LECTURE 22

Logistic Regression, Classification

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

Data 100, Fall 2021 @ UC Berkeley

Fernando Pérez and Alvin Wan

(content by Suraj Rampure, Josh Hug, Joseph Gonzalez, Ani Adhikari)

Agenda

- How to convert from probabilities to classifications (1 or 0) by using thresholds.
- Lots of metrics for evaluating logistic regression models and classifiers – accuracy, precision, recall, PR curves, and more.
- Exploring decision boundaries.
- Linear separability and regularization.

As in the last lecture, the concepts will be in the slides, and the coding details will be in the notebook.

Logistic regression

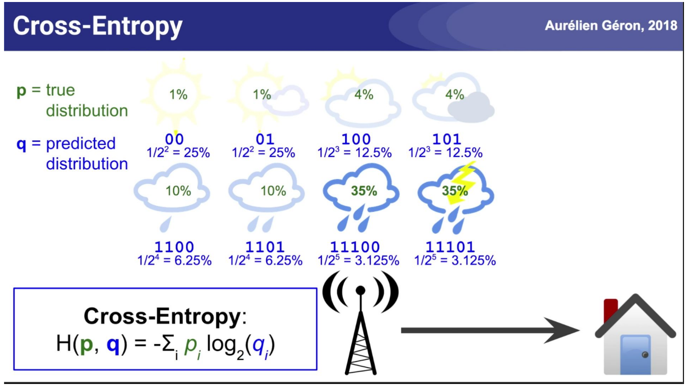
- In a **logistic regression** model, we predict a **binary categorical** variable (class 0 or class 1) as a linear function of features, passed through the logistic function.
  - Our **response** is the probability that our observation belongs to class 1.

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

- 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).

A quick note on Cross-Entropy Loss

- The basic motivation for the L2 and L1 loss:
  - How can I measure the size of an error?
  - I want it to be a difference between prediction and data.
  - I want it to be indifferent to sign
- That leads to both L2 and L1 as reasonable options for numbers and vectors.
- We now want to measure the difference (to compare) between **two probability distributions**.
- The video on the right provides some intuition about this perspective you may find useful.
  - NOT mandatory, but it may help you!



A Short Introduction to Entropy, Cross-Entropy and KL-Divergence
A short (11min) video by Aurélien Géron

Thresholding

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

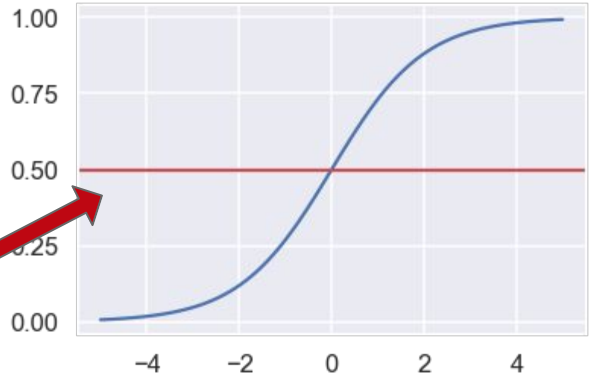
spam or ham

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

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

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 $T = 0.5$:

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

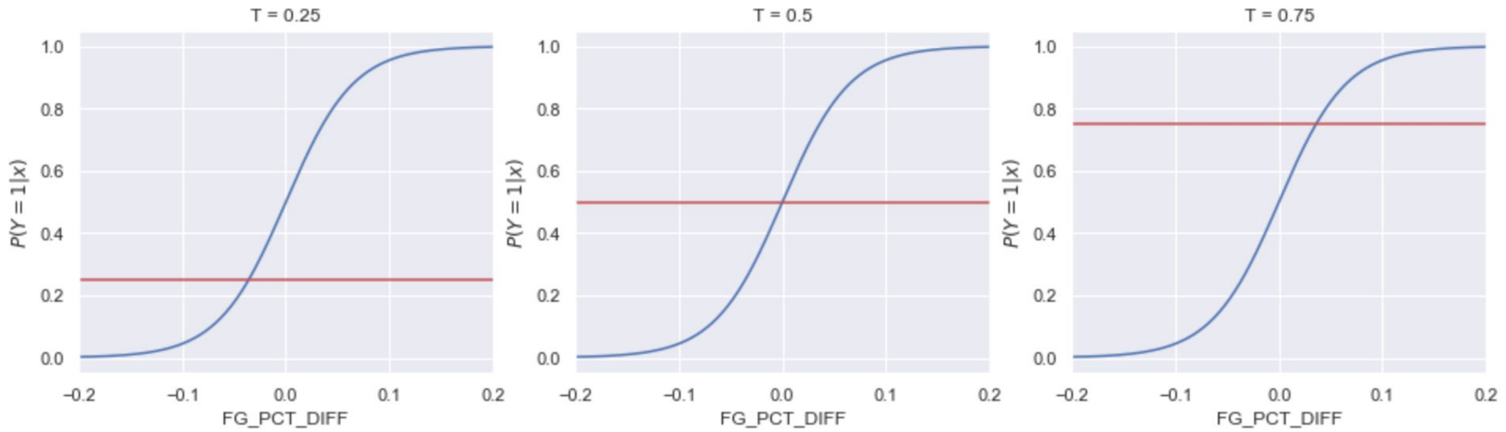
- Note: We **don't need** to set our **threshold** to 0.5. Depending on the type of errors we want to minimize, we can increase or decrease it.
  - 0.5 is the default in `scikit-learn`'s `LogisticRegression`, though.
- Logistic regression paired with a decision rule is a classifier.

Thresholds

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

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

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

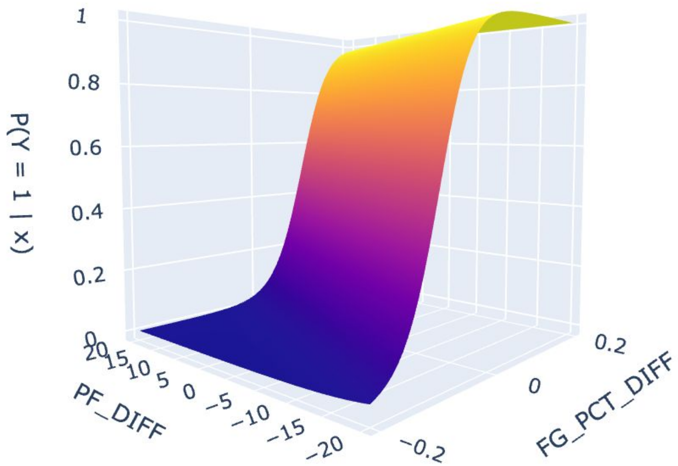


Thresholds in higher dimensions

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.

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



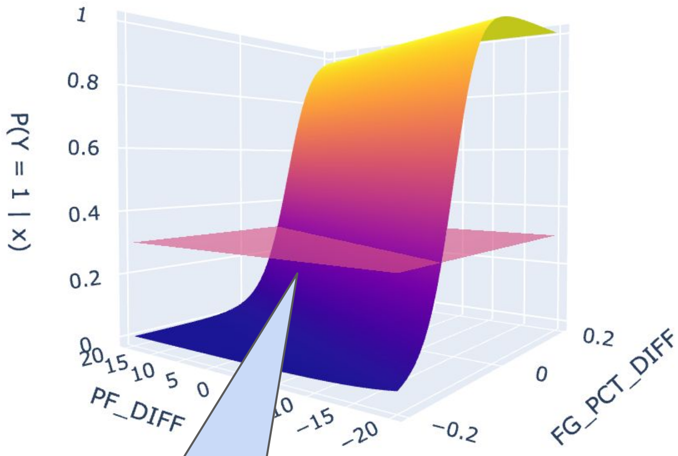
Thresholds in higher dimensions

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.

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

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

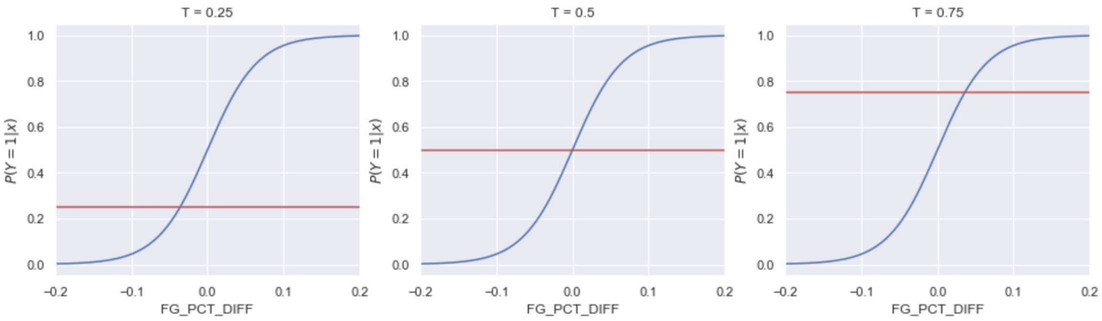


Threshold $T = 0.3$.

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

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**.
  - Widely used.
  - model.score in scikit-learn calculates this.
  - Changing the threshold can change our model's accuracy (will explore soon).
  - In the presence of class imbalance – not so meaningful!

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

Pitfalls of accuracy

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 **5** are truly **spam**, and the remaining **95** are **ham**.

- Your friend 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**.

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

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

- Your friend 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?

Accuracy is 95%.

- But we didn't detect any spam emails!
- Alternative: classify everything as spam.
  - We'd catch all spam emails!
  - But we'd also have a bunch of non-spam emails classified as spam.
- This suggests we need to be looking at more than just accuracy.

Types of classification errors

Moving forward, "positive" refers to 1 and "negative" refers to 0.

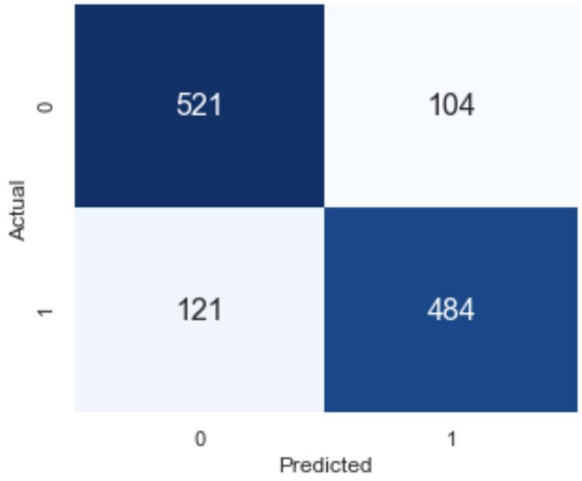
- **True positives and true negatives** are when we correctly classify an observation as being positive or negative.
- **False positives** are like "false alarms."
- **False negatives** are when we "fail to detect" things.

