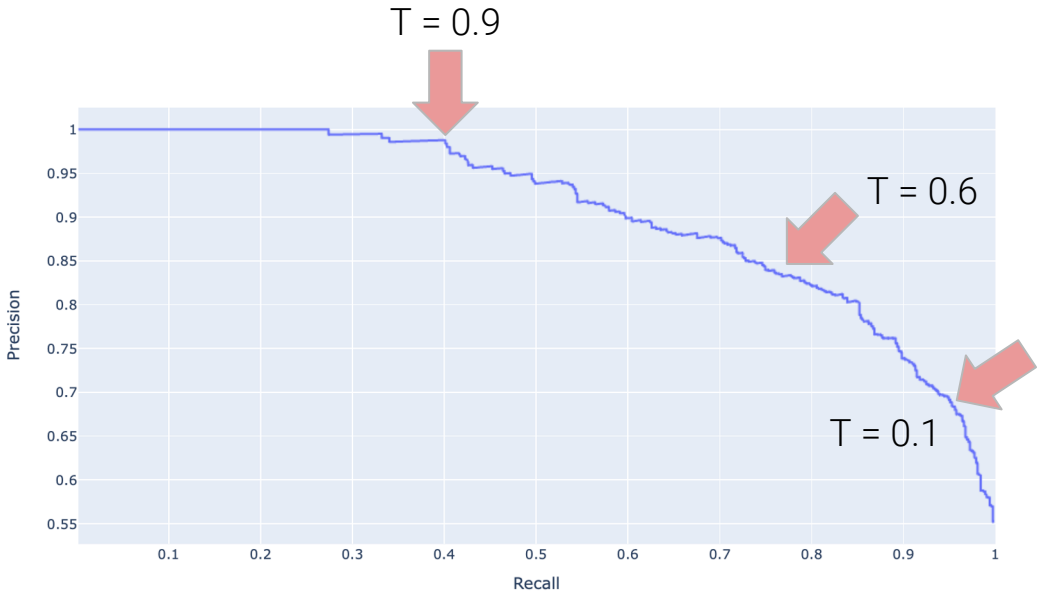


## Precision-recall curves

We can also plot precision vs. recall, for all possible thresholds.

Answer:

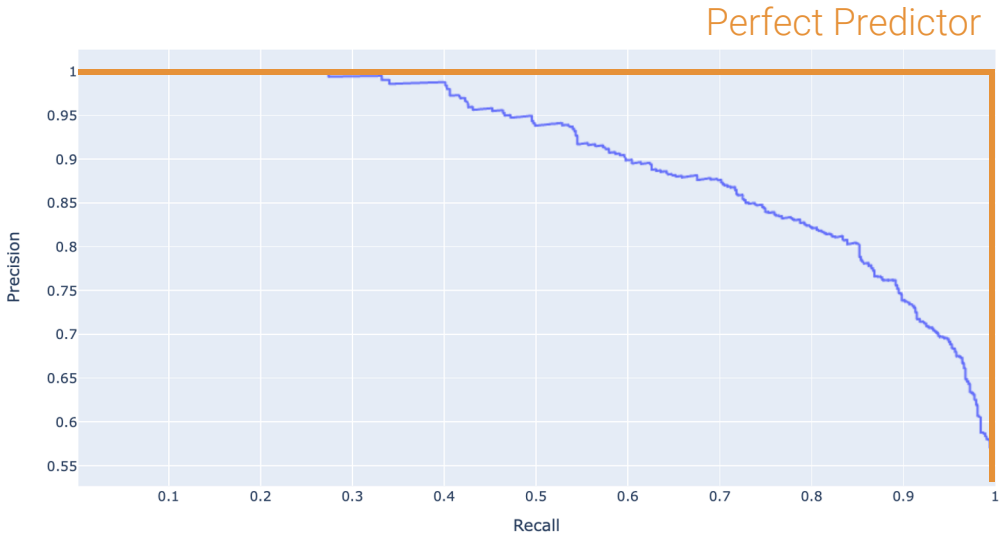
- Threshold decreases from the top left to the bottom right.
- In the notebook, there's an interactive version of this plot.



# Precision-recall curves

The “perfect classifier” is one with precision of 1 and recall of 1.

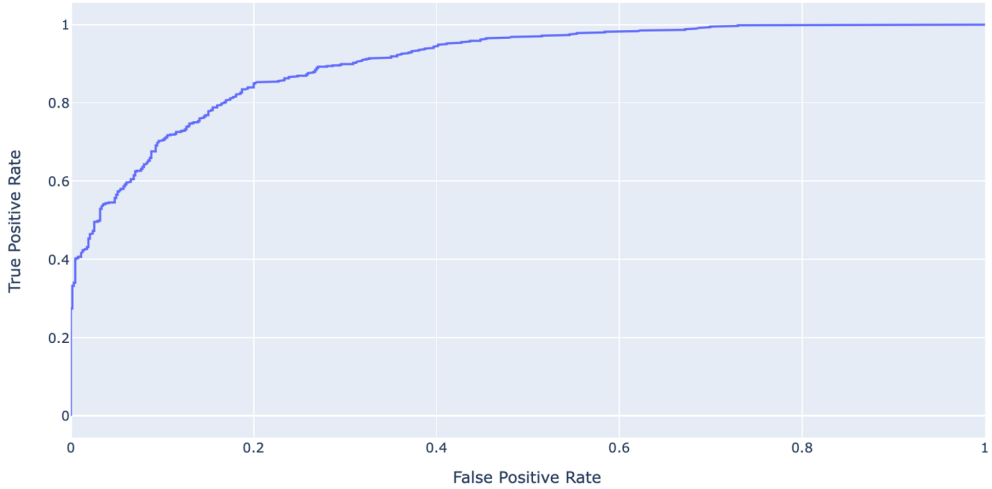
- We want our PR curve to be as close to the “top right” of this graph as possible.
- One way to compare our model is to compute its **area under curve (AUC)**.
  - The area under the “optimal PR curve” is 1.
  - More commonly, we look at the area under ROC curve.



# ROC curves

A ROC curve plots TPR vs. FPR.

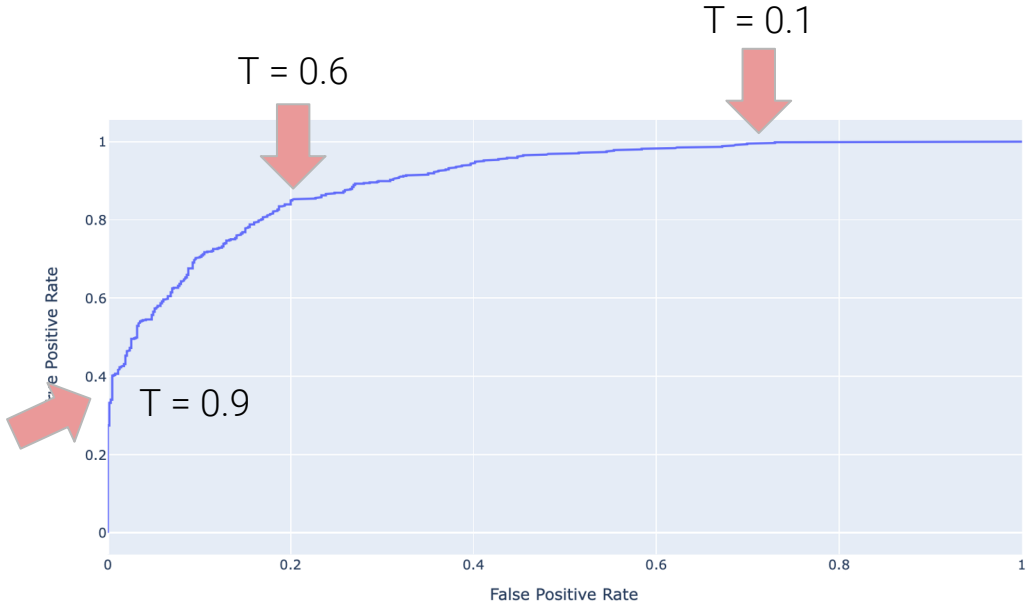
- As we increase our threshold, both TPR and FPR decrease.
  - A decreased TPR is bad (detecting fewer positives).
  - A decreased FPR is good (fewer false positives).
  - Tradeoff!
- ROC stands for “Receiver Operating Characteristic.”



# ROC curves

A ROC curve plots TPR vs. FPR.

