LECTURE 22

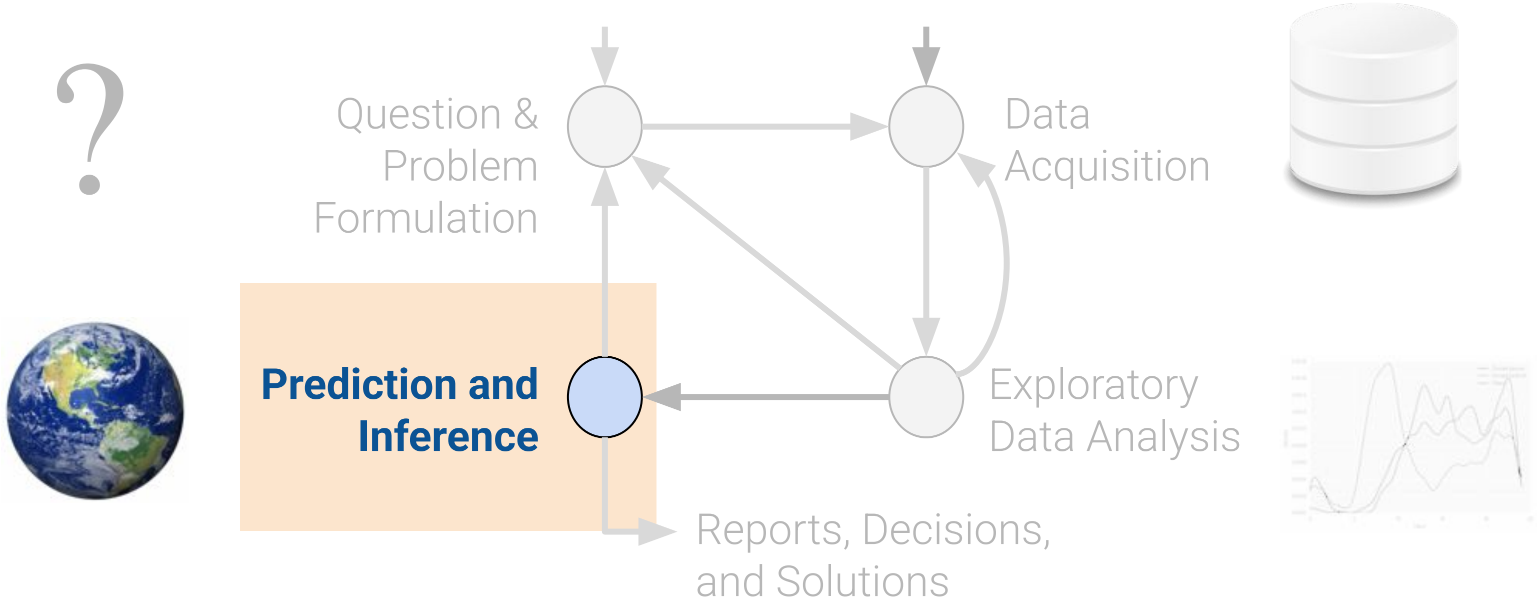
Logistic Regression II

Model Performance.

Data 100/Data 200, Spring 2022 @ UC Berkeley

Josh Hug and Lisa Yan

More Logistic Regression



Question & Problem Formulation

Data Acquisition

Prediction and Inference

Exploratory Data Analysis

Reports, Decisions, and Solutions

Logistic Regression I:
The Model
Cross-Entropy Loss
The Probabilistic View

(today)

Logistic Regression II:
Linear Separability
Accuracy, Precision, Recall
Classification Thresholds

Today's Roadmap

Lecture 22, Data 100 Spring 2022

Logistic Regression Model, continued

- sklearn demo
- Maximum Likelihood Estimation: high-level (live), detailed (recorded)

Linear separability and Regularization

Performance Metrics

- Accuracy
- Imbalanced Data, Precision, Recall

Adjusting the Classification Threshold

- A case study
- ROC curves, and AUC

[Extra] Detailed MLE, Gradient Descent, PR curves

Logistic Regression Model, continued

Lecture 22, Data 100 Spring 2022

Logistic Regression Model, continued

- sklearn demo
- Maximum Likelihood Estimation: high-level (live), detailed (recorded)

Linear separability and Regularization

Performance Metrics

- Accuracy
- Imbalanced Data, Precision, Recall

Adjusting the Classification Threshold

- A case study
- ROC curves, and AUC

[Extra] Detailed MLE, Gradient Descent, PR curves

Logistic Regression with sklearn

(documentation)

```python
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(fit_intercept=False)
model.fit(X, Y)
```

Task/Model

Binary Classification ($y \in \{0, 1\}$)

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

Fit to objective function

Average Cross-Entropy Loss

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

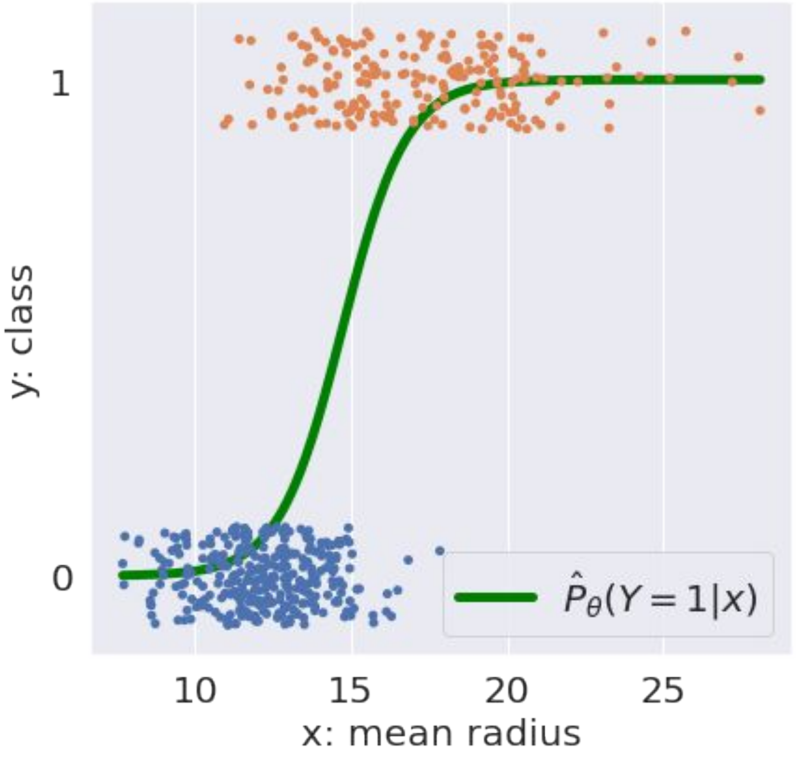
+ regularization

For logistic regression, sklearn applies regularization by default. We'll see why soon.

Demo

Sklearn: Predict Probabilities

```
model.predict_proba(X) # probs for all classes
model.classes_         # array([0, 1])
```



Demo

Demo

Sklearn: Classification

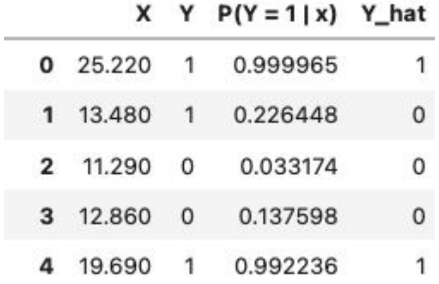
```
model.predict_proba(X) # probs for all classes
model.classes_         # array([0, 1])
```

```
model.predict(X)         # predict 1 or 0
```

$$\hat{y} = \text{classify}(x) = \begin{cases} 1 & \hat{P}_\theta(Y = 1|x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Equivalent "otherwise" condition: $\hat{P}_\theta(Y = 0|x) \geq 0.5$

Interpret: Given the input feature x:
If Y is more likely to be 1 than 0,
then predict $\hat{y} = 1$.
Else predict 0.



	X	Y	P(Y = 1 | x)	Y_hat
0	25.220	1	0.999965	1
1	13.480	1	0.226448	0
2	11.290	0	0.033174	0
3	12.860	0	0.137598	0
4	19.690	1	0.992236	1

[High-Level] Maximum Likelihood Estimation

This material will not be tested; I've recorded a detailed video instead (link).