		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.

Confusion matrix

A **confusion matrix** gives us the four quantities on the previous slide, for a particular classifier and set of data.




```
cm = confusion_matrix(games['WON'], y_pred)
sns.heatmap(cm, annot=True, fmt = 'd', cmap = 'Blues', annot_kws = {'size': 16})
```

Prediction

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

Precision and recall

		Prediction	
		0	1
Actual	0	TN	FP
	1	FN	TP

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

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

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

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 positives? Penalizes false negatives.

Precision and recall

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

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

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

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

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 positives? Penalizes false negatives.

Precision and recall

		Prediction	
		0	1
Actual	0	TN	FP
	1	FN	TP

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

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

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

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 positives? Penalizes false negatives.

Precision and recall

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 **5** are truly **spam**, and the remaining **95** are **ham**.

- Your friend suggests you classify every email as ham.
- What is the **accuracy?**
- What is the **precision?**
- What is the **recall?**

Accuracy is 95%.

- But we didn't detect any spam emails!
- Alternative: classify everything as spam.
  - We'd catch all spam emails!
  - But we'd also have a bunch of non-spam emails classified as spam.
- This suggests we need to be looking at more than just accuracy.

Precision and recall

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 **5** are truly **spam**, and the remaining **95** are **ham**.

- Your friend suggests you classify every email as ham.
- What is the **accuracy?**
- What is the **precision?**
- What is the **recall?**

TP = 0, FP = 0, TN = 95, FN = 5

Accuracy = **95%** $\frac{0+95}{100}$

Precision = 0 / (0 + 0) = **undefined**

Recall = 0 / (0 + 5) = **0%**

Accuracy doesn't tell the full story.

- **Class imbalance** – the distribution of true observations is skewed.
- Here, 95% of true observations are negative.

Trade-off between precision and recall

Prediction

Actual	0	1
0	TN	FP
1	FN	TP

- Precision penalizes false positives, and recall penalizes false negatives.
- We can achieve **100% recall** by making our classifier output “1”, regardless of the input.
  - 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. **Precision tends to increase.**
  - **Lower threshold** – fewer false negatives. **Recall increases**.

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?

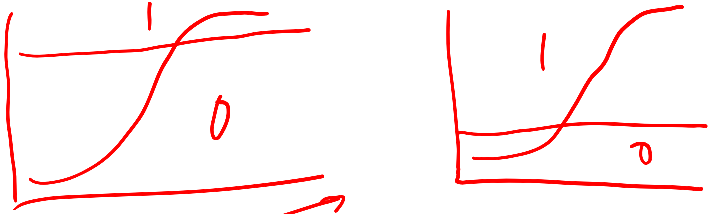
- Predicting whether or not a patient has some disease.
  - Maximize **recall**.
  - Presumably if we say someone has the disease, they will go through further testing.
  - If we say they don't, the condition may be left untreated, which is dangerous.
- Determining whether or not someone should be sentenced 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.

Visual metrics

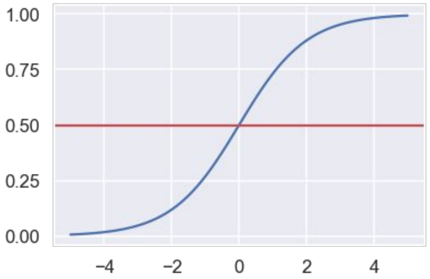
The effect of thresholds

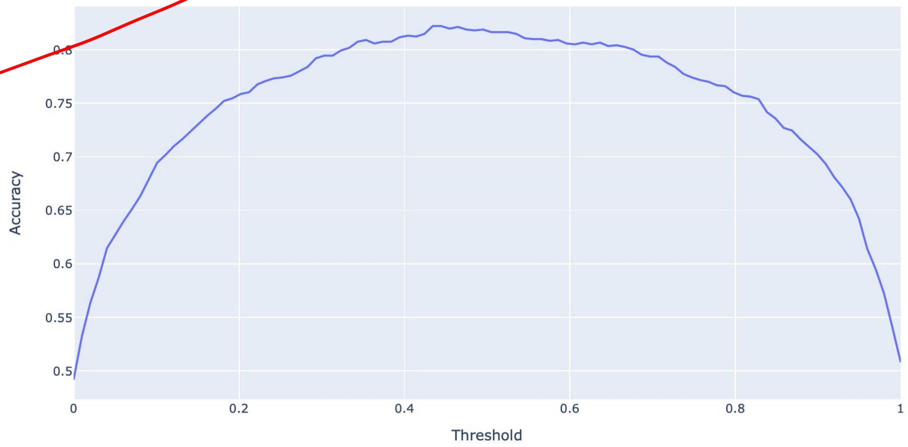
- Our choice of threshold impacts our model's:
    - Accuracy.
    - Precision.
    - Recall.
- Let's explore!

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.





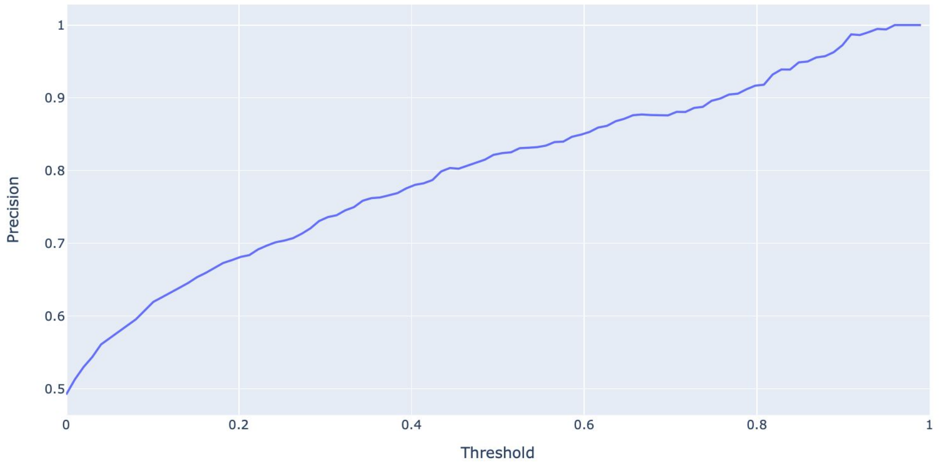
Accuracy vs. threshold for our two feature NBA model, $P(Y=1|x) = \sigma(\theta_0 + \theta_1 \cdot \mathrm{FG\_PCT\_DIFF} + \theta_2 \cdot \mathrm{PF\_DIFF})$

Precision vs. threshold

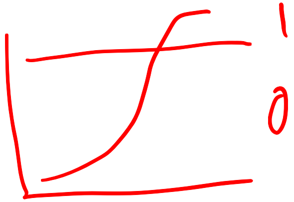
- As we increase our threshold, we have fewer and fewer false positives.
- Thus, precision tends to increase.

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

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



Recall vs. threshold

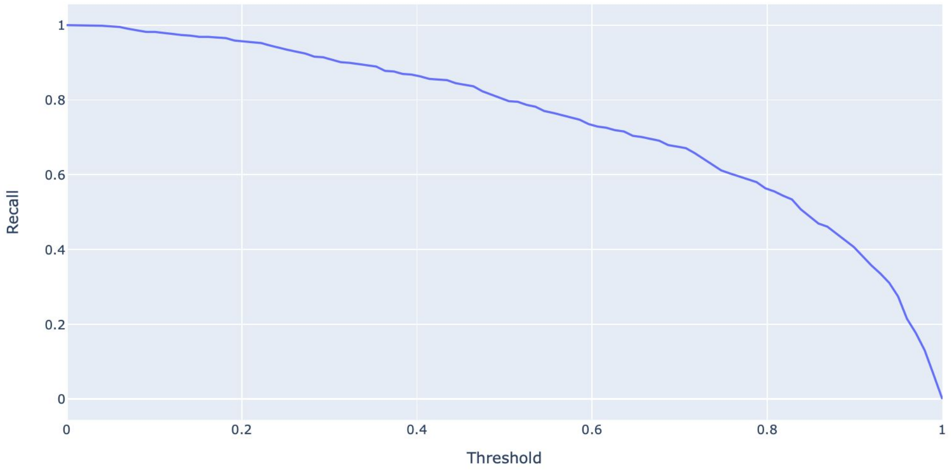


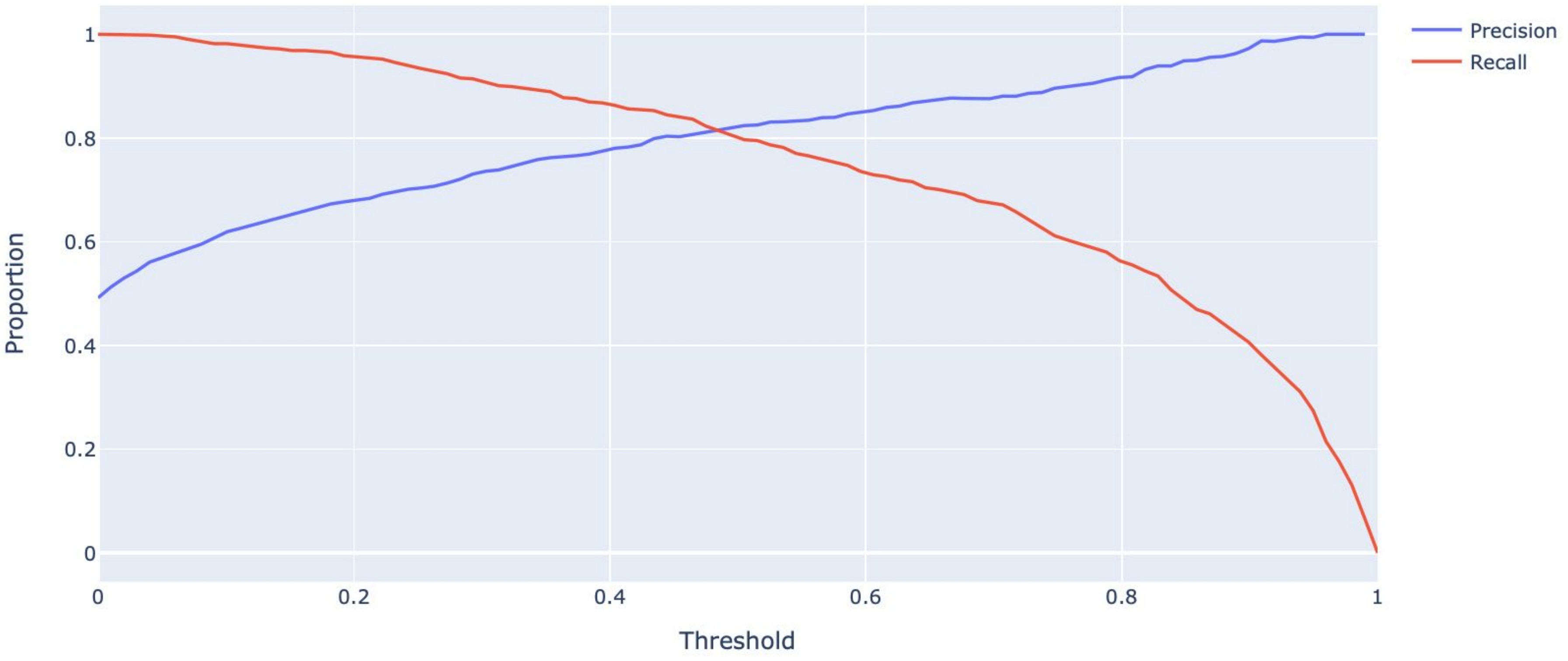
- As we increase our threshold, we have more and more false negatives.
- Thus, recall tends to decrease.