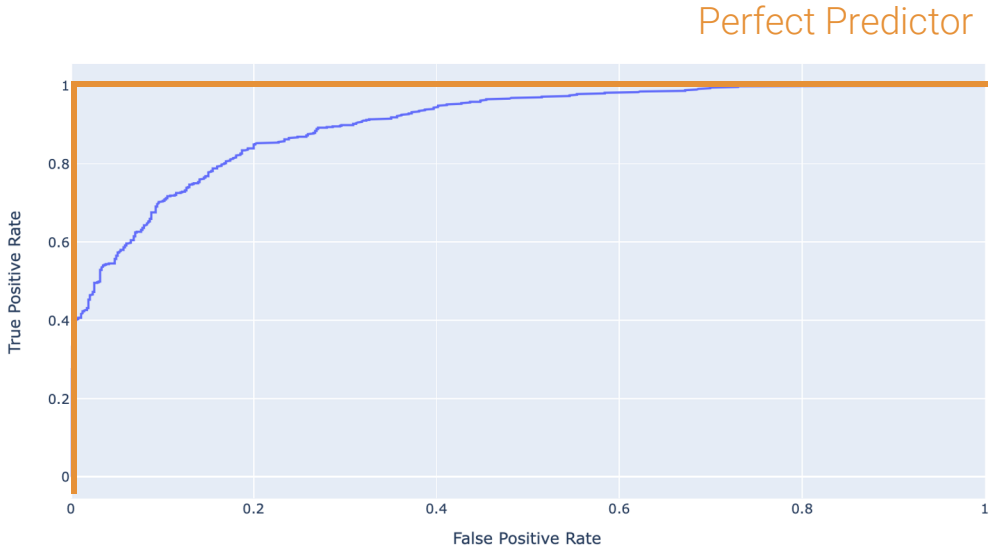
- As we increase our threshold, both TPR and FPR decrease.
  - A decreased TPR is bad (detecting fewer positives).
  - A decreased FPR is good (fewer false positives).
  - Tradeoff!
- ROC stands for “Receiver Operating Characteristic.”



# ROC curves and AUC

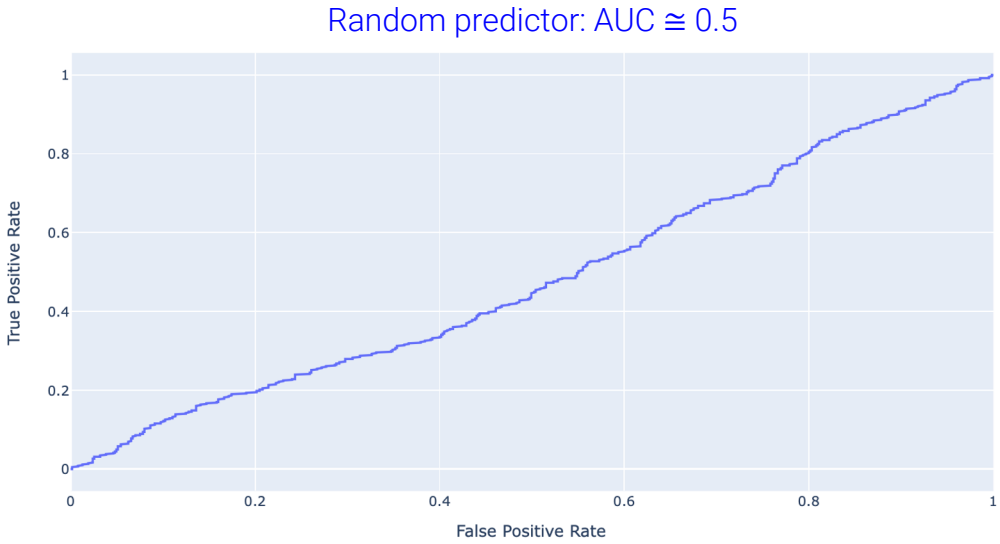
The “perfect” classifier is the one that has a TPR of 1, and FPR of 0.

- We want our logistic regression model to match that as well as possible.
- We want our ROC curve to be as close to the “top left” of this graph as possible.
- We can compute the **area under curve (AUC)** of our model.
  - Best possible AUC = 1.
  - Terrible AUC = 0.5 (randomly guessing).



# ROC curves and AUC

- We can compute the **area under curve (AUC)** of our model.
  - Different AUCs for both ROC curves and PR curves, but ROC is more common.
- Best possible AUC = 1.
- Terrible AUC = 0.5.
  - Random predictors have an AUC of around 0.5. Why?
- Your model's AUC: somewhere between 0.5 and 1.



# Common techniques for evaluating classifiers

Numerical assessments:

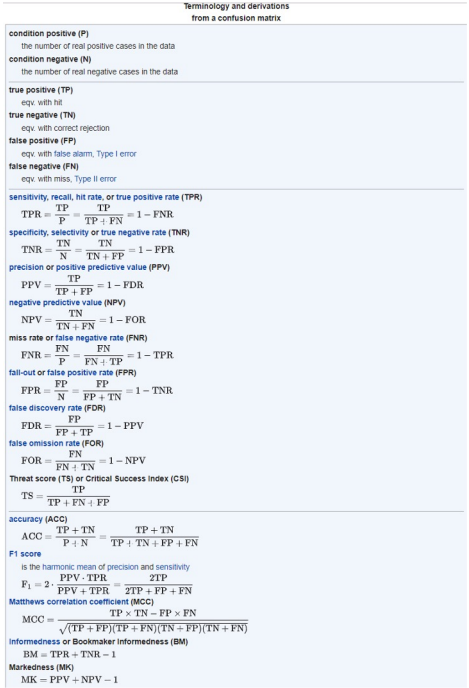
- **Accuracy, precision, recall, F1, TPR, FPR.**
- Area under curve (AUC), for both PR and ROC.

Visualizations:

- **Confusion matrices.**
- Precision/recall curves.
- ROC curves.

The **bolded** metrics depend only on our predictions.

The non-bolded metrics depend on our underlying regression model.



Terminology and derivations from a confusion matrix

**condition positive (P)**
the number of real positive cases in the data
**condition negative (N)**
the number of real negative cases in the data

**true positive (TP)**
eqv. with hit
**true negative (TN)**
eqv. with correct rejection
**false positive (FP)**
eqv. with false alarm, Type I error
**false negative (FN)**
eqv. with miss, Type II error

**sensitivity, recall, hit rate, or true positive rate (TPR)**
$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = 1 - \mathrm{FNR}$$
**specificity, selectivity or true negative rate (TNR)**
$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} = 1 - \mathrm{FPR}$$
**precision or positive predictive value (PPV)**
$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = 1 - \mathrm{FDR}$$
**negative predictive value (NPV)**
$$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}} = 1 - \mathrm{FOR}$$
**miss rate or false negative rate (FNR)**
$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{P}} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} = 1 - \mathrm{TPR}$$
**fall-out or false positive rate (FPR)**
$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} = 1 - \mathrm{TNR}$$
**false discovery rate (FDR)**
$$\mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TP}} = 1 - \mathrm{PPV}$$
**false omission rate (FOR)**
$$\mathrm{FOR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TN}} = 1 - \mathrm{NPV}$$
**Threat score (TS) or Critical Success Index (CSI)**
$$\mathrm{TS} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP}}$$

**accuracy (ACC)**
$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$
**F1 score**
is the harmonic mean of precision and sensitivity
$$\mathrm{F}_1 = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$
**Matthews correlation coefficient (MCC)**
$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TN} + \mathrm{FN})}}$$
**Informedness or Bookmaker Informedness (BM)**
$$\mathrm{BM} = \mathrm{TPR} + \mathrm{TNR} - 1$$
**Markedness (MK)**
$$\mathrm{MK} = \mathrm{PPV} + \mathrm{NPV} - 1$$

We're only scratching the surface here.

# Decision Boundaries

Thresholding

Evaluating Classifiers

Visual Metrics

**Decision Boundaries**

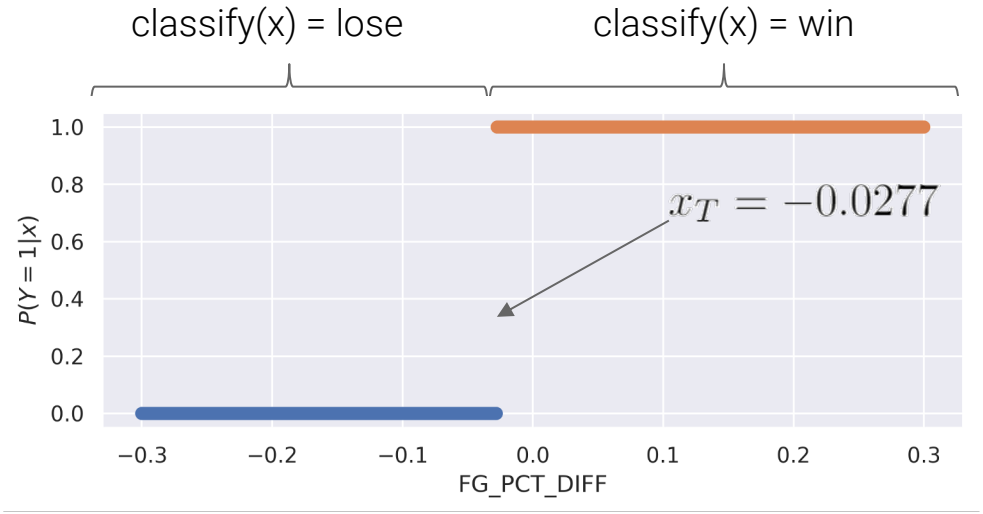
Linear separability, Regularization

Multiclass classification

# Decision boundaries

If we pick a threshold, e.g. T = 0.3, we get a predicted class from probabilities.