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

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

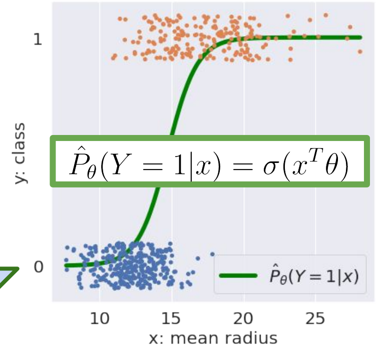
For logistic regression, let $p_i = \sigma(X_i^T\theta)$

Assumption: all data are independent Bernoulli random variables.

$$\underset{p_1, p_2, \ldots, p_n}{\textbf{argmax}} \quad \prod_{i=1}^{n} \underbrace{p_i^{y_i}(1-p_i)^{(1-y_i)}}_{\text{Prob. that i-th response is yi}}$$

Main takeaway: The optimal theta that minimizes mean cross-entropy loss "pushes" all probabilities in the direction of the true class.

Want $\sigma(x^T\theta) \to 0$



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

Want $\sigma(x^T\theta) \to 1$

[High-Level] Maximum Likelihood Estimation

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

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

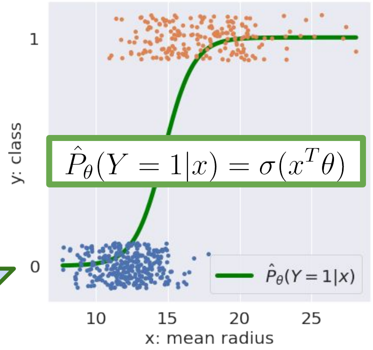
For logistic regression,
let $p_i = \sigma(X_i^T\theta)$

Assumption: all data are independent Bernoulli random variables

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

Prob. that i-th response is yi

Main takeaway: The optimal theta that minimizes mean cross-entropy loss "pushes" all probabilities in the direction of the true class.



Want

$\sigma(x^T\theta) \rightarrow 1$

Want

$\sigma(x^T\theta) \rightarrow 0$

Linear separability and Regularization

Lecture 22, Data 100 Spring 2022

Logistic Regression Model, continued

- sklearn demo
- Maximum Likelihood Estimation: high-level (live), detailed (recorded)

Linear separability and Regularization

Performance Metrics

- Accuracy
- Imbalanced Data, Precision, Recall

Adjusting the Classification Threshold

- A case study
- ROC curves, and AUC

[Extra] Detailed MLE, Gradient Descent, PR curves

Logistic Regression with sklearn

(documentation)