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

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

Recall strictly decreases as we increase our threshold. Why?

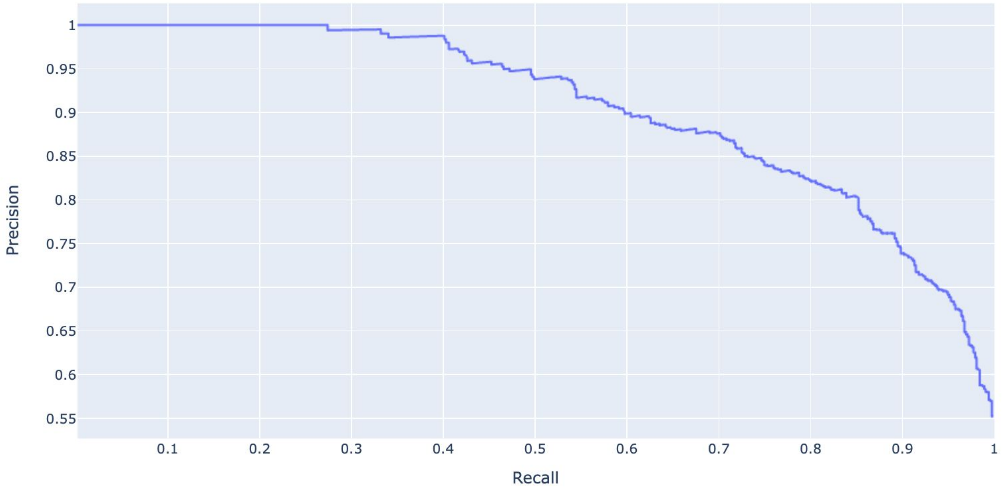




Precision
Recall
Proportion
Threshold
0
0.2
0.4
0.6
0.8
1

Precision-recall curves

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



Question:

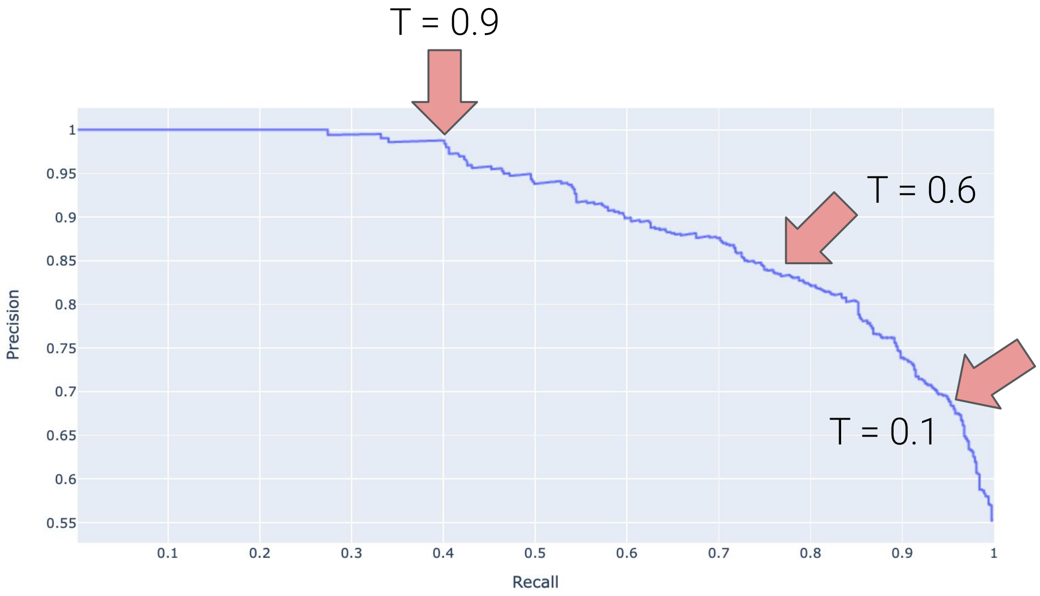
- Which part of this curve corresponds to T = 0.9?
- Which part of this curve corresponds to T = 0.1?

Precision-recall curves

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

Answer:

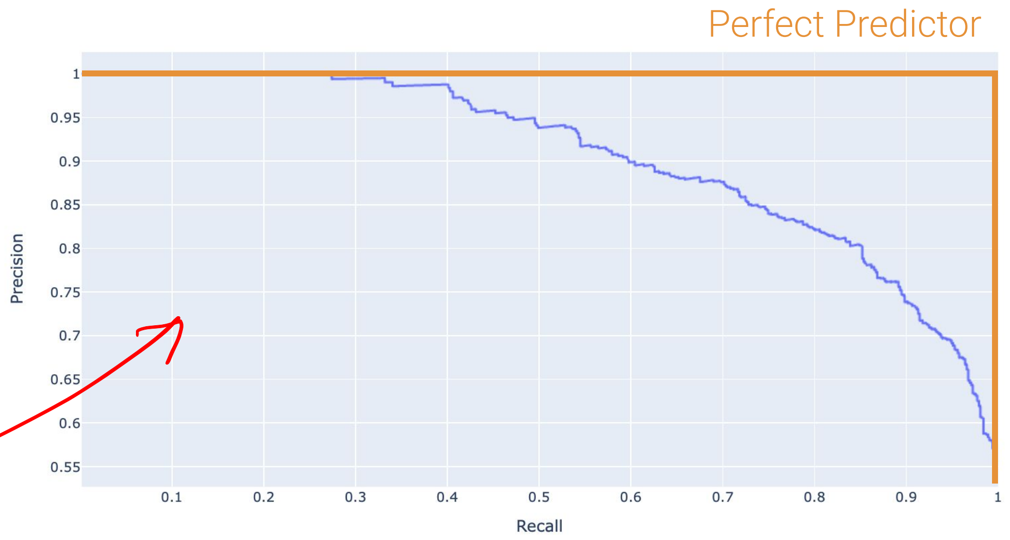
- Threshold decreases from the top left to the bottom right.
- In the notebook, there's an interactive version of this plot.



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.



Other metrics

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

False Positive Rate (FPR):

- FP / (FP + TN)
- “What proportion of innocent people did I convict?”

True Positive Rate (TPR):

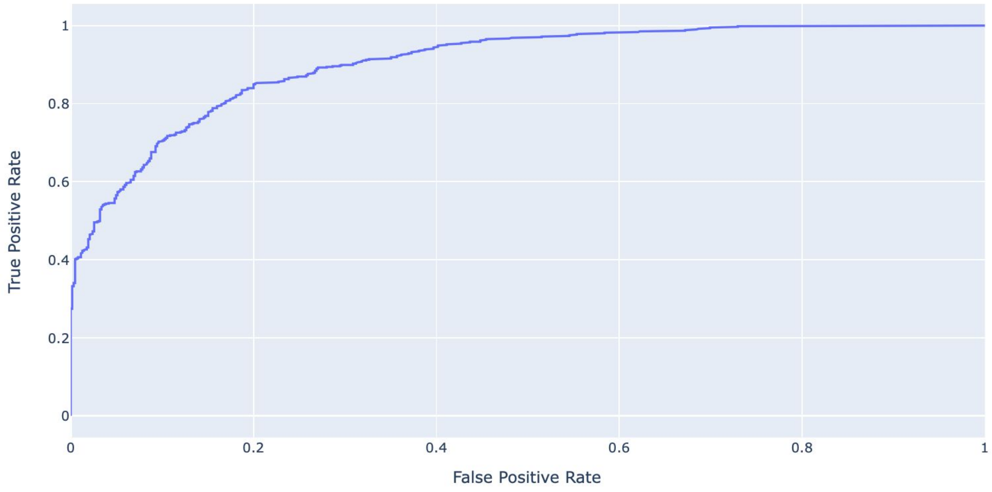
- TP / (TP + FN)
- “What proportion of guilty people did I convict?”
- Same thing as recall.

As we increase our threshold, what happens to FPR? TPR?

ROC curves

A ROC curve plots TPR vs. FPR.