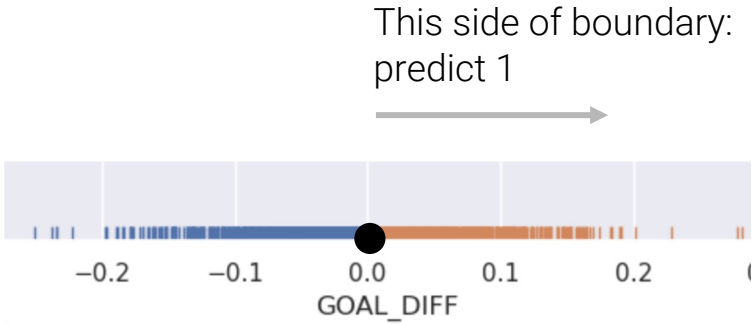
- The effect is that x < $x_T$ predicts class 0, and x > $x_T$ predicts class 1.
- $x_T$ is known as a **decision boundary**.
  - $x_T$ is a function of model parameters and T.



# 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

# Decision Boundaries

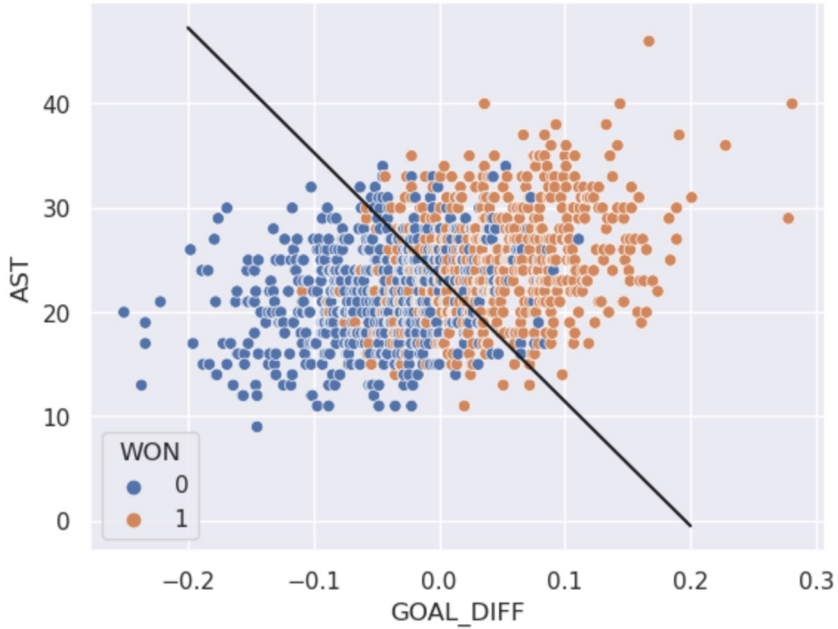
We can recover the decision boundary from the final logistic regression model.

Example: model with two features

$$T = \frac{1}{1 + e^{-(\theta_0 + \theta_1 \text{GOAL\_DIFF} + \theta_2 \text{AST})}}$$

$$\theta_0 + \theta_1 \text{GOAL\_DIFF} + \theta_2 \text{AST} = -\log\left(\frac{1}{T} - 1\right)$$

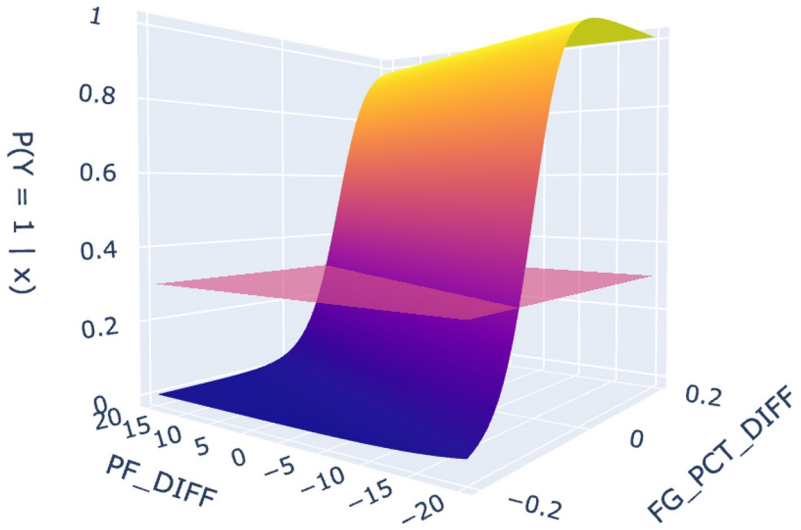
The decision boundary is a line in terms of the features.




Here, the color is the *true* class, rather than the model's predictions.

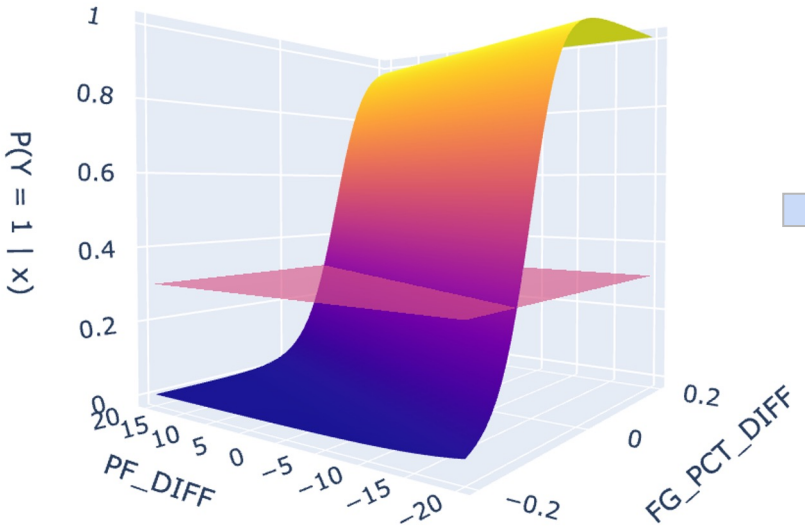
# Decision boundaries for 2D models

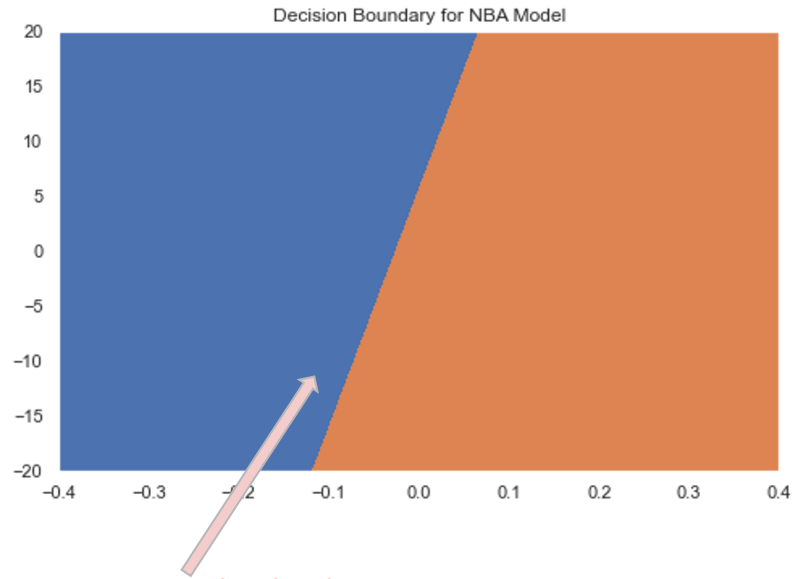
Now consider our "better" model, $P(Y = 1|x) = \sigma(\theta_0 + \theta_1 \cdot \text{FG\_PCT\_DIFF} + \theta_2 \cdot \text{PF\_DIFF})$ . It is drawn below with the thresholding line T = 0.3. **What does its decision boundary look like?**



# Decision boundaries for 2D models

Now consider our "better" model, $P(Y = 1|x) = \sigma(\theta_0 + \theta_1 \cdot \text{FG\_PCT\_DIFF} + \theta_2 \cdot \text{PF\_DIFF})$ . It is drawn below with the thresholding line T = 0.3. **The decision boundary is linear.**




Line depends on $\theta_0$, $\theta_1$, $\theta_2$, T.