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(fit_intercept=False)
model.fit(X, Y)
```

Task/Model

Binary Classification ($y \in \{0, 1\}$)

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

Fit to objective function

Average Cross-Entropy Loss

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

+ **regularization**

Why does sklearn always apply regularization?

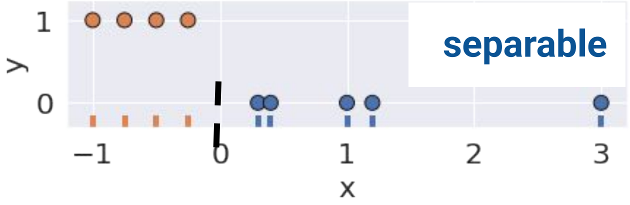
Demo

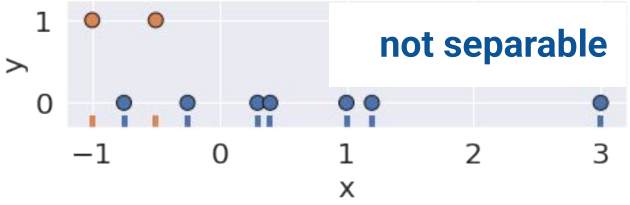
Linear Separability

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; the input feature is **1-D**.

- Class label is not a feature; it is output.
- Use rug plot to see separability.



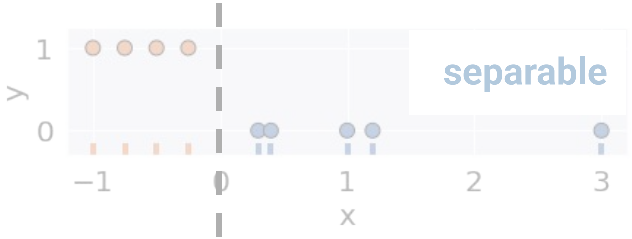


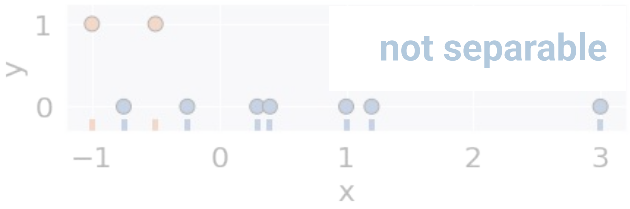
Linear Separability

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

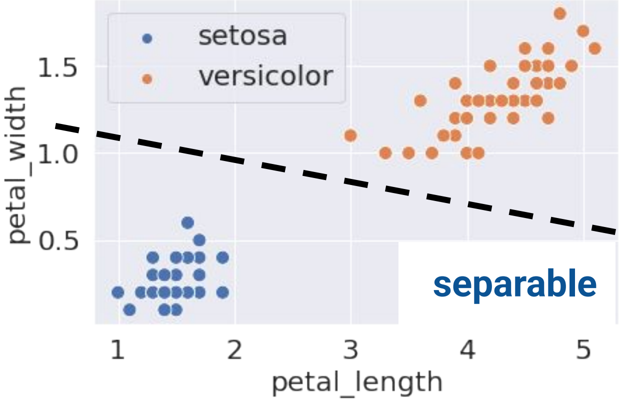
If there is one feature; the input feature is **1-D**.

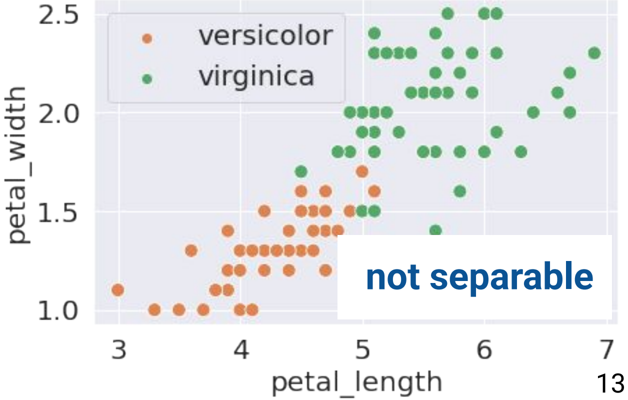
- Class label is not a feature; it is output.
- Use rug plot to see separability.





If there are two features, the input feature is **2-D**. Use scatter plot to see separability.





Linearly Separability Creates Diverging Weights

Consider the simplified logistic regression model fit to the toy data:

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

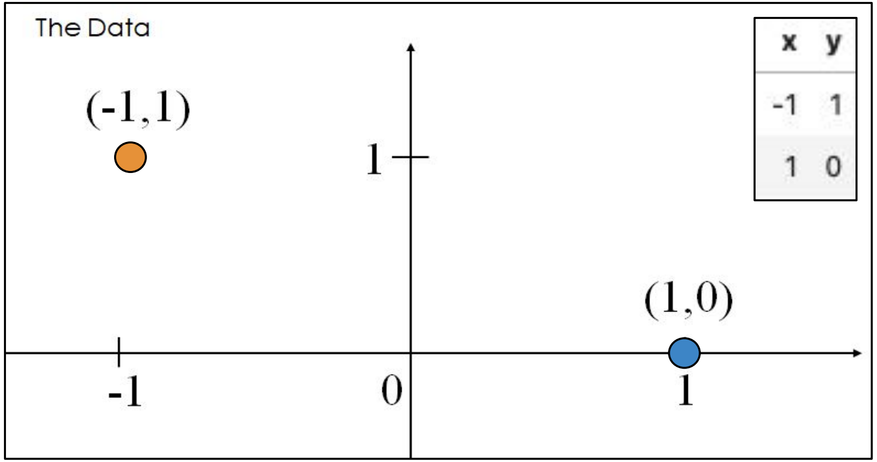
What will be the optimal weight theta? Why?

A. $\hat{\theta} = -1$

B. $\hat{\theta} = 1$

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

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



x	y
-1	1
1	0

[Hint] The optimal theta should "push" probabilities in the direction of the true class:

● $\hat{P}_\theta(Y = 1|x = -1) = \dfrac{1}{1 + e^{\theta}} \quad \to 1$

● $\hat{P}_\theta(Y = 1|x = 1) = \dfrac{1}{1 + e^{-\theta}} \quad \to 0$

Linearly Separability Creates Diverging Weights

Consider the simplified logistic regression model fit to the toy data:

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

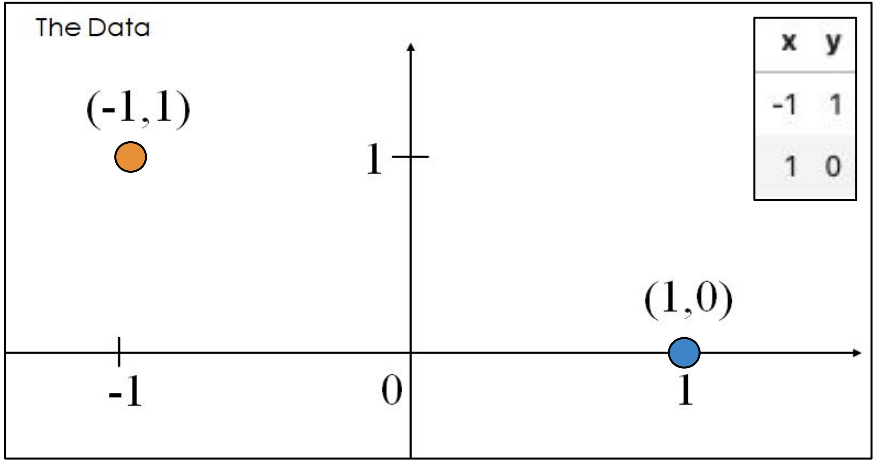
What will be the optimal weight theta? Why?

A. $\hat{\theta} = -1$

B. $\hat{\theta} = 1$

[✓ C.] $\hat{\theta} \to -\infty$

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



x	y
-1	1
1	0

[Hint] The optimal theta should "push" probabilities in the direction of the true class:

● $$\hat{P}_\theta(Y = 1|x = -1) = \frac{1}{1 + e^{\theta}} \to 1$$

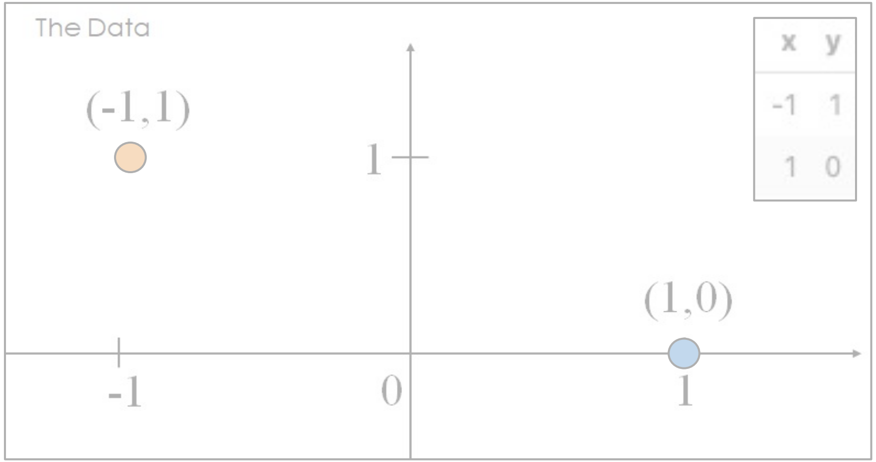
● $$\hat{P}_\theta(Y = 1|x = 1) = \frac{1}{1 + e^{-\theta}} \to 0$$

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

Linearly Separability Creates Diverging Weights