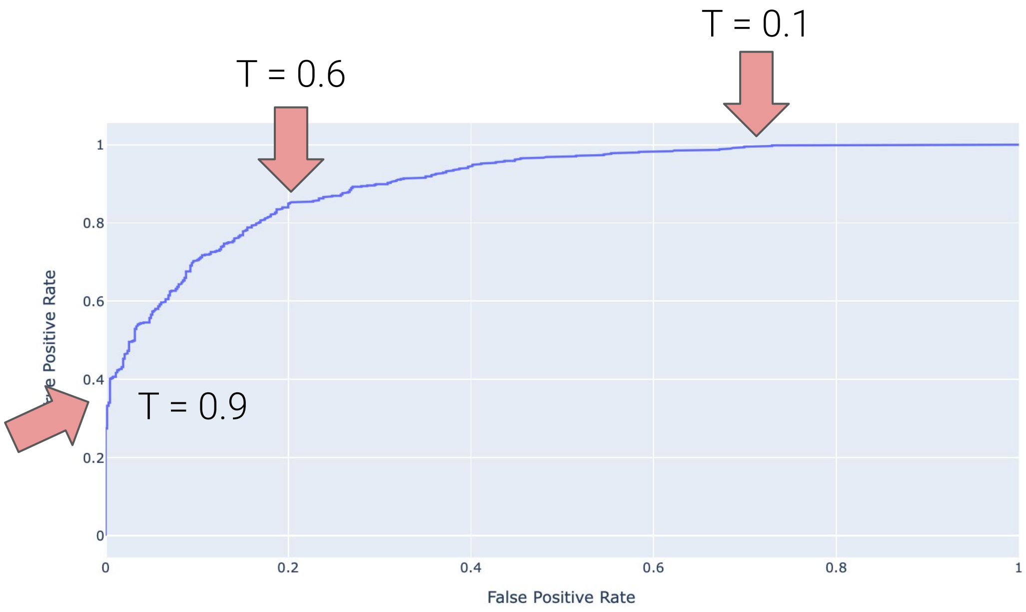
- As we increase our threshold, both TPR and FPR decrease.
  - A decreased TPR is bad (detecting fewer positives).
  - A decreased FPR is good (fewer false positives).
  - Tradeoff!
- ROC stands for “Receiver Operating Characteristic.”



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

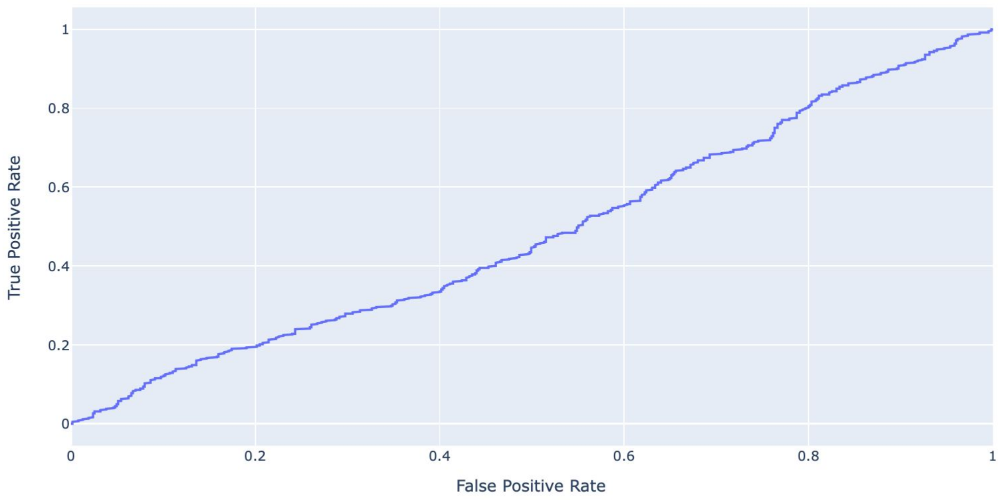
The “perfect” classifier is the one that has a TPR of 1, and FPR of 0.

- 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.

Random predictor: AUC ≅ 0.5



Common techniques for evaluating classifiers

Numerical assessments:

- **Accuracy, precision, recall, 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.

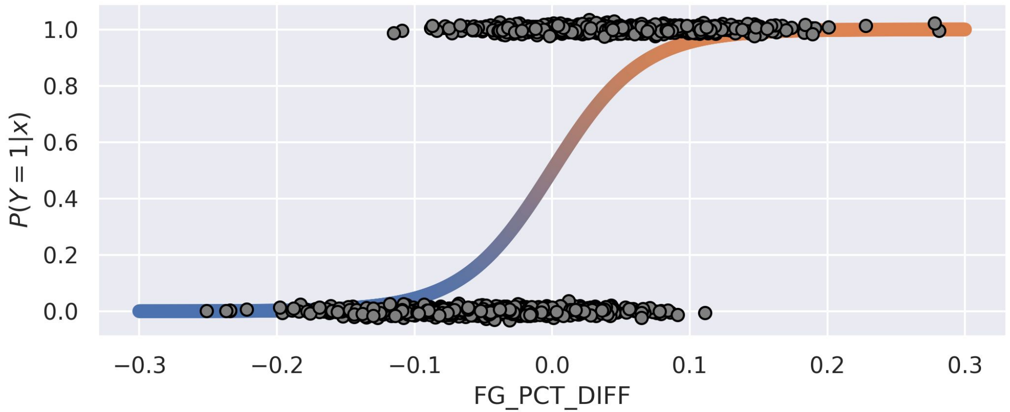
Decision boundaries

Decision boundaries

Consider our original single-feature model.

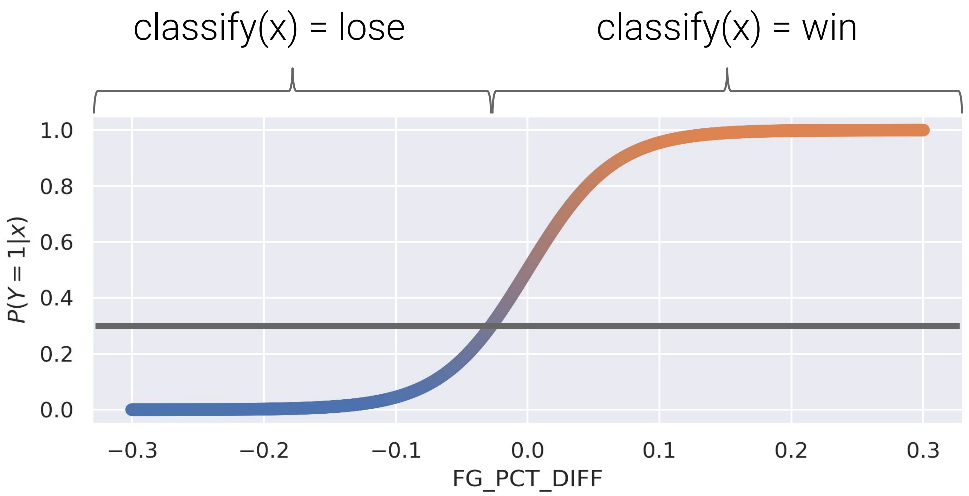
$$P(Y = 1|x) = \sigma(\theta_1 \cdot \mathbf{FG\_PCT\_DIFF})$$

The grey dots are true observations from the 2017-18 NBA season.



Decision boundaries

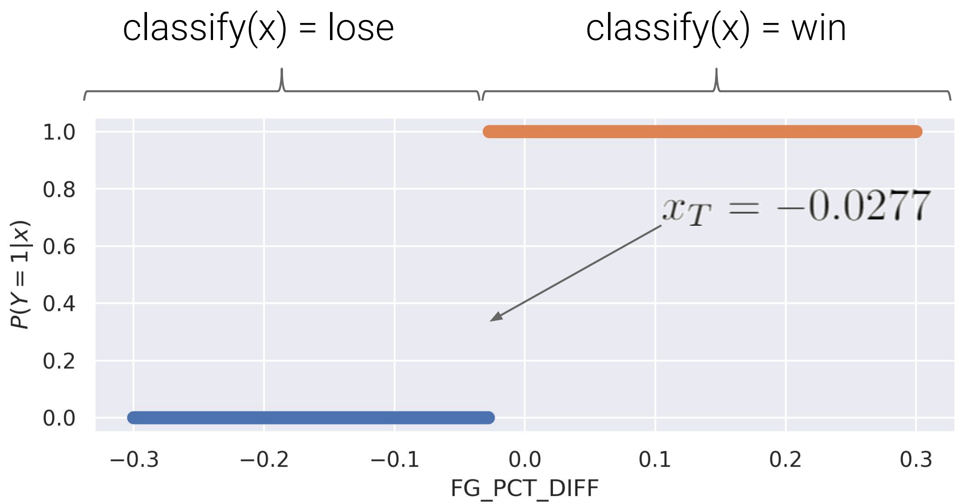
If we pick a threshold, e.g. T = 0.3, we get a predicted class from probabilities.



Decision boundaries

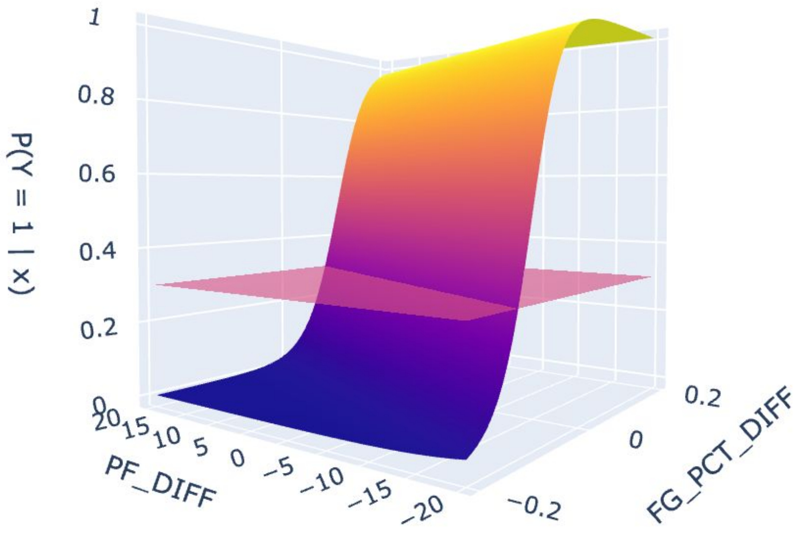
If we pick a threshold, e.g. T = 0.3, we get a predicted class from probabilities.

- 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.