# Decision boundaries for 2D models

Suppose we minimized mean cross-entropy loss to determine the optimal model parameters for this model, and found

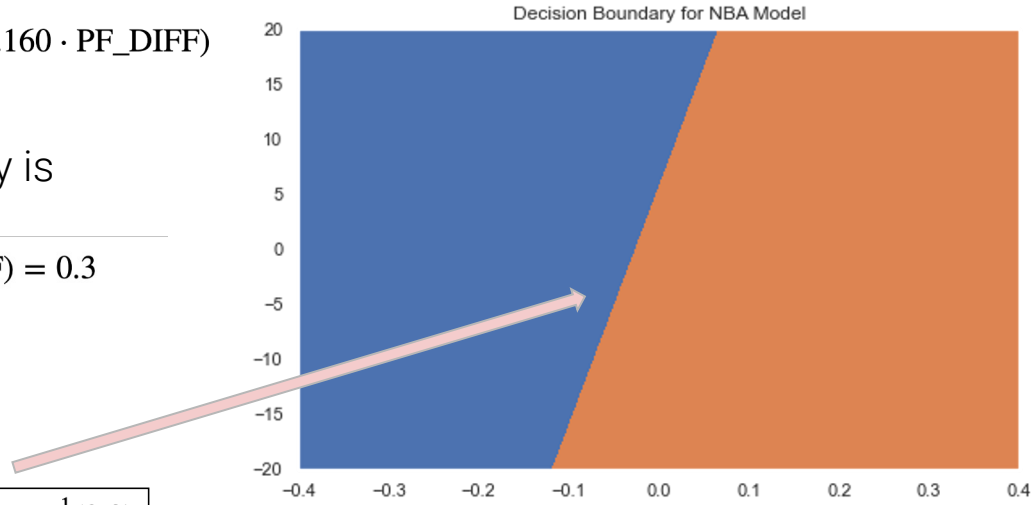
$$P(Y = 1|x) = \sigma(0.035 + 34.705 \cdot \text{FG\_PCT\_DIFF} - 0.160 \cdot \text{PF\_DIFF})$$

If we set T = 0.3, our decision boundary is

$$\sigma(0.035 + 34.705 \cdot \text{FG\_PCT\_DIFF} - 0.160 \cdot \text{PF\_DIFF}) = 0.3$$

Which simplifies to

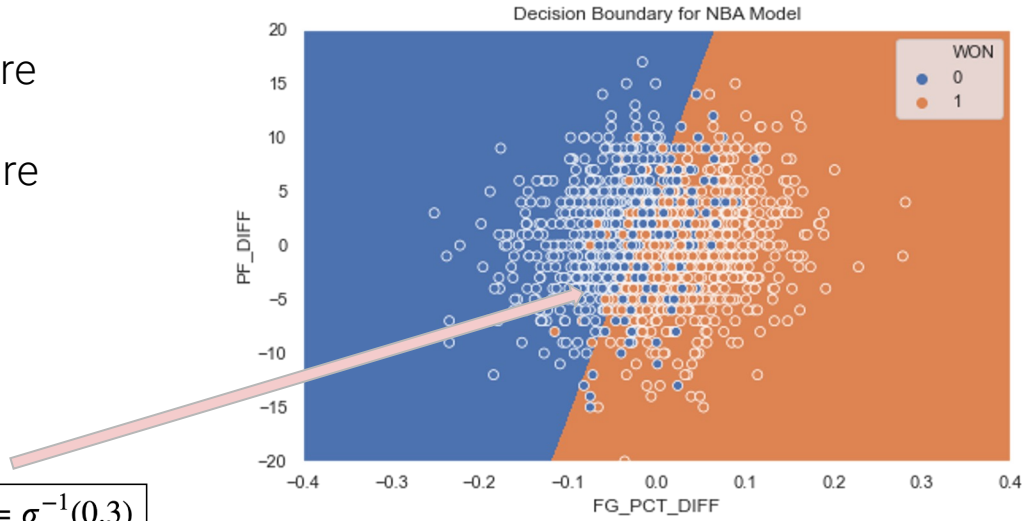
$$0.035 + 34.705 \cdot \text{FG\_PCT\_DIFF} - 0.160 \cdot \text{PF\_DIFF} = \sigma^{-1}(0.3)$$



# Decision boundaries for 2D models

If we overlay our true observations onto our decision boundary, we can get a rough sense of the accuracy of our model and the types of errors it makes.

- Blue points in the orange region are false positives.
- Orange points in the blue region are false negatives.




$$0.035 + 34.705 \cdot \text{FG\_PCT\_DIFF} - 0.160 \cdot \text{PF\_DIFF} = \sigma^{-1}(0.3)$$

# Linear separability, Regularization

Thresholding

Evaluating Classifiers

Visual Metrics

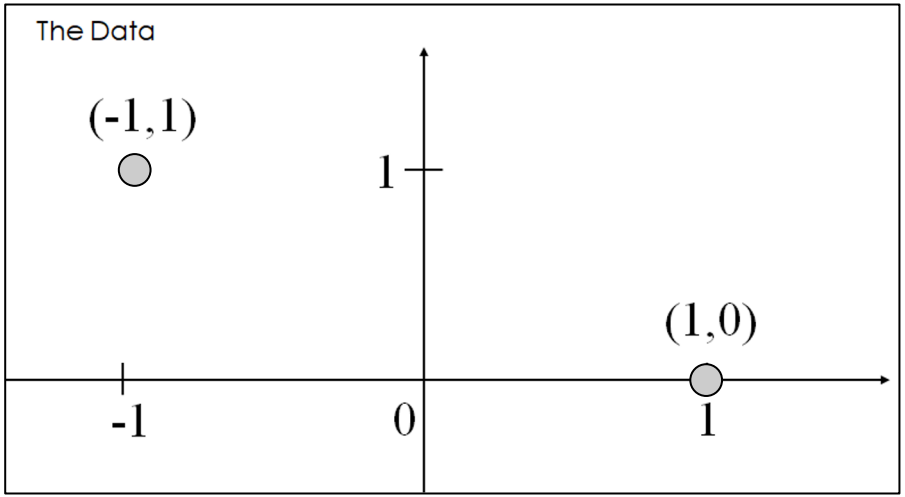
Decision Boundaries

**Linear separability, Regularization**

Multiclass classification

# Question

Suppose we're training a single-parameter logistic regression model on the data to the right.




What $\theta$ minimizes mean cross-entropy loss?

$\hat{\theta} = -1$

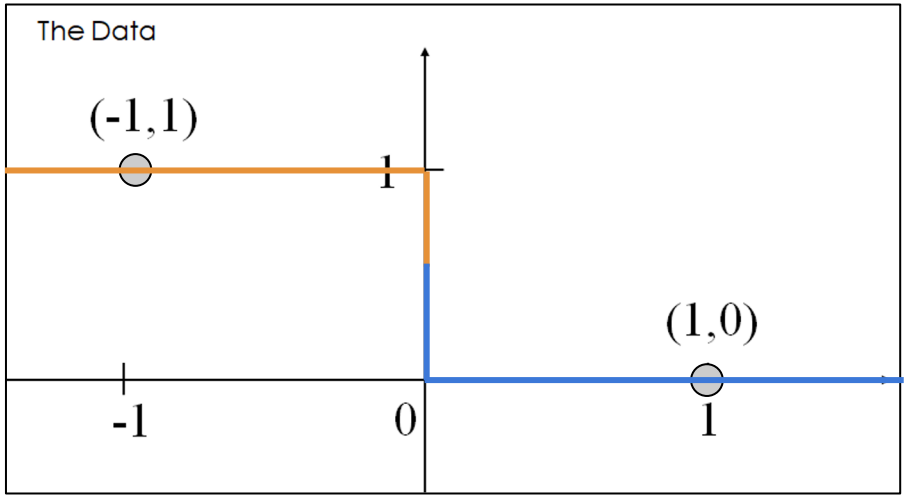
$\hat{\theta} = 1$

$\hat{\theta} \to -\infty$

$\hat{\theta} \to \infty$

# Question

Suppose we're training a single-parameter logistic regression model on the data to the right.




What $\theta$ minimizes mean cross-entropy loss?

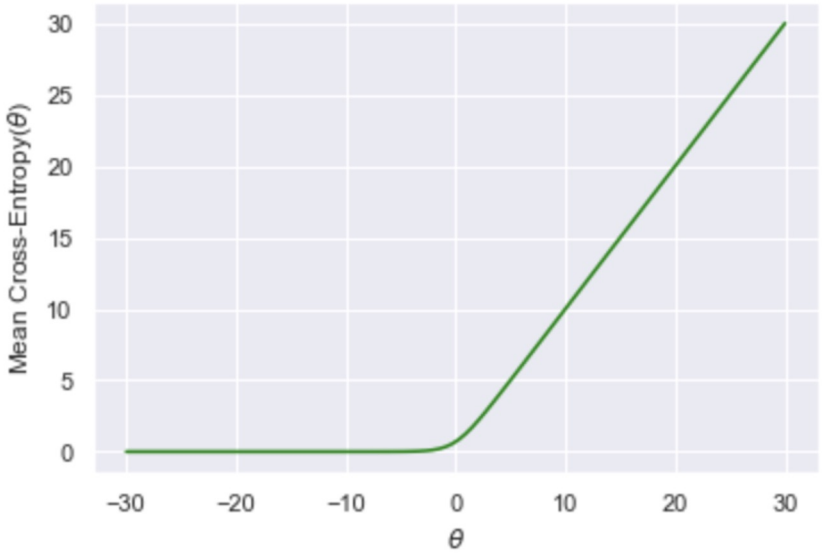
$\hat{\theta} \to -\infty$

$$\hat{y} = f_\theta(x) = P(Y = 1|x) = \sigma(x\theta) \qquad \sigma(t) = \frac{1}{1 + e^{-t}}$$

# Loss surface

On the right is the loss surface for mean cross-entropy loss.