Consider the simplified logistic regression model fit to the toy data:

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



x	y
-1	1
1	0

What will be the optimal weight theta? Why?

A. $\hat{\theta} = -1$

B. $\hat{\theta} = 1$

[✓ C.] $\hat{\theta} \rightarrow -\infty$

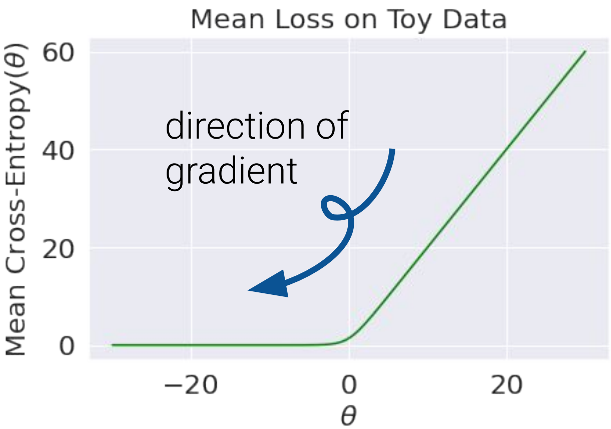
D. $\hat{\theta} \rightarrow \infty$

[Hint] The optimal theta should "push" probabilities in the direction of the true class:

● $\hat{P}_\theta(Y = 1|x = -1) = \dfrac{1}{1 + e^{\theta}}$ $\rightarrow 1$

● $\hat{P}_\theta(Y = 1|x = 1) = \dfrac{1}{1 + e^{-\theta}}$ $\rightarrow 0$

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



direction of gradient

(Impossible to see, but) plateau is slightly tilted downwards.

Loss approaches 0 as theta decreases.

Linearly Separability Creates Diverging Weights

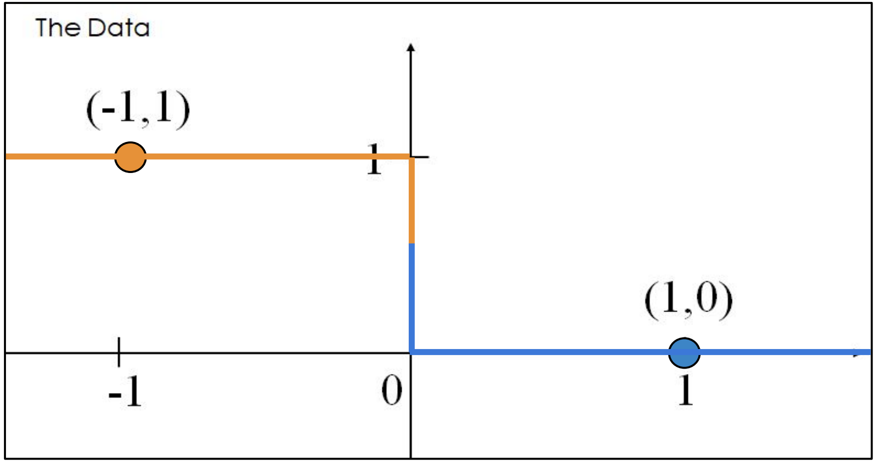
Consider the simplified logistic regression model fit to the toy data:

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

$$\hat{P}_\theta(Y = 1|x = -1) = \frac{1}{1 + e^{\theta}} \to 1$$

$$\hat{P}_\theta(Y = 1|x = 1) = \frac{1}{1 + e^{-\theta}} \to 0$$

$$\hat{P}_\theta(Y = 1|x) = \sigma(\theta x) \to \begin{cases} 1 & \text{if } x < 0 \\ 0 & \text{if } x \geq 0 \end{cases}$$



Linearly Separability Creates Diverging Weights

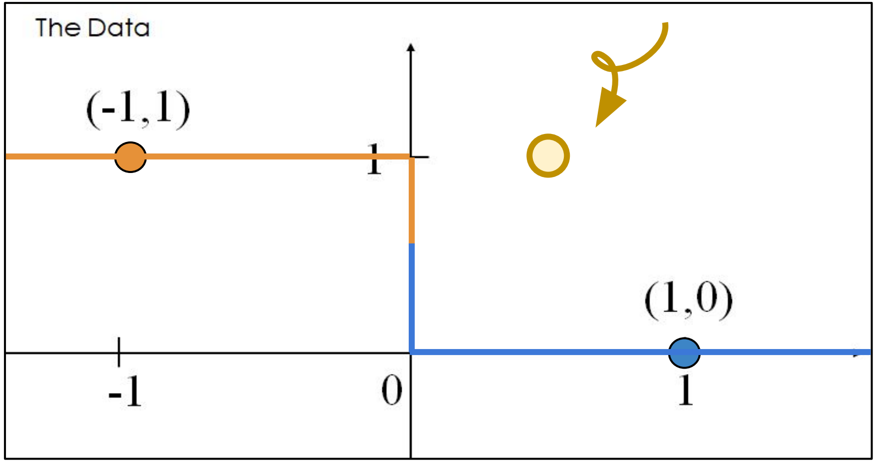
Consider the simplified logistic regression model fit to the toy data:

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

$$\hat{P}_\theta(Y = 1|x = -1) = \frac{1}{1 + e^{\theta}} \to 1$$

$$\hat{P}_\theta(Y = 1|x = 1) = \frac{1}{1 + e^{-\theta}} \to 0$$

$$\hat{P}_\theta(Y = 1|x) = \sigma(\theta x) \to \begin{cases} 1 & \text{if } x < 0 \\ 0 & \text{if } x \geq 0 \end{cases}$$



This model is **overconfident**.

- Consider a new point (0.5, **1**).
- The model incorrectly says **p = 0**, so it predicts **0**.

Typo fixed 5/3

$$\overset{=}{-}(\overset{1}{y}\log(p) + (1 - y)\log(1 - p))$$

$$\to 1\log(0)$$

Loss is infinite.

Divergent weights (i.e., | θ | →∞) occur with **linearly separable** data.

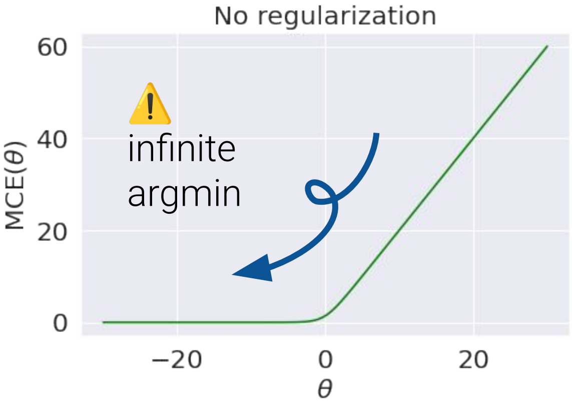
“Overconfidence” is a particularly dangerous version of overfitting.

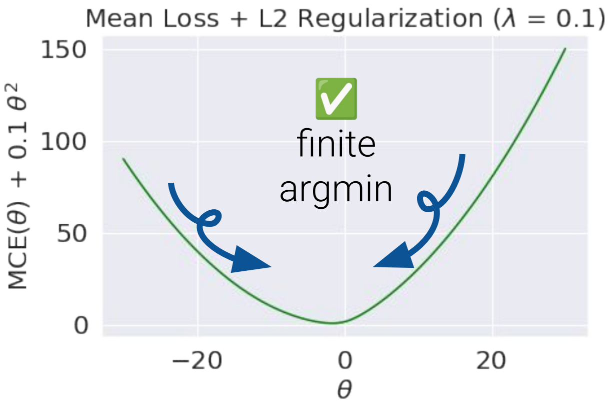
Regularized Logistic Regression

To avoid large weights (particularly on linearly separable data), use **regularization**.

- As with linear regression, standardize features first.

$$\underset{\theta}{\text{argmin}} - \frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\sigma(X_i^T\theta) + (1-y_i)\log(1-\sigma(X_i^T\theta))\right) + \lambda\sum_{j=1}^{d}\theta_j^2$$