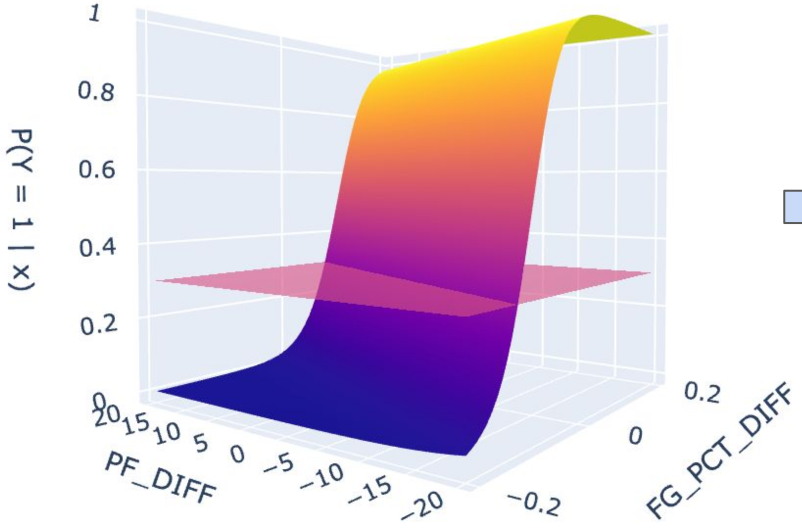
Decision boundaries for 2D models

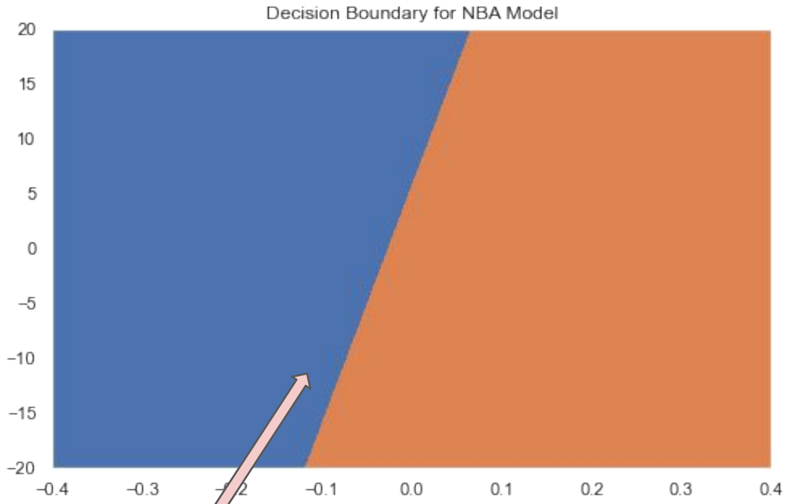
Now consider our "better" model, $P(Y=1|x) = \sigma(\theta_0 + \theta_1 \cdot \mathbf{FG\_PCT\_DIFF} + \theta_2 \cdot \mathbf{PF\_DIFF})$. It is drawn below with the thresholding line T = 0.3. **What does its decision boundary look like?**



Decision boundaries for 2D models

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





Line depends on θ_0, θ_1, θ_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

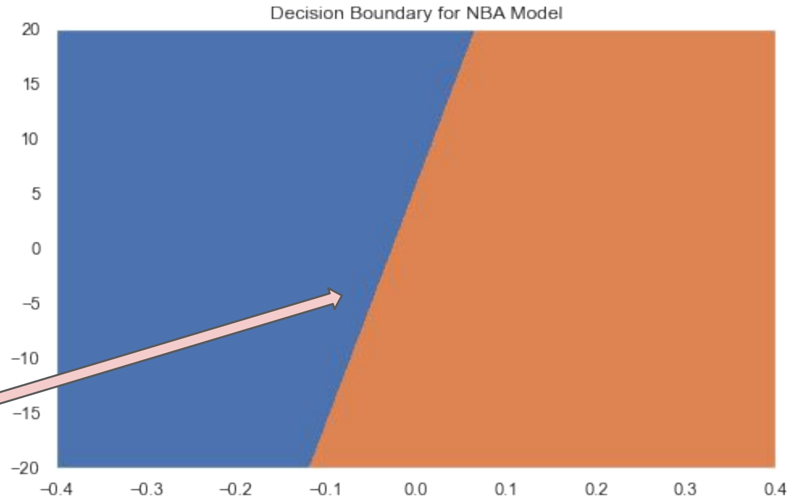
$$P(Y = 1|x) = \sigma(0.035 + 34.705 \cdot \text{FG\_PCT\_DIFF} - 0.160 \cdot \text{PF\_DIFF})$$

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

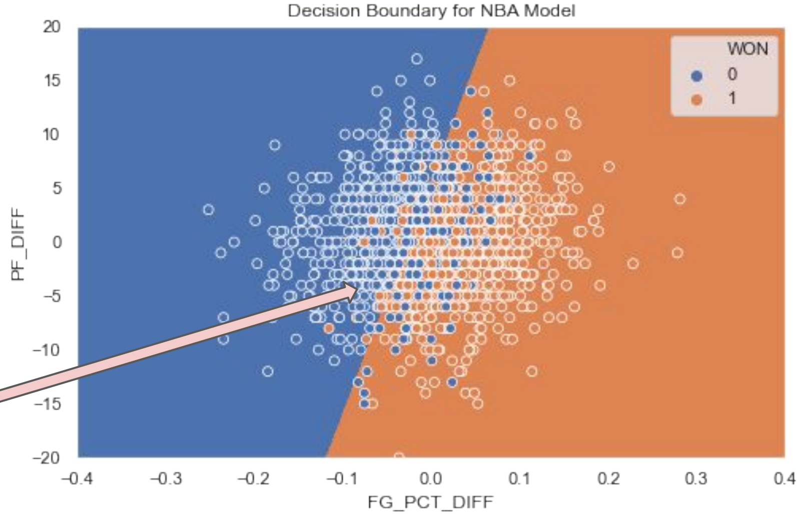
$$\boxed{0.035 + 34.705 \cdot \text{FG\_PCT\_DIFF} - 0.160 \cdot \text{PF\_DIFF} = \sigma^{-1}(0.3)}$$



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.



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

Linear separability, regularization

Question

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

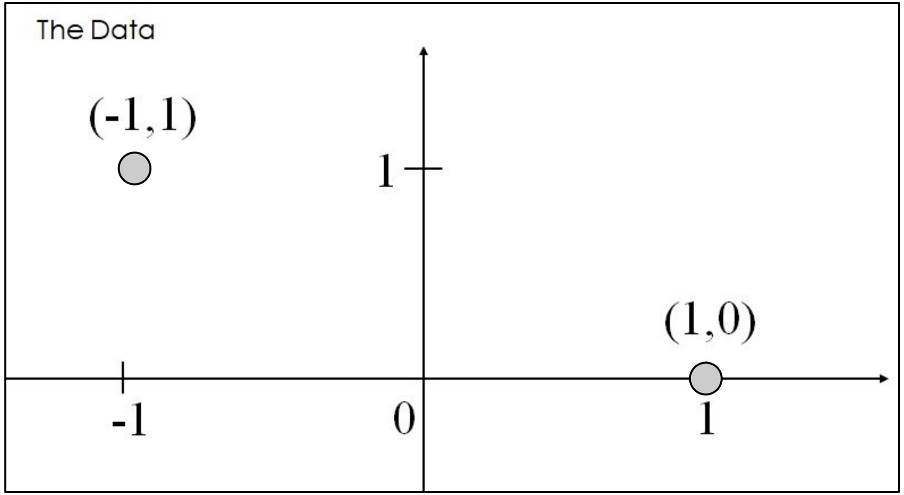
What θ minimizes mean cross-entropy loss?

$\hat{\theta} = -1$

$\hat{\theta} = 1$

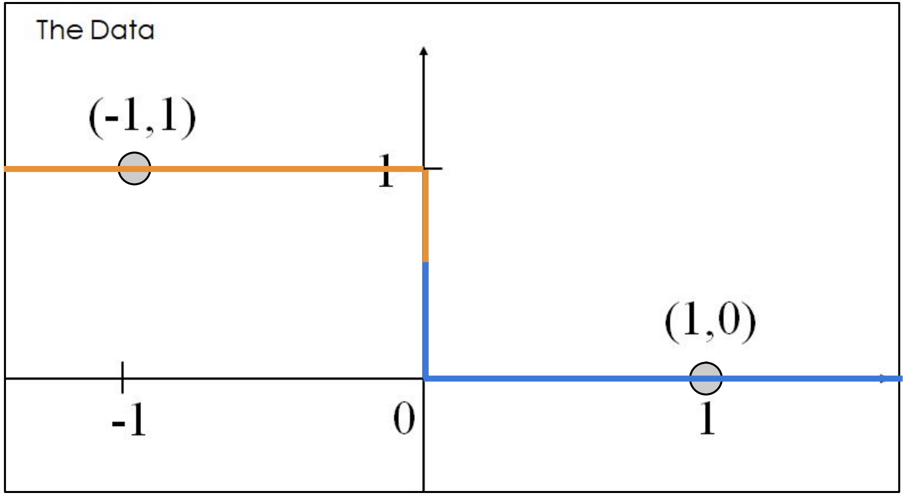
$\hat{\theta} \rightarrow -\infty$

$\hat{\theta} \rightarrow \infty$



Question

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



What θ minimizes mean cross-entropy loss?