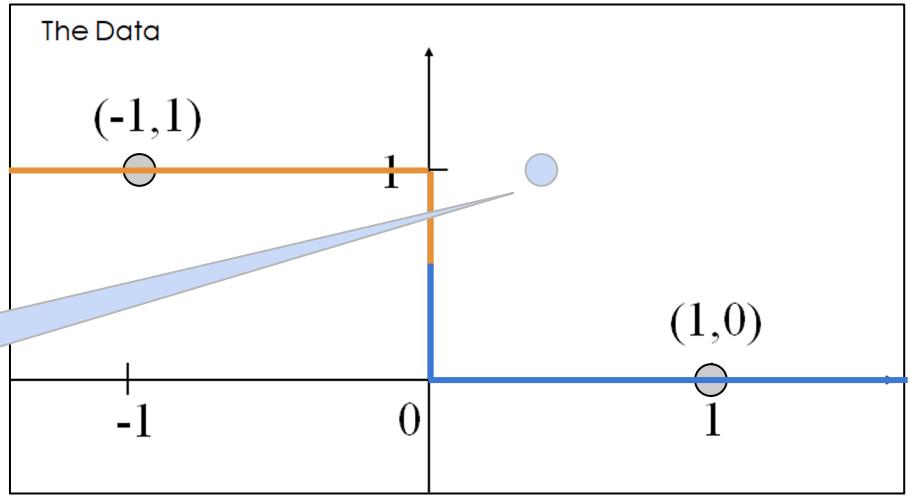
- Gradient descent will (correctly) push our guess for theta towards negative infinity.
- It's almost impossible to see, but that's not a plateau – loss keeps decreasing and decreasing to the left.
  - Loss approaches 0.
- **Why is an infinitely large theta a bad idea?** ( $\hat{\theta} \to -\infty$ )



# Issues with large parameters

**Why is an infinitely large theta a bad idea?** ( $\hat{\theta} \to -\infty$ )

- A single wrong prediction will have infinite loss.
- "Overconfident".




Say we come across a new observation (0.5, 1). This model predicts P(Y = 1 | 0.5) to be 0.

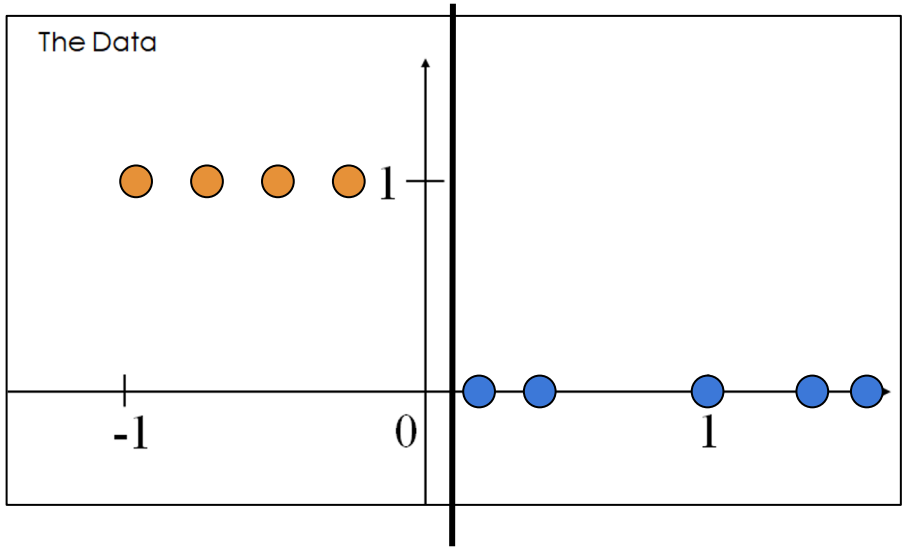
**The cross-entropy loss is infinite!**

# Linear separability

Points are linearly separable if we can correctly separate the classes with a line.

When considering linear separability, the class label does not count as a dimension.

- The data to the left has only one feature, so it is **1D**.
- We are looking for a degree 0 “hyperplane” to separate them, which is a vertical line.



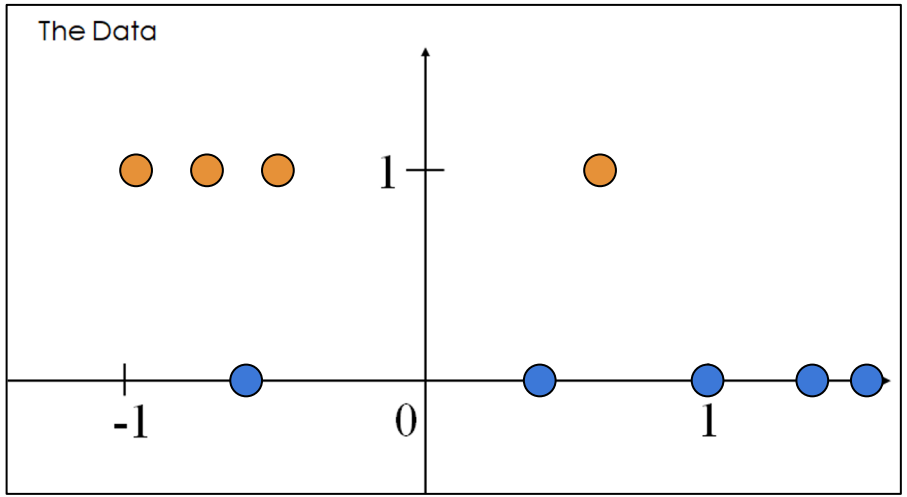

**Linearly separable!**

# Linear separability

Points are linearly separable if we can correctly separate the classes with a line.

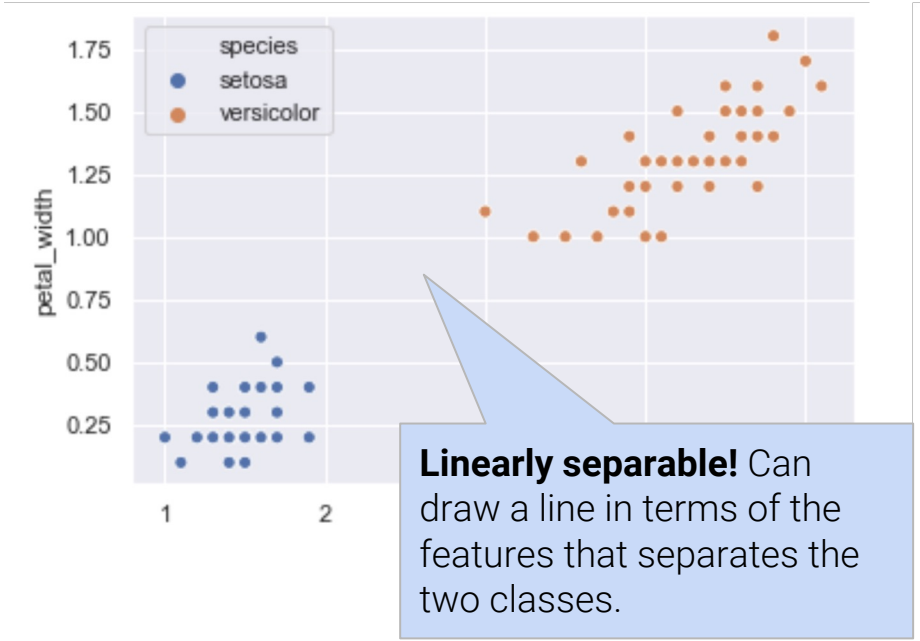
When considering linear separability, we **ignore** the class label.

- The data to the left has only one feature, so it is **1D**.
- We are looking for a degree 0 “hyperplane” to separate them, which is a vertical line.