```
# sklearn defaults
model = LogisticRegression(
          penalty='l2', C=1.0, …)
model.fit()
```

Regularization hyperparameter C is the inverse of λ. C = 1 / λ.

Set C big for minimal regularization, e.g., `C=300.0`.

Performance Metrics

Lecture 22, Data 100 Spring 2022

Logistic Regression Model, continued

- sklearn demo
- Maximum Likelihood Estimation: high-level (live), detailed (recorded)

Linear separability and Regularization

Performance Metrics

- Accuracy
- Imbalanced Data, Precision, Recall

Adjusting the Classification Threshold

- A case study
- ROC curves, and AUC

[Extra] Detailed MLE, Gradient Descent, PR curves

Next Time

1. Choose a model ✅

2. Choose a loss function ✅

3. Fit the model

4. Evaluate model performance

Regression ($y \in \mathbb{R}$)

Linear Regression

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

Squared Loss or Absolute Loss

Regularization

Sklearn/Gradient descent

R^2, Residuals, etc.

Classification ($y \in \{0, 1\}$)

Logistic Regression

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

Average Cross-Entropy Loss

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

Regularization

Sklearn/~~Gradient descent~~

Let's do it!

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) # 0.8691
```

```
model.score(X, Y) # 0.8691
```

(sklearn documentation)

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

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).

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

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 Successes/Errors: The Confusion Matrix

- **True positives** and **true negatives** are when we correctly classify an observation as being positive or negative, respectively.

- **False positives** are “false alarms”: we predicted 1, but the true class was 0.
- **False negatives** are “failed detections”: we predicted 0, but the true class was 1.

Prediction $\hat{y}$

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

“**positive**” means a prediction of **1**.
“**negative**” means a prediction of **0**.

Types of Classification Successes/Errors: The Confusion Matrix

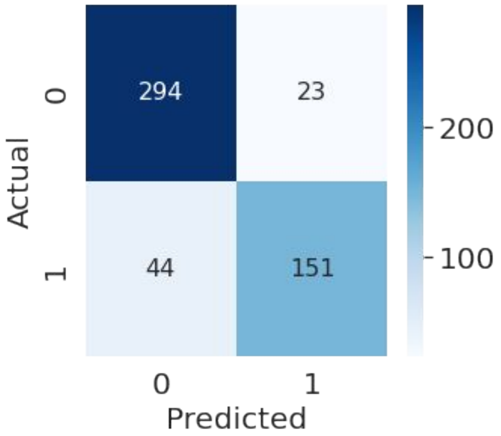
- **True positives** and **true negatives** are when we correctly classify an observation as being positive or negative, respectively.

- **False positives** are “false alarms”: we predicted 1, but the true class was 0.
- **False negatives** are “failed detections”: we predicted 0, but the true class was 1.

		Prediction $\hat{y}$	
		0	**1**
Actual y	0	True **negative** (TN)	False **positive** (FP)
	1	False **negative** (FN)	True **positive** (TP)