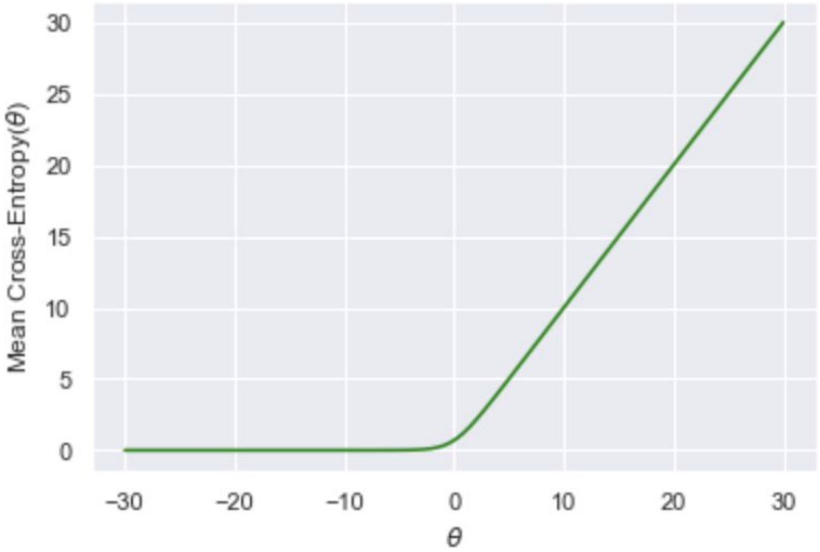
$$\boxed{\hat{\theta} \to -\infty}$$

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

Loss surface

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

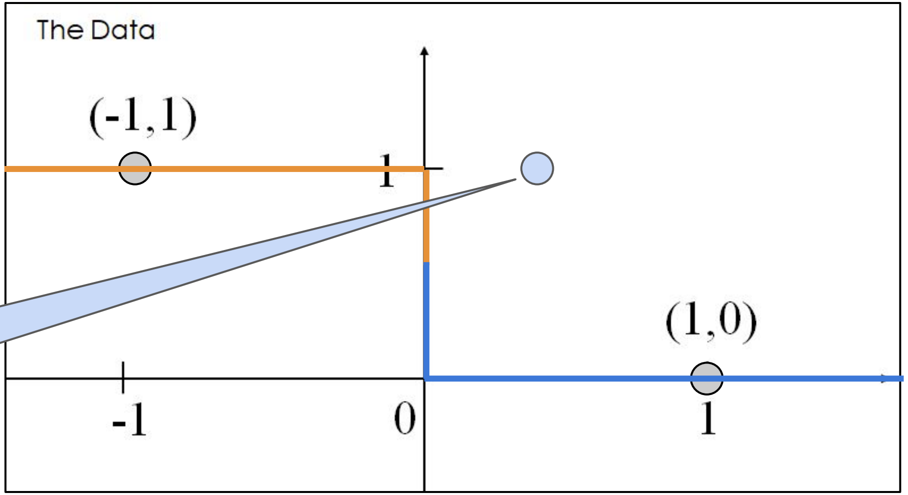
- Gradient descent will (correctly) push our guess for theta towards negative infinity.
- It's almost impossible to see, but that's not a plateau – loss keeps decreasing and decreasing to the left.
    - Loss approaches 0.
- **Why is an infinitely large theta a bad idea?** ($\hat{\theta} \to -\infty$)



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.

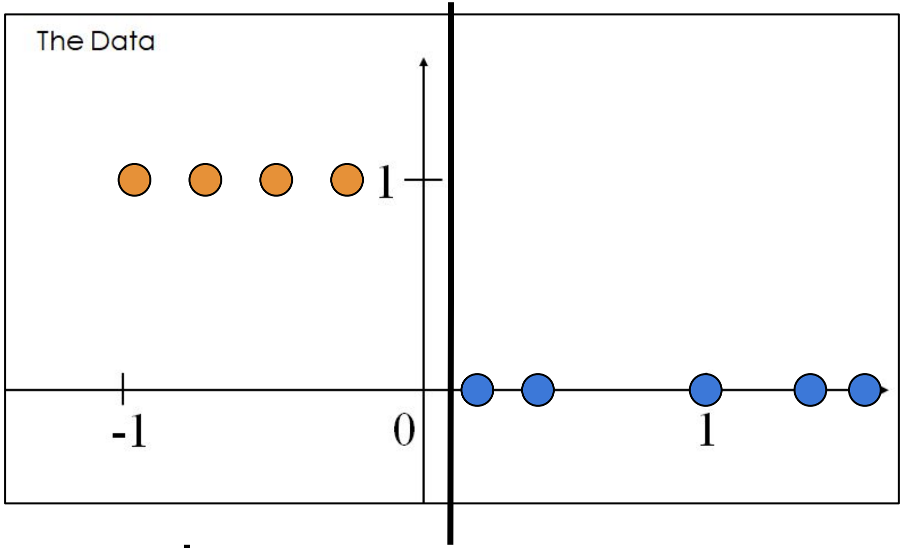
The cross-entropy loss is infinite!

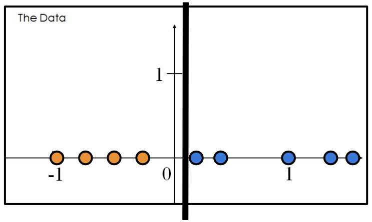
Linear separability

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** (see bottom).
- We are looking for a degree 0 “hyperplane” to separate them, which is a single point (illustrated as a vertical line across that point).





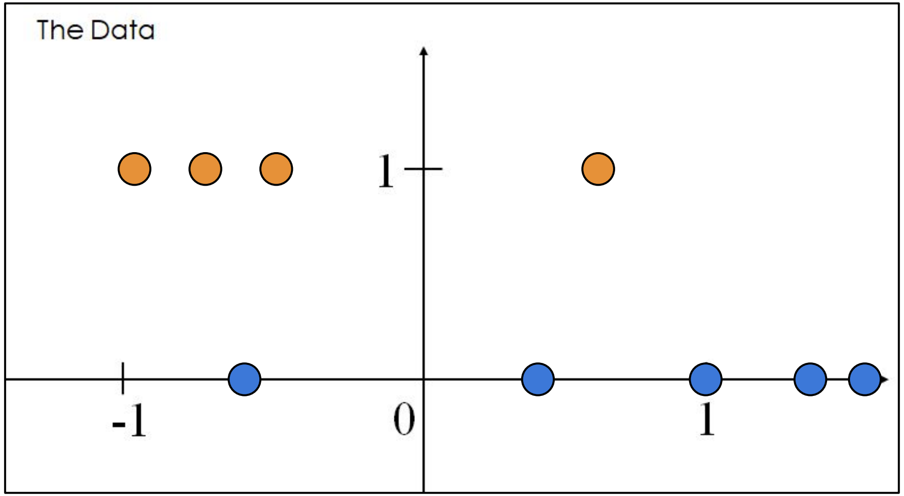
Linearly separable!

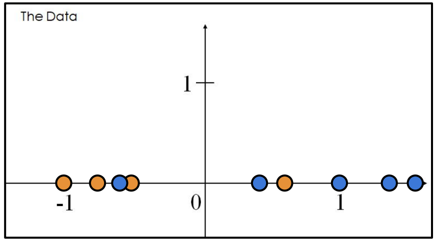
Linear separability

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.

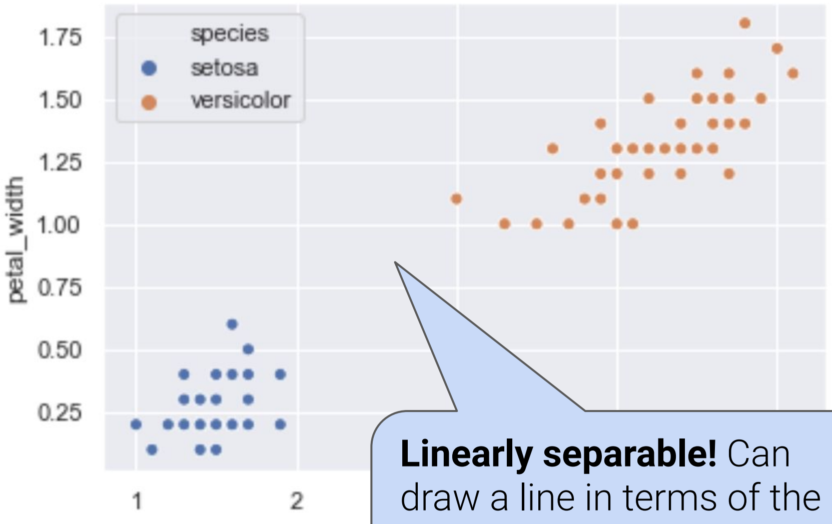
- The data to the left has only one feature, so it is **1D**.
- We are looking for a degree 0 "hyperplane" to separate them, which is a single point (illustrated as a vertical line across that point).



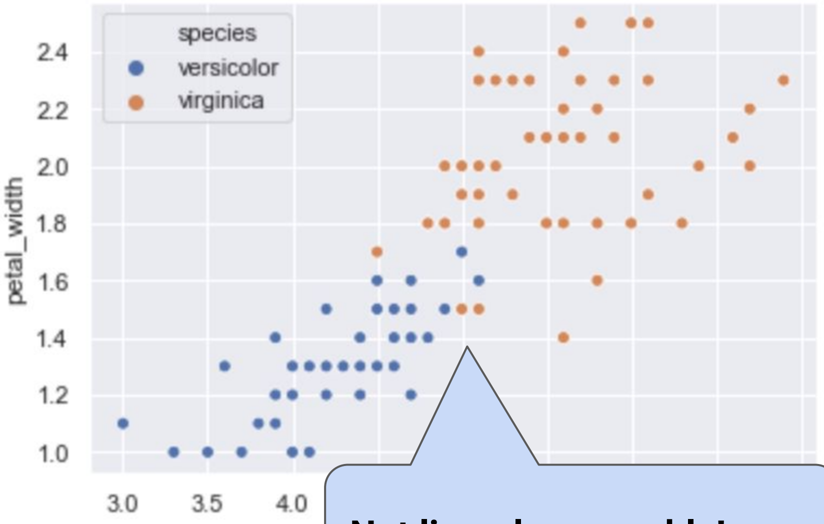


NOT linearly separable!

Linear separability in 2D



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



Not linearly separable!

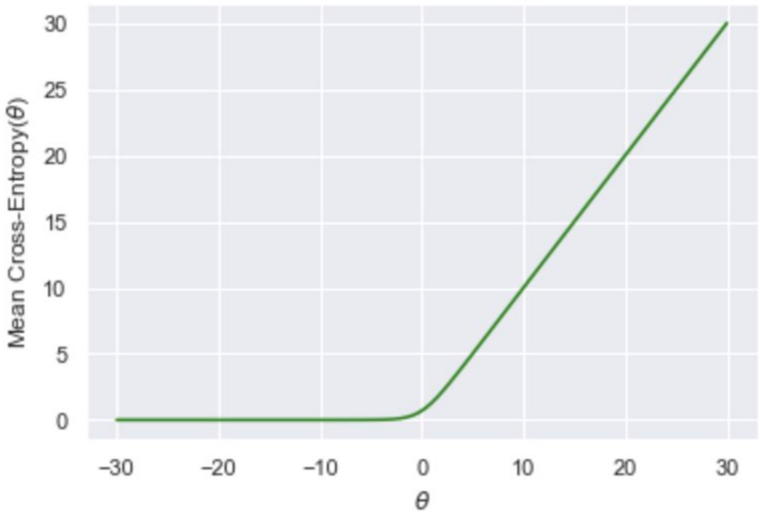
More formally: A set of d-dimensional points is **linearly separable** if we can draw a degree d-1 hyperplane (line) that separates the points perfectly.