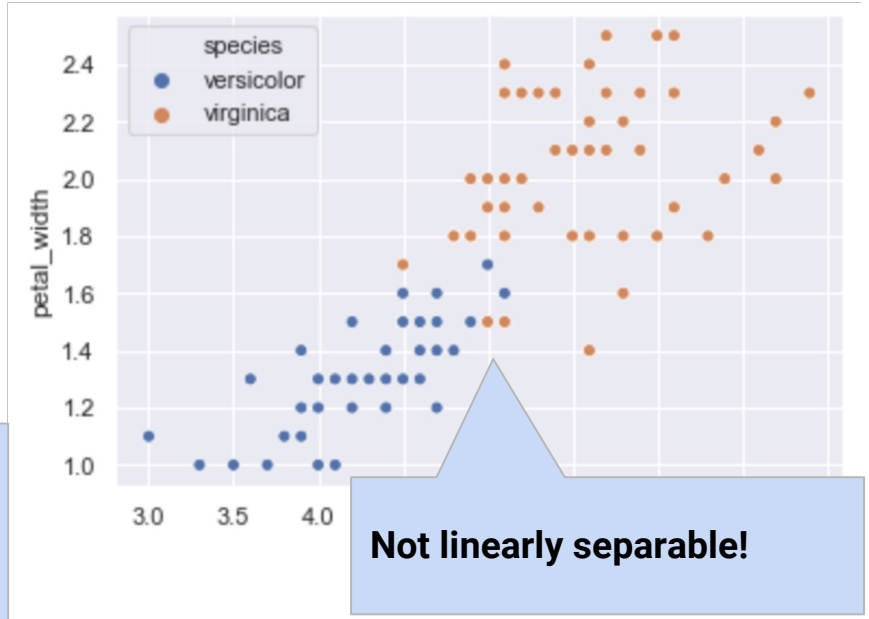



**NOT linearly separable!**

# Linear separability in 2D




**Linearly separable!** Can draw a line in terms of the features that separates the two classes.




**Not linearly separable!**

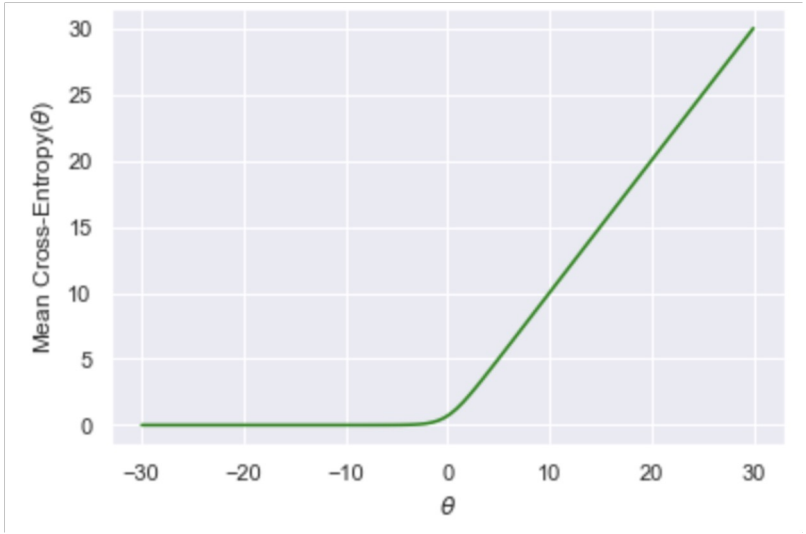
More formally: A set of $d$-dimensional points is **linearly separable** if we can draw a degree $d$-$1$ hyperplane (line) that separates the points perfectly.

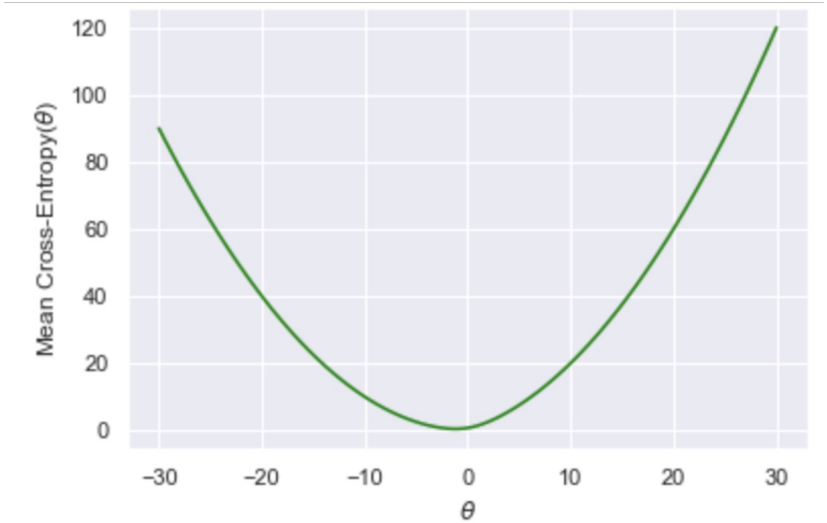
# Regularized logistic regression

- If our training data is linearly separable, some of our weights will diverge to infinity (either positive or negative).
  - This is because our numeric solver (e.g. gradient descent) will keep "rolling" further and further down the loss surface.
  - Will eventually stop at some excessively large weight.
- To avoid large weights, we use **regularization**.
  - As with linear regression, we should standardize our features before applying regularization.
- For instance, using L2 regularization, our objective function becomes:

$$R(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\sigma(\mathbb{X}_i^T\theta)) + (1-y_i)\log(1-\sigma(\mathbb{X}_i^T\theta))\right) + \lambda\sum_{i=1}^{p}\theta_i^2$$

# Regularized logistic regression



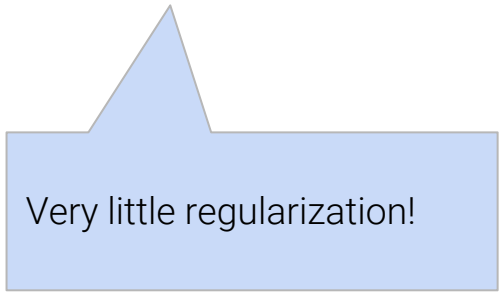

Loss surfaces for linearly separable toy dataset from before.

- Left: no regularization.
- **Right: L2 regularization, with lambda = 0.1**.

# Regularized logistic regression in scikit-learn

- scikit-learn's LogisticRegression package applies regularization by default.
  - L2 by default, but you can change the **`penalty`** parameter.
- But, its regularization hyperparameter **`C`** is the inverse of the lambda that we've discussed.
  - **`C`** = 1 / lambda.
- By default, C = 1.

```
LogisticRegression(C = 300)
```



# Summary

- Logistic regression models the probability of belonging to class 1.
  - Designed for binary classification.
- In order to make classifications, we employ a threshold, or decision rule.
  - Different thresholds yield different decision boundaries.
- To evaluate our models, we can look at several numeric and visual metrics.
  - Accuracy, precision, recall.
  - PR curves, ROC curves.
- Decision boundaries for logistic regression are linear in terms of the model's features.
- Using regularization for logistic regression is a good idea.
  - It is necessary when our data is linearly separable to prevent our weights from diverging.

# Multiclass classification

Thresholding

Evaluating Classifiers

Visual Metrics

Decision Boundaries

Linear separability, Regularization

**Multiclass classification**

# Multiclass classification

Sometimes we have more than one class that we're interested in.

Example, we want to predict what kind of animal an image contains, of the following 5 choices.

- Dog
- Cat
- Lion
- Zebra
- Other

# Multiclass classification: one vs. rest

The simplest way to do multiclass classification is to build N binary classifiers, one for each category.

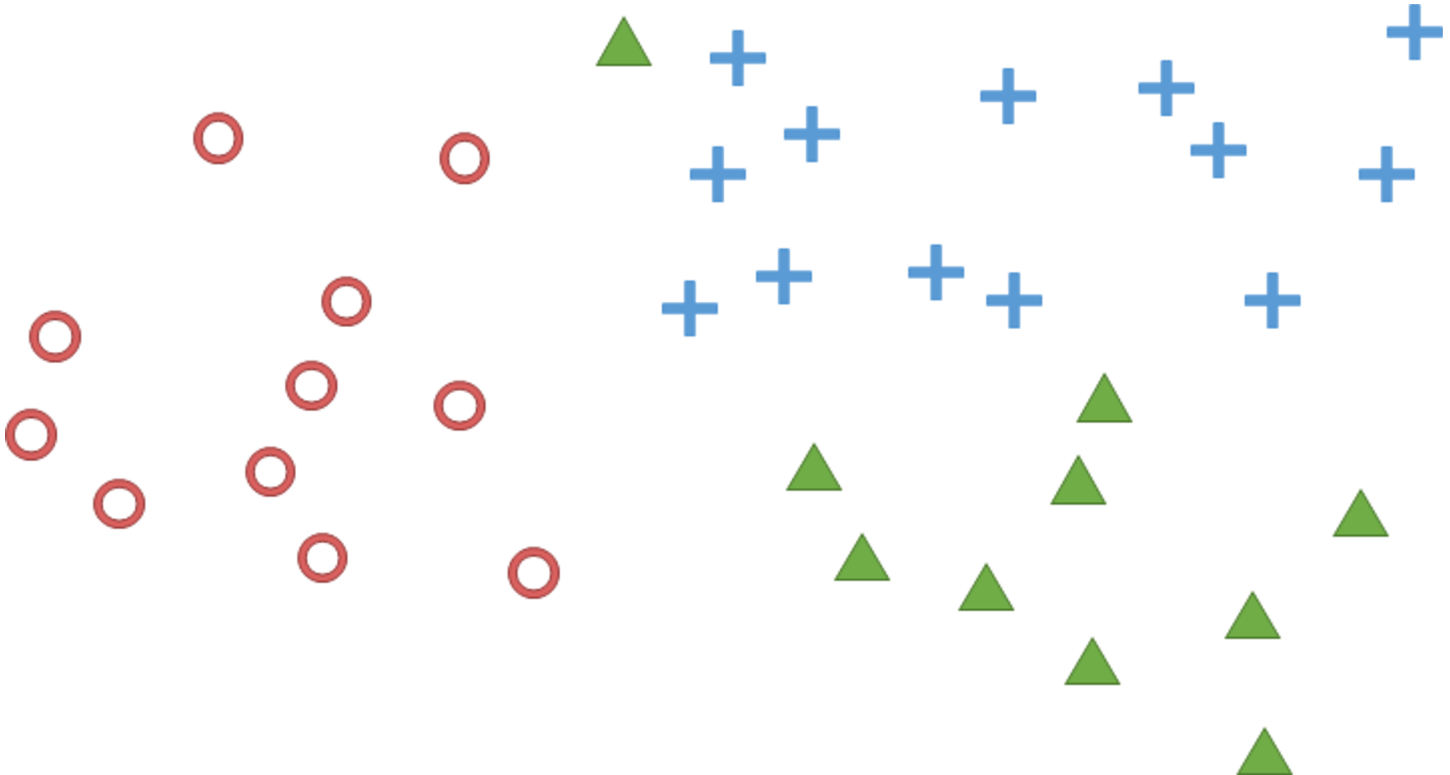
- Resulting prediction will just be whichever classifier gives highest probability.