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

```python
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	
		0	1
Actual	0	TN	FP
	1	FN	TP

Accuracy, 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?

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.

Accuracy, 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?

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.

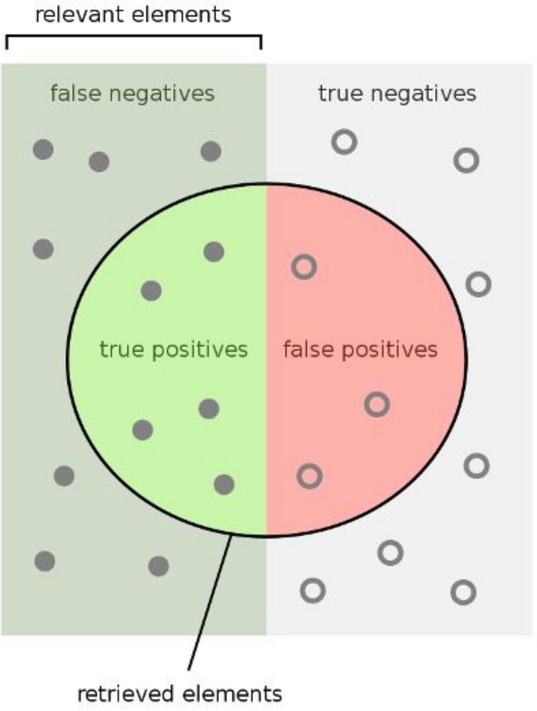
$$\text{recall} = \frac{TP}{TP + FN}$$

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

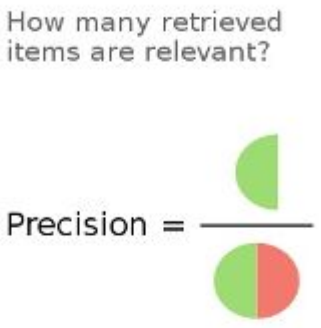
- How **sensitive** is our classifier to **positives**?
- Penalizes false negatives.

One of the Most Valuable Graphics on Wikipedia

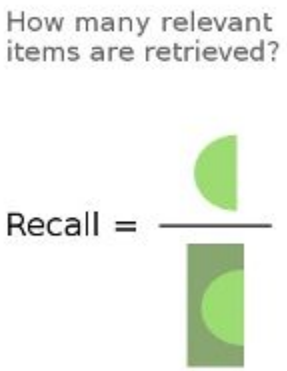
(i.e., positive;
predicted class is 1)



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



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



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

[adapted from Wikipedia]

Back to the Spam

$$\begin{aligned} \text{accuracy} &= \frac{TP + TN}{n} \\ \text{precision} &= \frac{TP}{TP + FP} \\ \text{recall} &= \frac{TP}{TP + FN} \end{aligned}$$

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).

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

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

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

Never positive!

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

$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

Precision penalizes false positives, and Recall penalizes false negatives.

We can achieve **100% recall** by making our classifier output “1”, regardless of the input.

- Friend 2’s “always predict spam” classifier.
- 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 often inversely related.

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

(see extra slides re: the precision-recall curve)

Which Performance Metric?

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

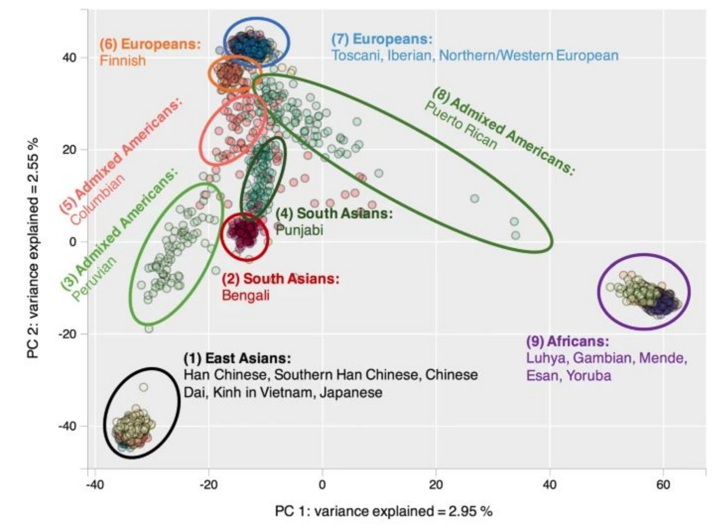
$$\text{precision} = \frac{TP}{TP + FP}$$

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

In many settings, there might be a much higher cost to missing positive cases.
For our tumor classifier:

- We really don't want to miss any malignant tumors (avoid false negatives).
- We might be fine with classifying benign tumors as malignant (OK to have false positives), since pathologists could do further studies to verify all malignant tumors.
- This context would prioritize **recall**.

How do we engineer classifiers to meet the performance goals of our problem?

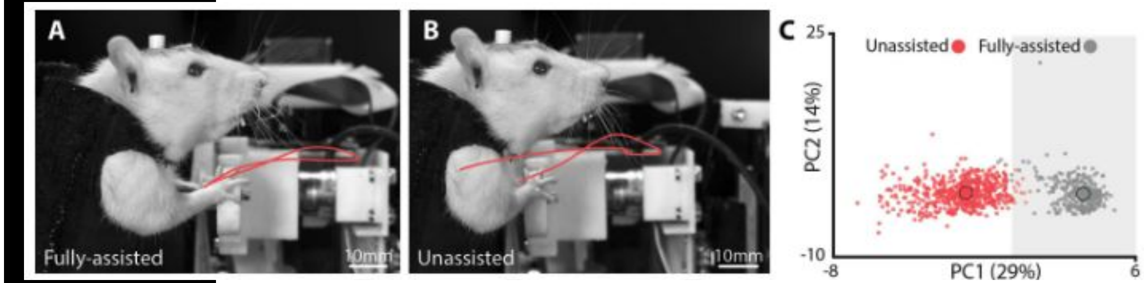


PCA clustering Principal Component Analysis (PCA) plot of 20 populations from 1000 Genomes Project, built using 2 first principal components. The following populations were not used to build the

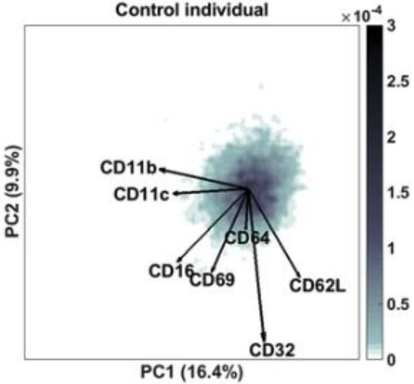
Interlude

PCA is commonly used in biomedical contexts, which have many named variables!

1. To cluster data (Paper 1, Paper 2)