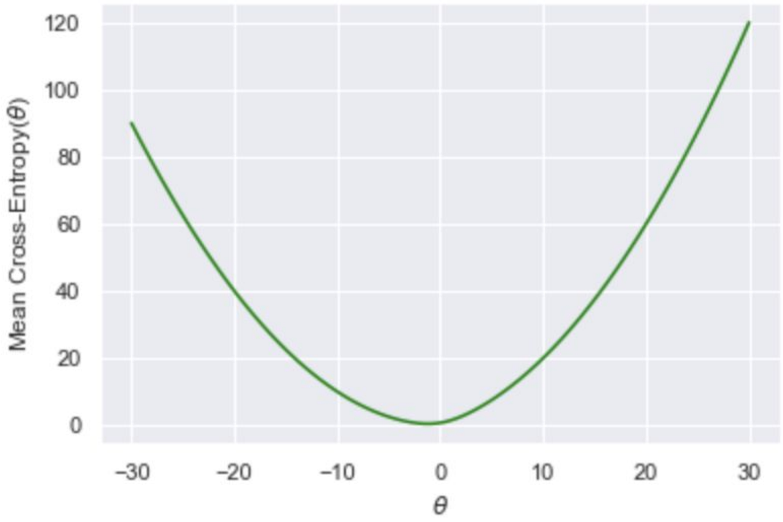
Regularized logistic regression

- 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)
```

Very little regularization!

Summary

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 (Extra)

Note

- We will not cover these slides in lecture.
- They are meant to serve as a reference for the lab assignment that covers multiclass classification in the context of decision trees.

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.

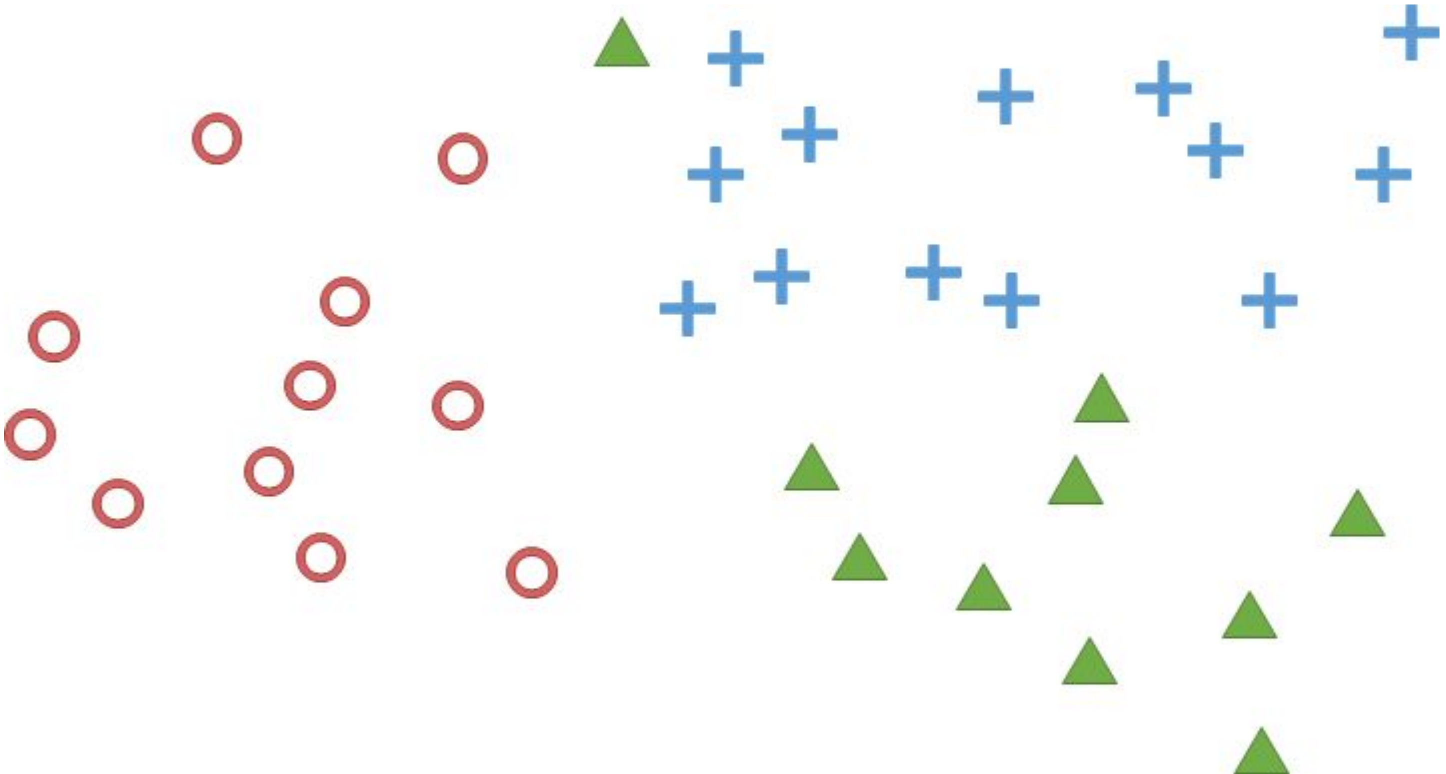
- Resulting prediction will just be whichever classifier gives highest probability.