Example from before:

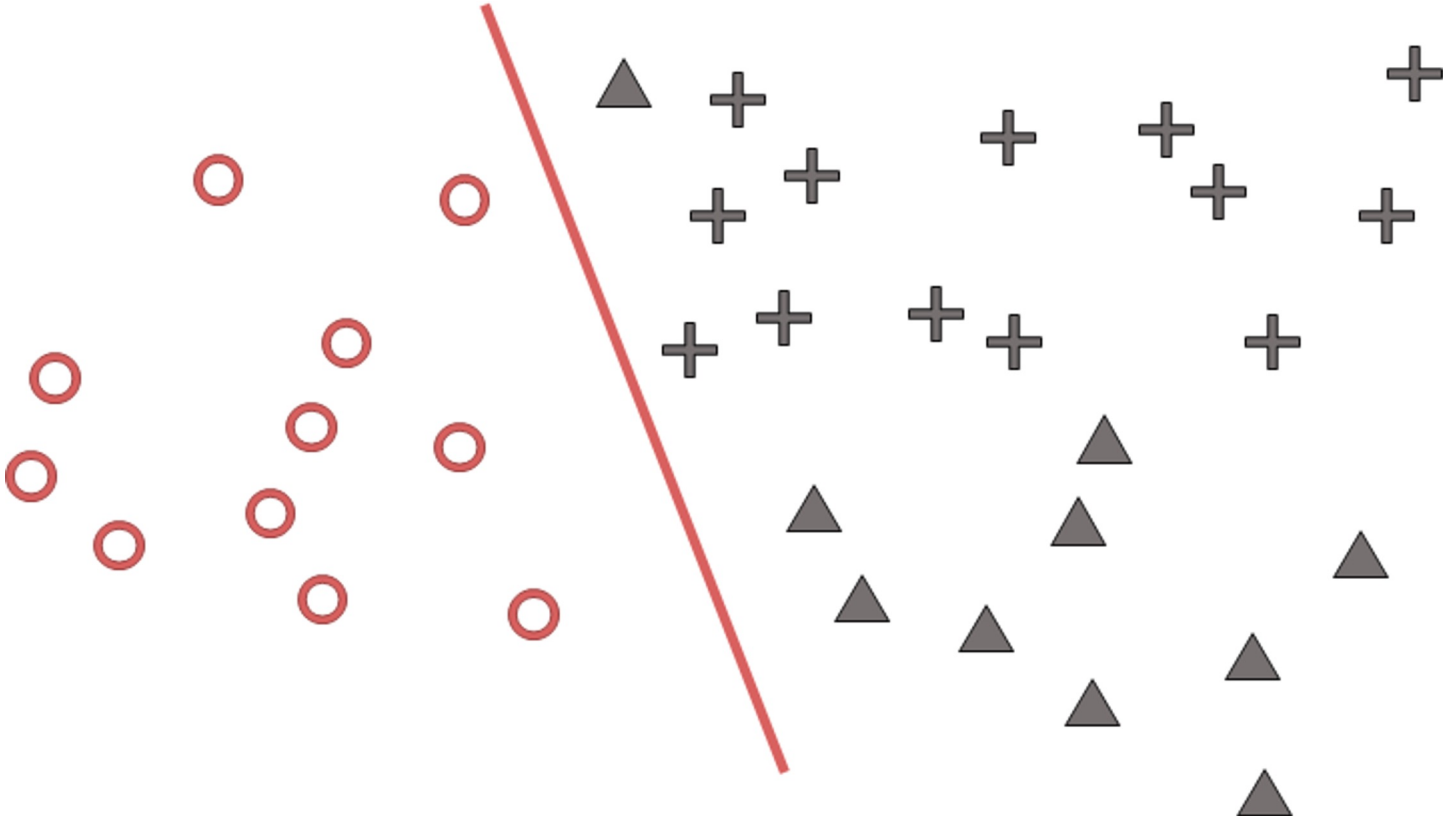
- Build a dog classifier.
- Build an cat classifier.
- Build a lion classifier.
- Build an zebra classifier…

Given a voter, assign the class which has the highest probability among all N.