2. To identify correlated variables (interpret rows of V^T as linear coefficients) (Paper 3). Uses biplots.



Break (2 min)

Adjusting the Classification Threshold

Lecture 22, Data 100 Spring 2022

Logistic Regression Model, continued

- sklearn demo
- Maximum Likelihood Estimation: high-level (live), detailed (recorded)

Linear separability and Regularization

Performance Metrics

- Accuracy
- Imbalanced Data, Precision, Recall

Adjusting the Classification Threshold

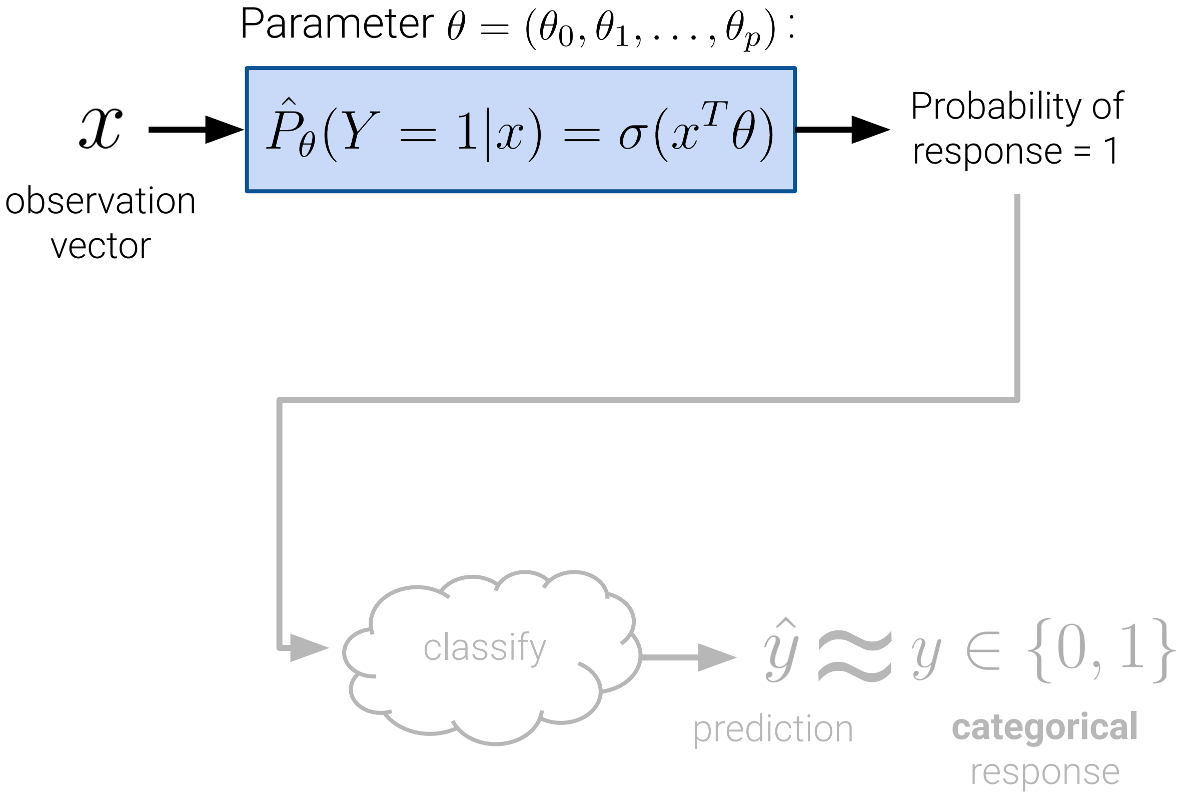
- A case study
- ROC curves, and AUC

[Extra] Detailed MLE, Gradient Descent, PR curves

Engineering, Part 1: Deciding a Model

Feature Engineering:

What are the features x that generate great probabilities for prediction?



Parameter $\theta = (\theta_0, \theta_1, \ldots, \theta_p)$:

x

observation vector

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

Probability of response = 1

classify

$\hat{y} \approx y \in \{0, 1\}$

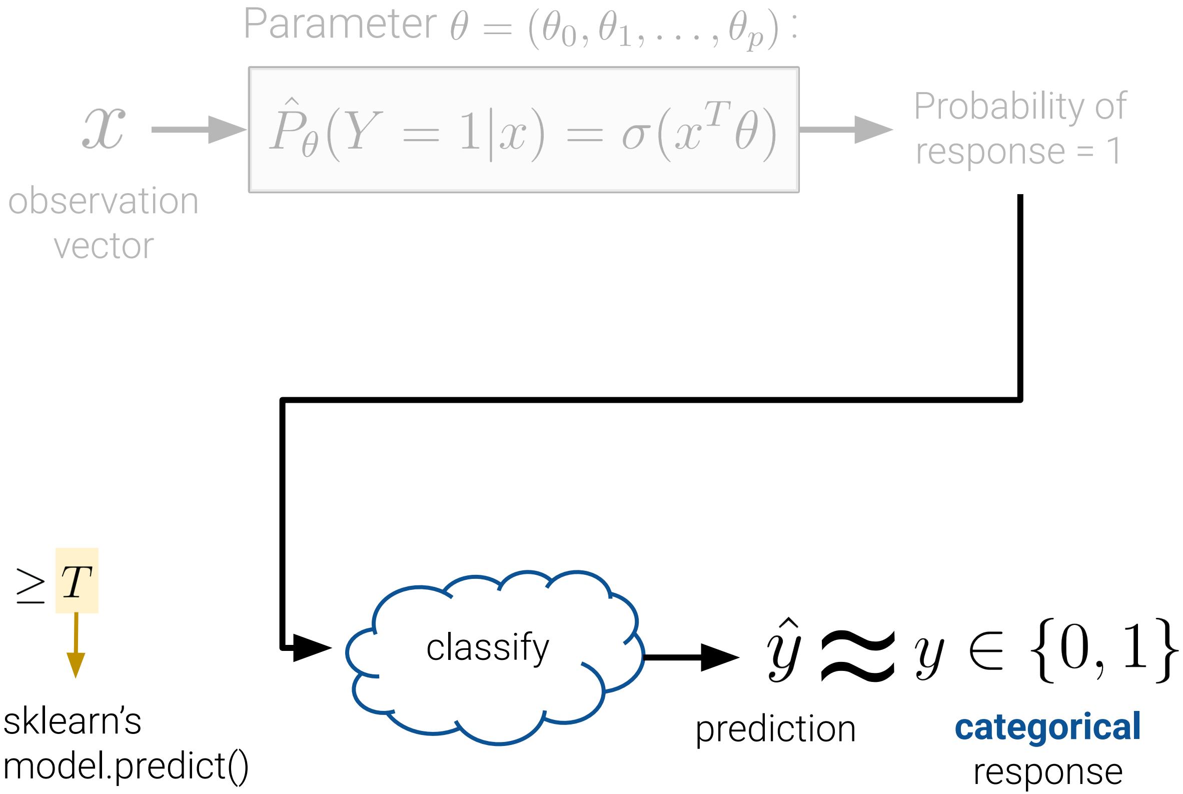
prediction

categorical response

Engineering, Part 2: Deciding a Classification Threshold

Classification:

What is the best **classification threshold T** to choose that best fits our problem context?



Parameter $\theta = (\theta_0, \theta_1, \ldots, \theta_p)$:
x
observation vector
$\hat{P}_\theta(Y = 1|x) = \sigma(x^T\theta)$
Probability of response = 1
classify
$\hat{y} \approx y \in \{0, 1\}$
prediction
categorical response

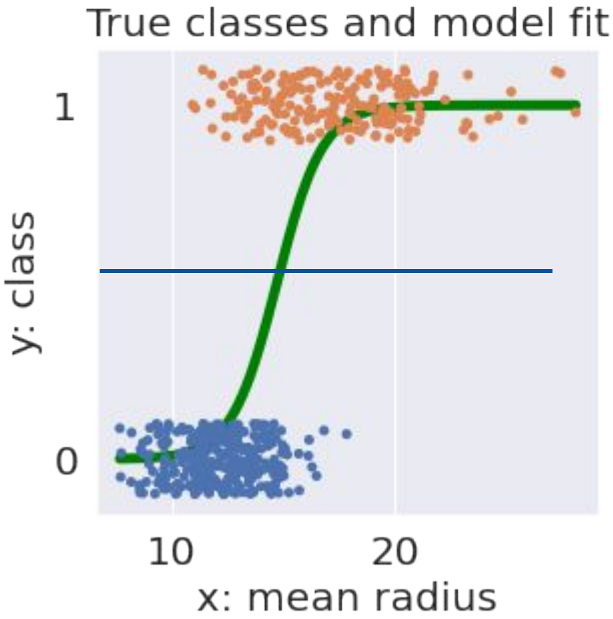
$$\hat{y} = \text{classify}(x) = \begin{cases} 1 & \hat{P}_\theta(Y = 1|x) \geq T \\ 0 & \text{otherwise} \end{cases}$$

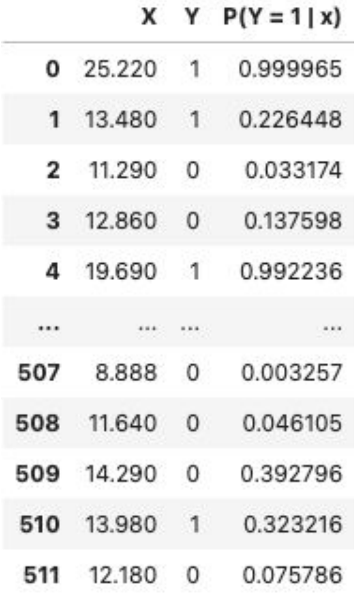
sklearn's model.predict() uses fixed 0.5

Classification Threshold

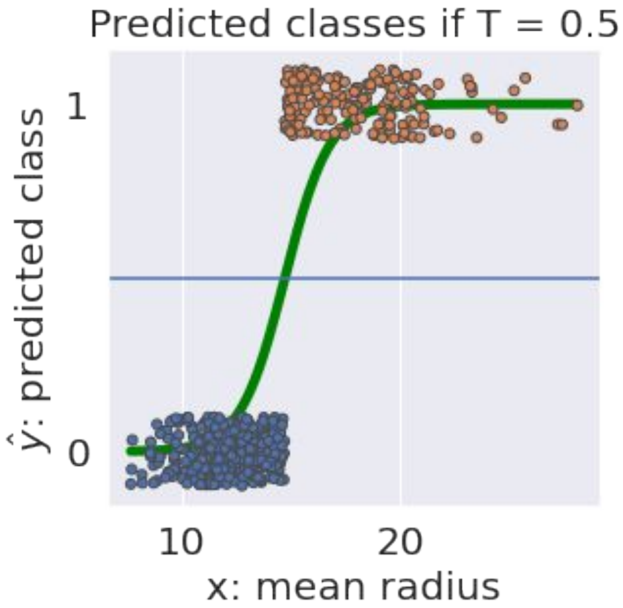
$$\hat{y} = \text{classify}(x) = \begin{cases} 1 & \hat{P}_\theta(Y = 1|x) \geq T \\ 0 & \text{otherwise} \end{cases}$$

The default threshold in sklearn is T = 0.5.


True classes and model fit



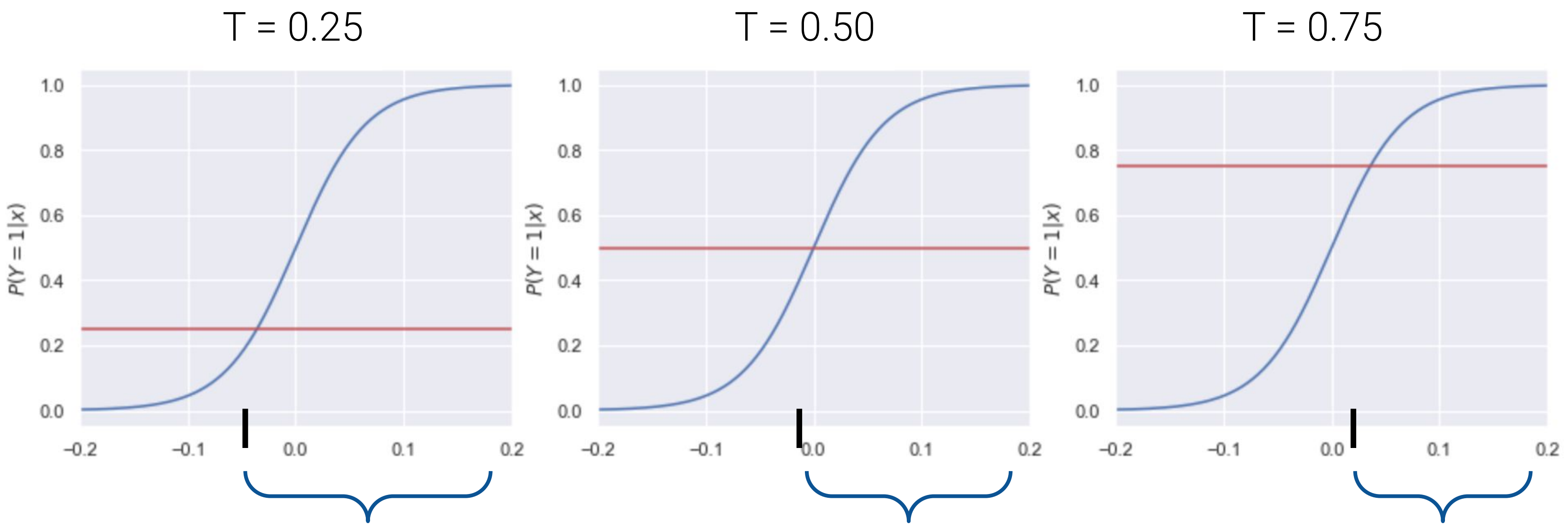
	X	Y	P(Y = 1 | x)
0	25.220	1	0.999965
1	13.480	1	0.226448
2	11.290	0	0.033174
3	12.860	0	0.137598
4	19.690	1	0.992236
...	...	...	...
507	8.888	0	0.003257
508	11.640	0	0.046105
509	14.290	0	0.392796
510	13.980	1	0.323216
511	12.180	0	0.075786


Predicted classes if T = 0.5

Classification Threshold

$$\hat{y} = \text{classify}(x) = \begin{cases} 1 & \hat{P}_\theta(Y = 1|x) \geq T \\ 0 & \text{otherwise} \end{cases}$$

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

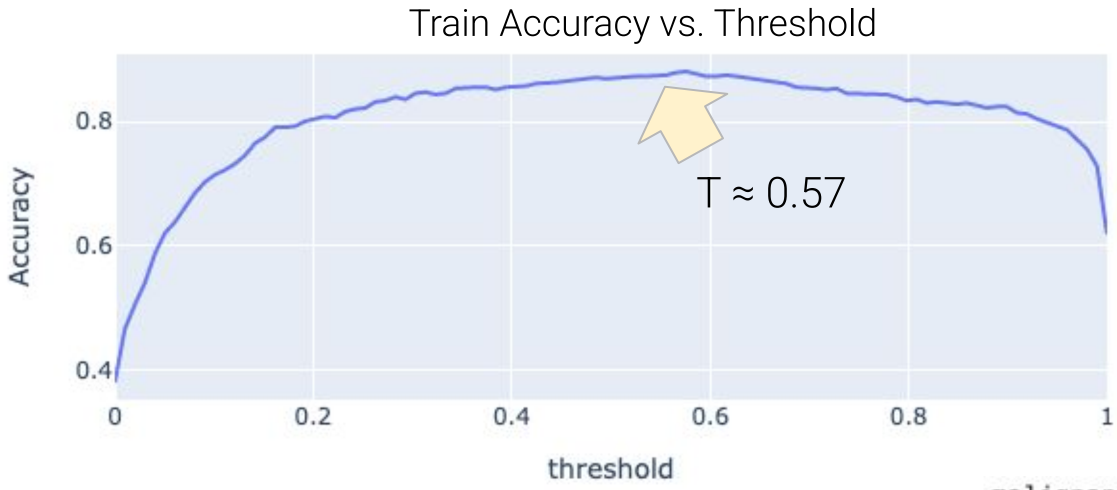
Fewer positives

Choosing an Accuracy Threshold

The choice of threshold T impacts our classification performance.

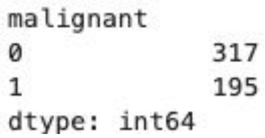
- High T: Most predictions are 0. Lots of false negatives.
- Low T: Most predictions are 1. Lots of false positives.

Do we get max accuracy when T ≈ 0.5? Not always the case…



T ≈ 0.57

Best T ≈ 0.57 likely due to class imbalance. There are fewer malgnant tumors and so we want to be more confident before classifying a tumor as malignant.