Example from before:

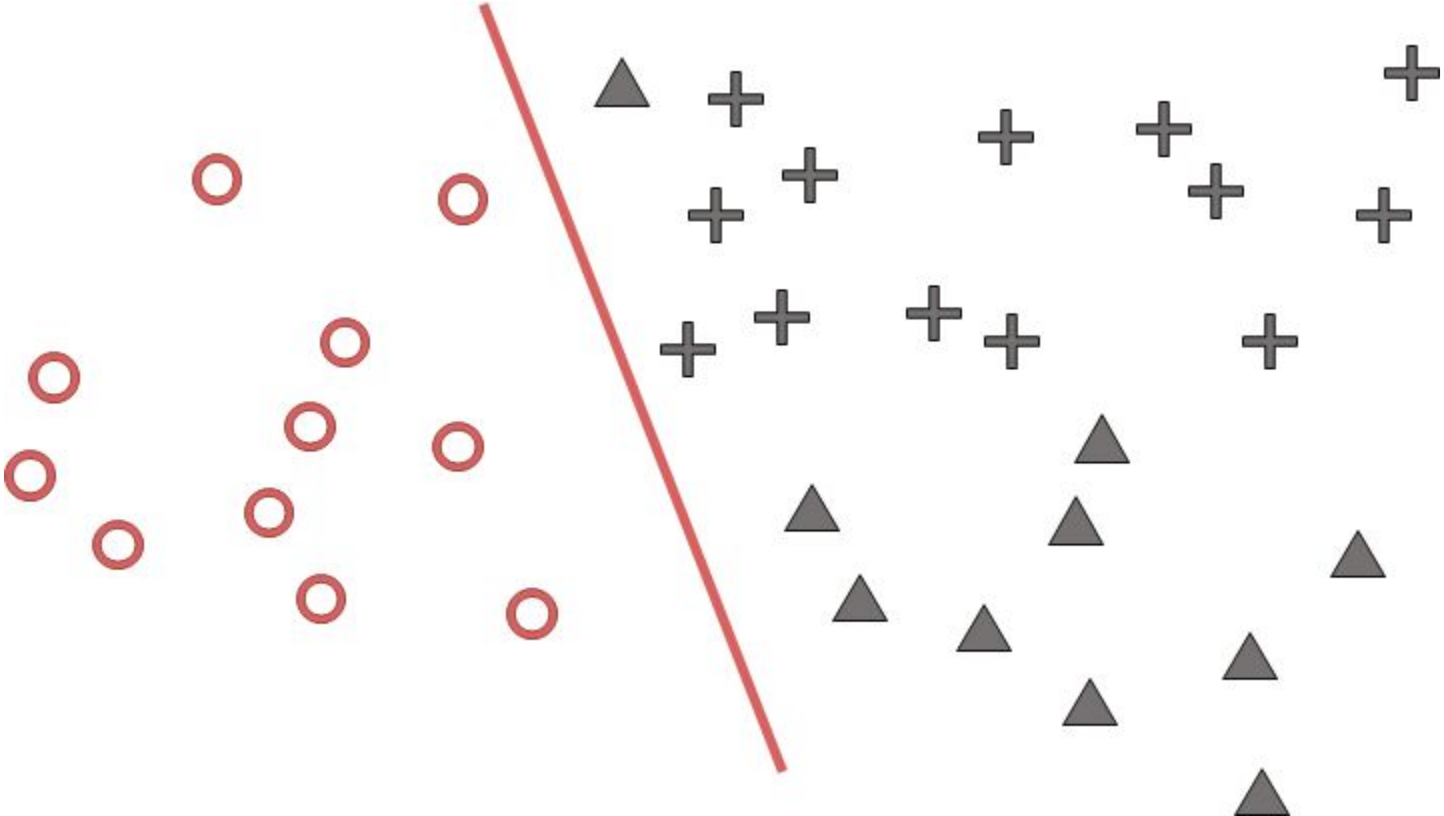
- Build a dog classifier.
- Build an cat classifier.
- Build a lion classifier.
- Build an zebra classifier…

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

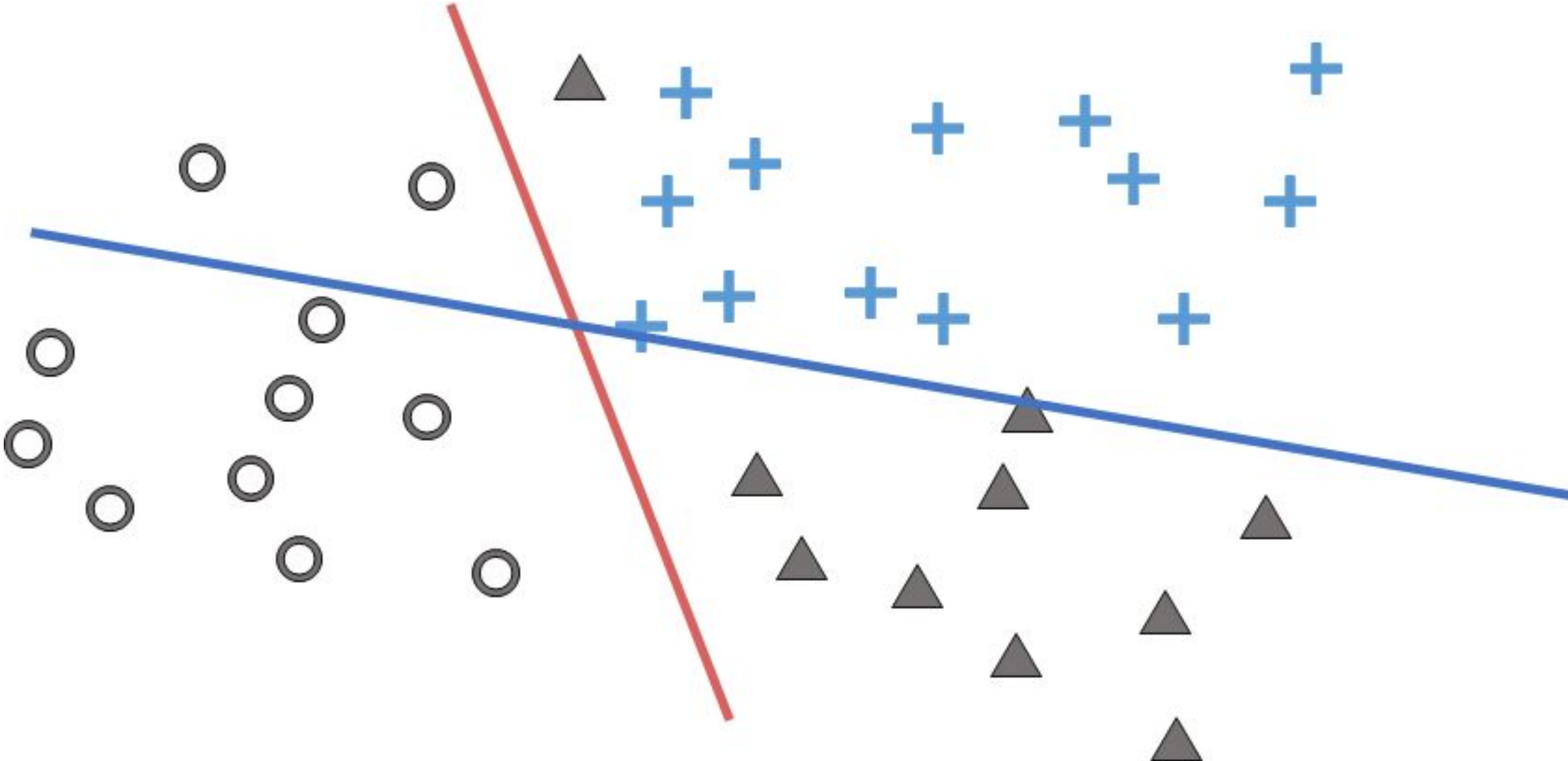
Visual example



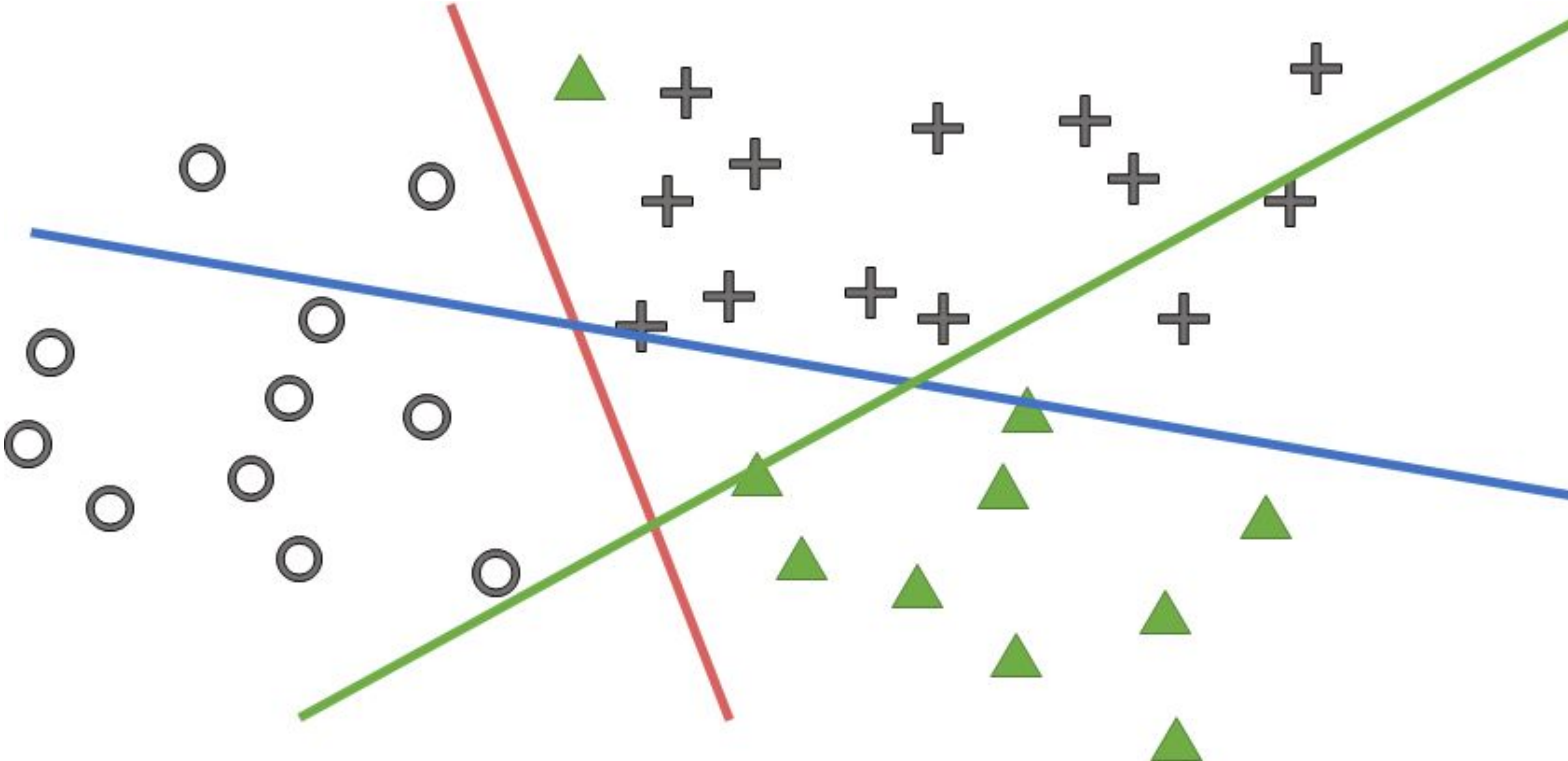
Visual example



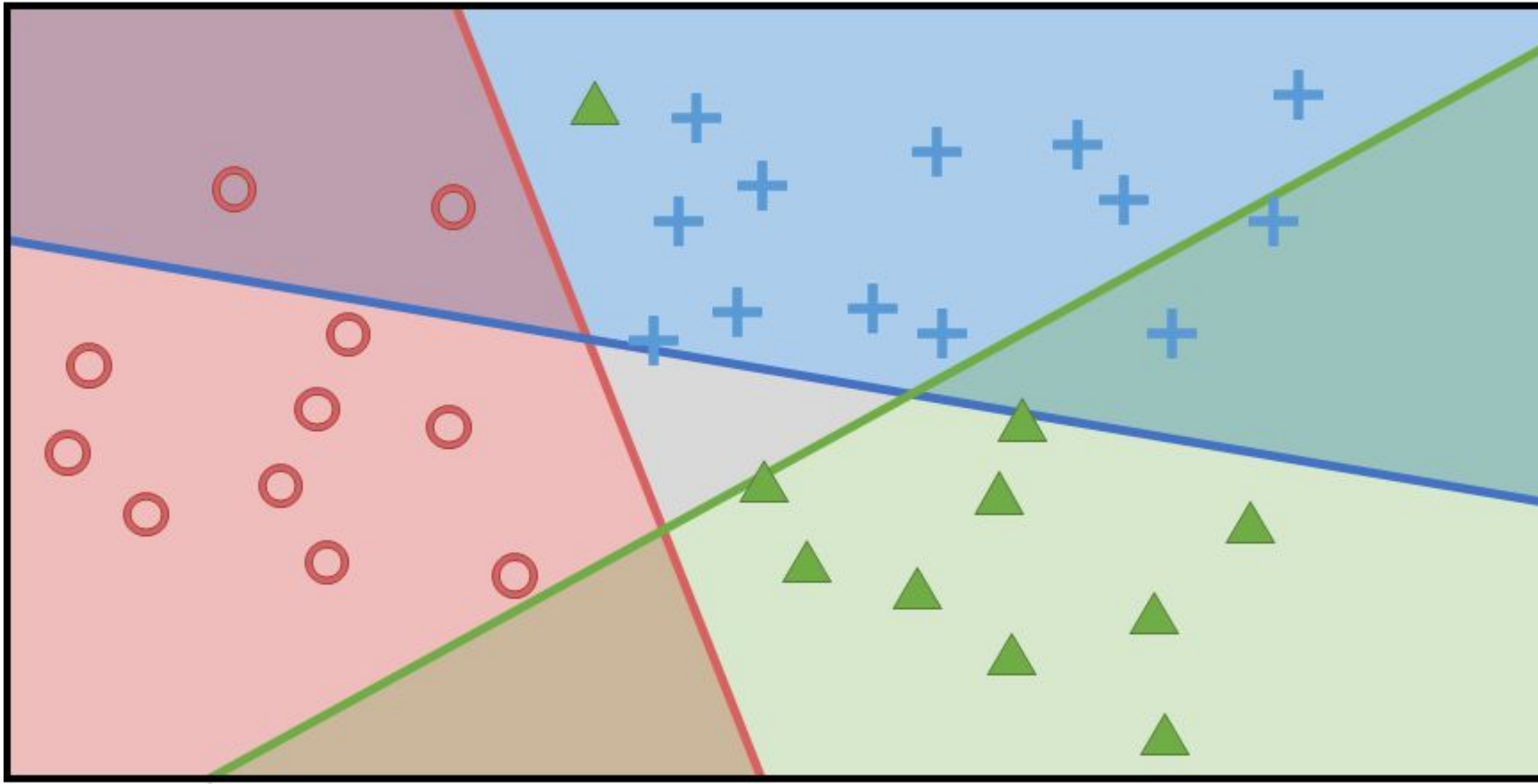
Visual example



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 $\theta^{(j)}$.
- Won't discuss in Data 100 . See CS188, CS189, CS182, Info 254, or Stat 151A.

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