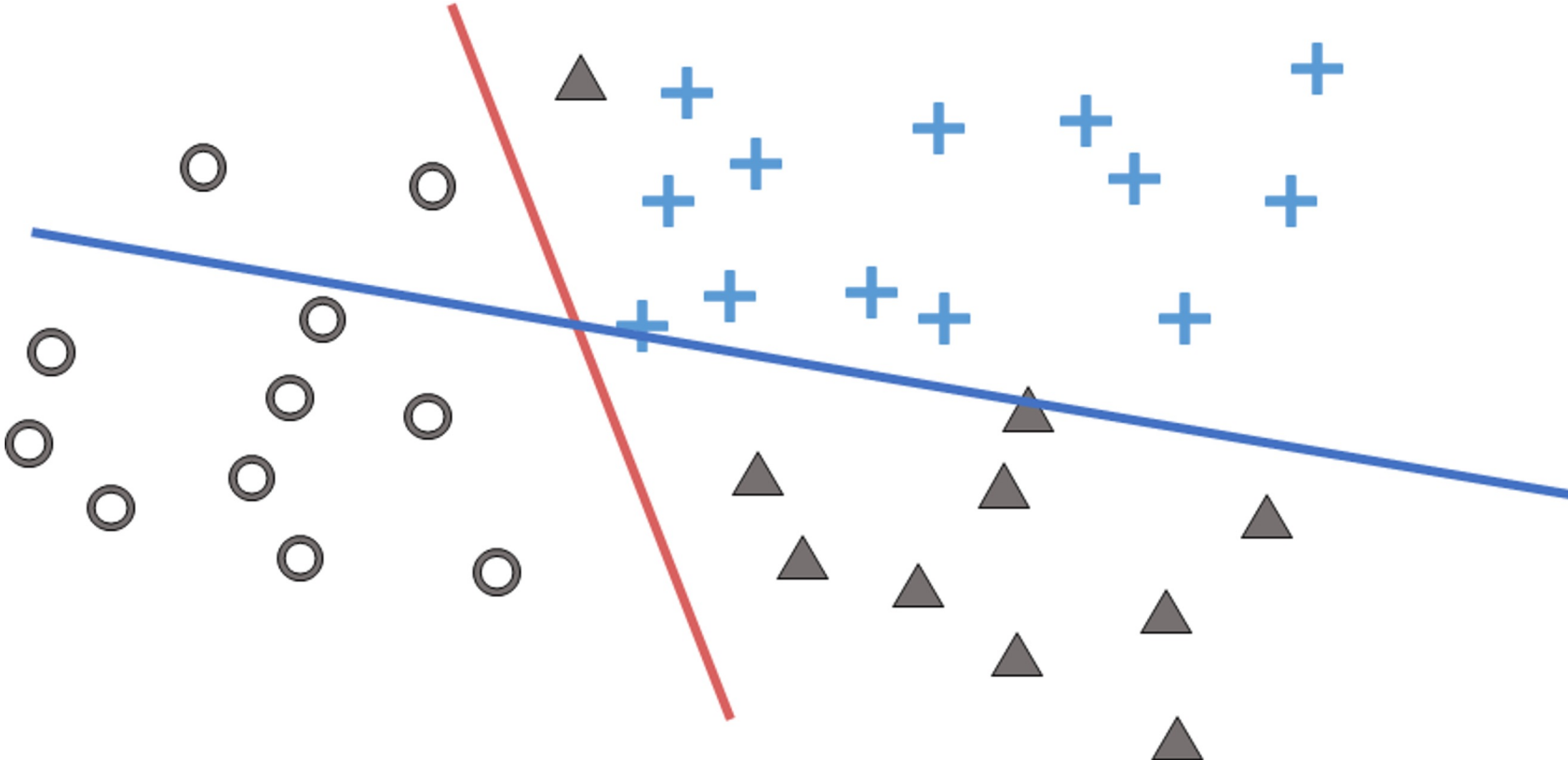
# Visual example



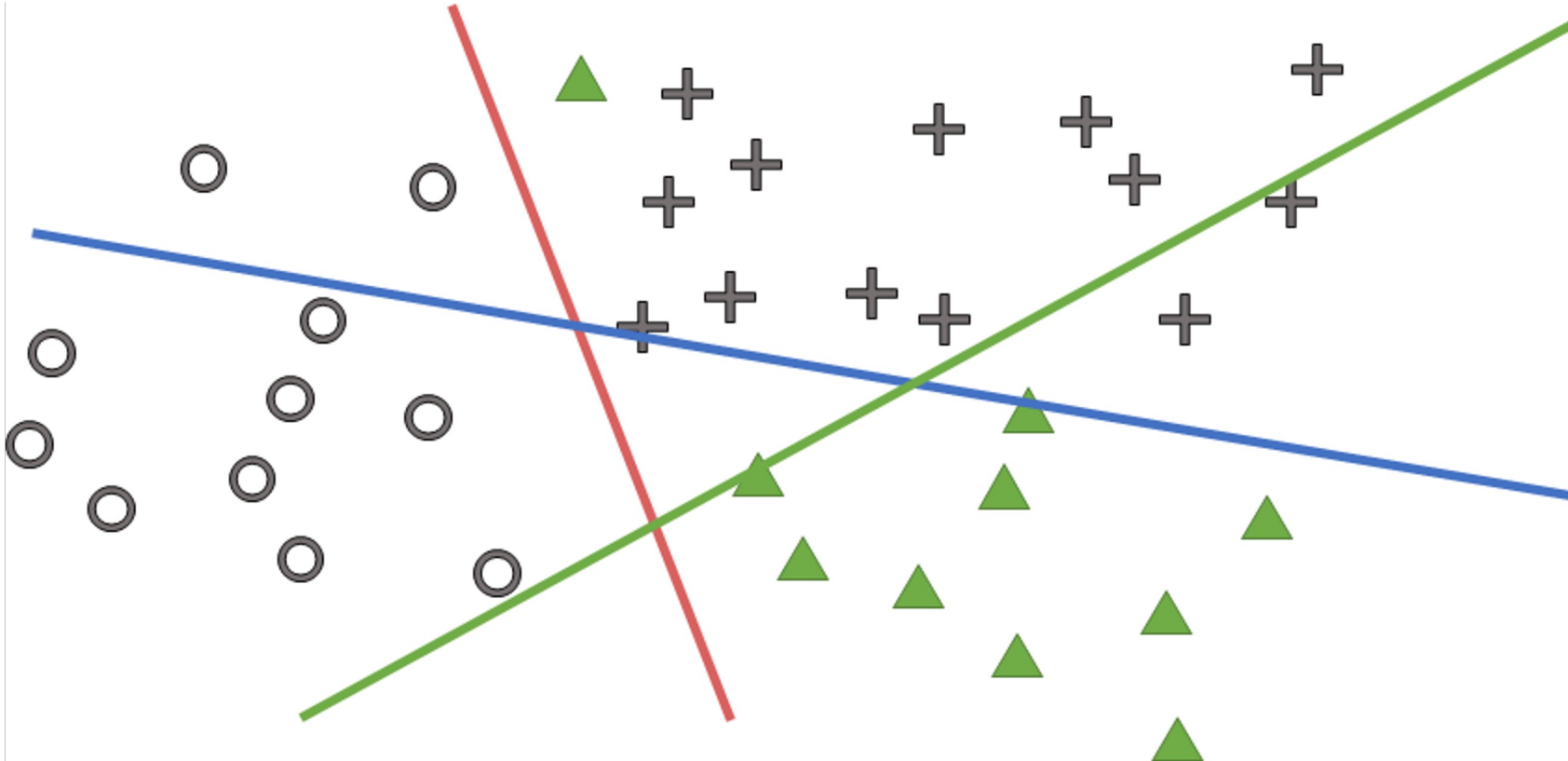
# Visual example



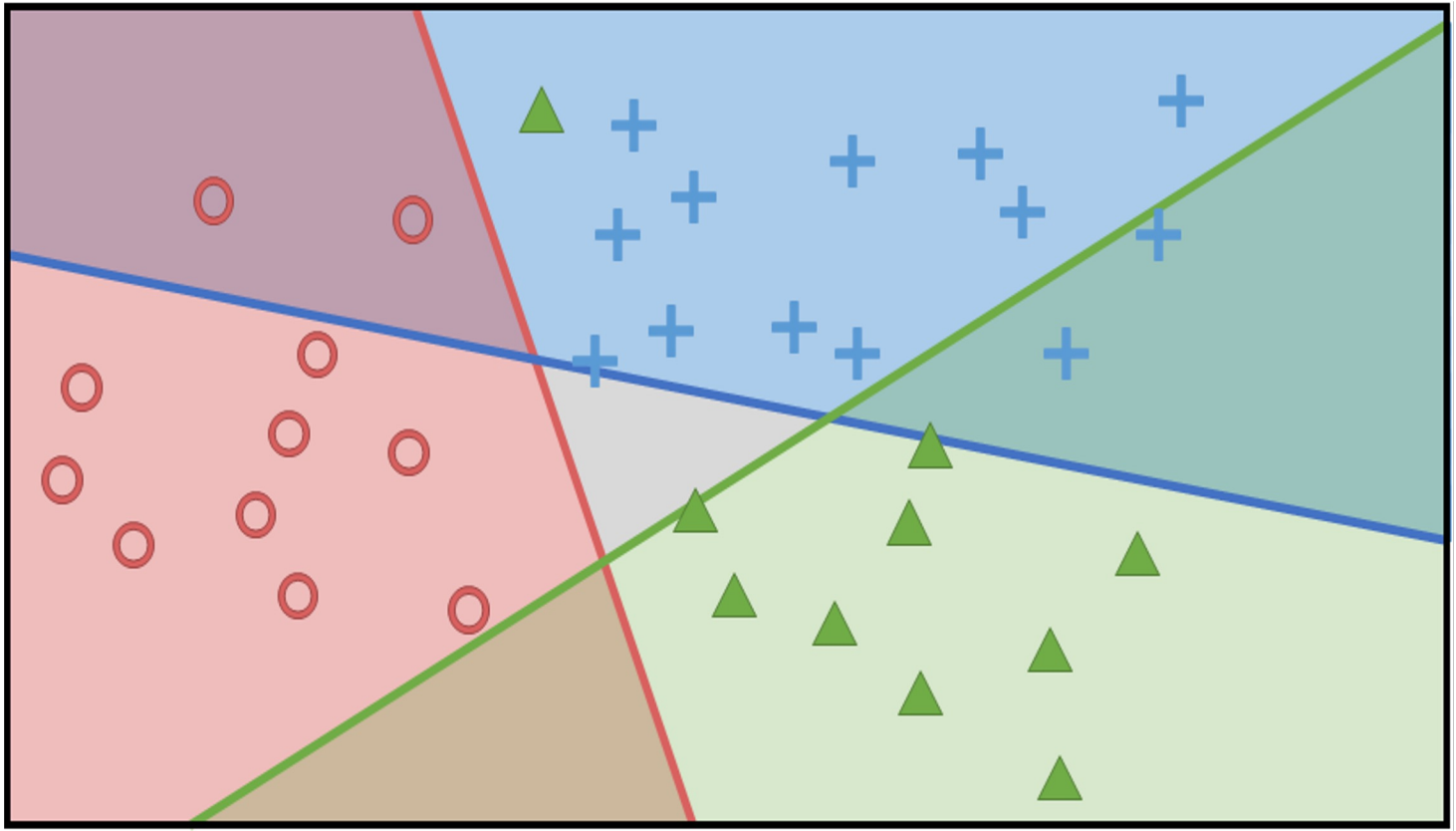
# Visual example



# Visual example



## Visual example



# Multiclass classification: softmax

One downside of building N binary classifiers: Class imbalance.

Alternate techniques exist that we will not discuss.

Example: Softmax.

- Related to neural networks.
- Idea: Different theta for every class, i.e. for class j we have $\theta^{(j)}$ .

$$\mathbf{P}\left(Y = j \,|\, x\right) = \frac{\exp\left(x^T \theta^{(j)}\right)}{\sum_{m=1}^{k} \exp\left(x^T \theta^{(m)}\right)}$$