```
malignant
0          317
1          195
dtype: int64
```

Demo

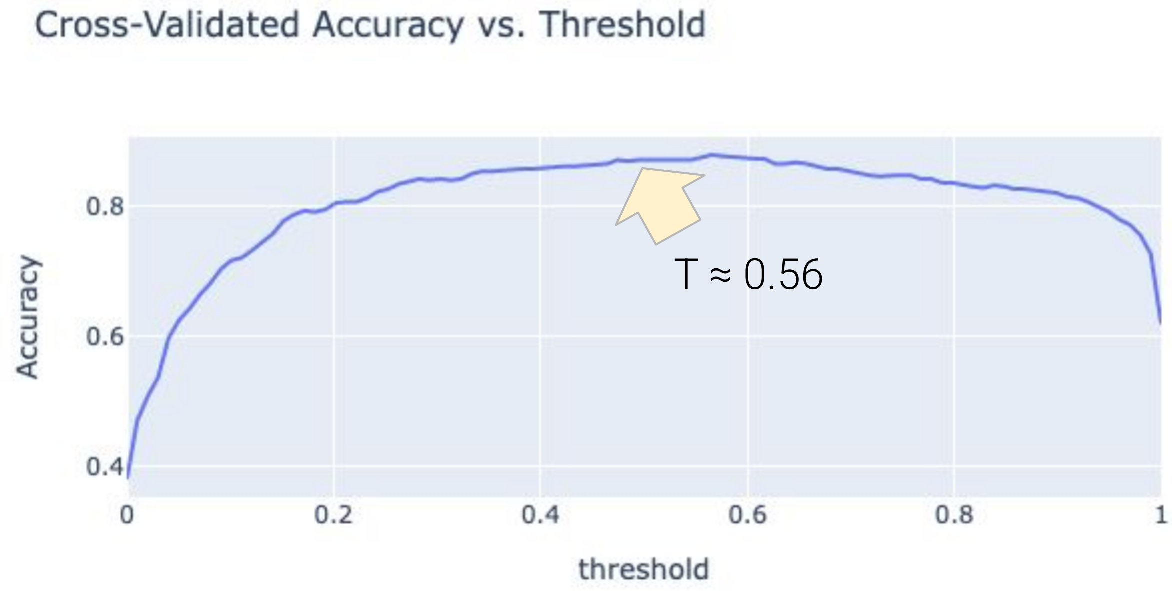
See notebook for code snippets.

Tune Thresholds with Cross Validation

The threshold should typically be tuned using cross validation.

For a threshold T:

cross_val_acc = (1/k) [Model fit to train set 1, Acc on val set 1 + … + Model fit to train set k, Acc on val set k]



T ≈ 0.56

Demo

See notebook for code snippets.

[documentation]

Demo

Choosing a Threshold According to Other Metrics?

The choice of threshold T impacts our classification performance.

- High T: Most predictions are 0. Lots of false negatives.
- Low T: Most predictions are 1. Lots of false positives.

Could we choose a threshold T based on metrics that measure false positives/false negatives?

Yes! Two options:

- Precision-Recall Curve (PR Curve). Covered in extra slides.
- “Receiver Operating Characteristic” Curve (**ROC Curve**).

Each of these visualizations have an associated performance metric: **AUC (Area Under Curve)**.

Demo

Two More Metrics

Prediction

	0	1
0	TN	FP
1	FN	TP

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

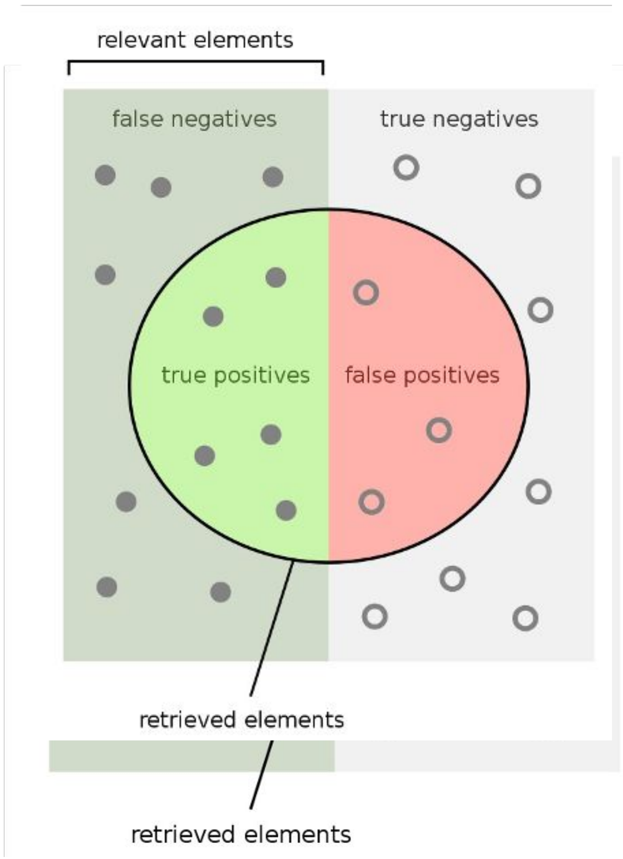
True Positive Rate (TPR):
“What proportion of spam did I mark correctly?
Same thing as **recall**. In statistics, sensitivity.

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

False Positive Rate (FPR):
“What proportion of regular email did I mark as spam?
In statistics, also called specificity.

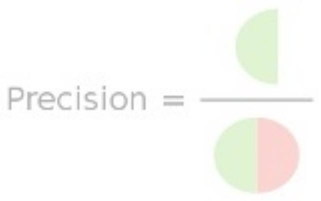
The ROC curve plots TPR vs FPR for different classification thresholds.

One of the Most Valuable Graphics on Wikipedia, Now With FPR



[adapted from Wikipedia]

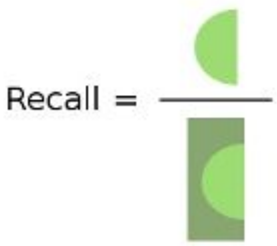
How many retrieved items are relevant?



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

Want close to 1.0

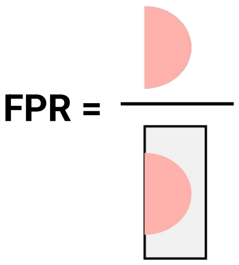
How many relevant items are retrieved?



$$\underset{\textbf{TPR}}{\text{recall}} = \frac{TP}{TP+FN}$$

Want close to 1.0

How many irrelevant items are retrieved?



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

Want close to 0.0

The ROC curve plots TPR vs FPR for different classification thresholds.

Demo

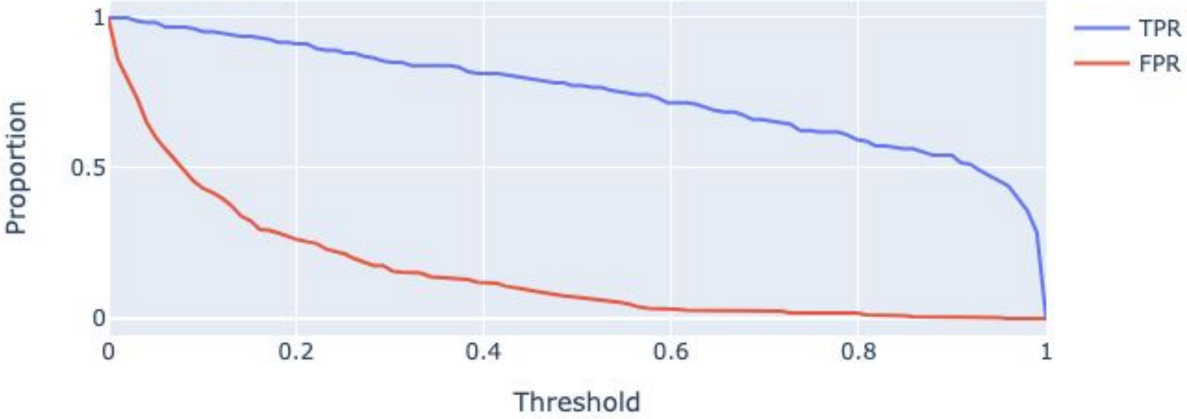
Not the ROC Curve, but Useful to Start with

The choice of threshold T impacts our classification performance.

- High T: Most predictions are 0. Lots of false negatives.
- Low T: Most predictions are 1. Lots of false positives.

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

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



As we increase T, **both** **TPR** and FPR decrease.

- A decreased **TPR** is bad (detecting fewer positives).
- A decreased **FPR** is good (fewer false positives).

Demo

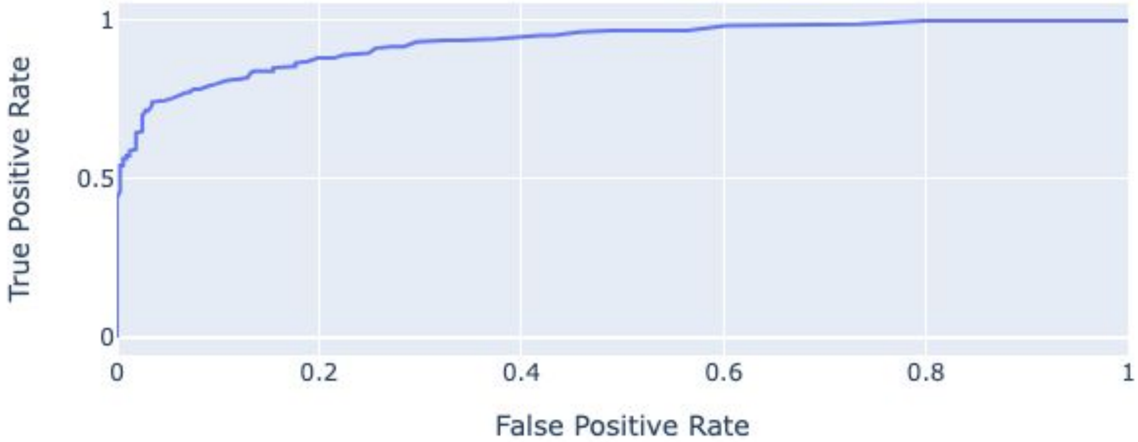
The ROC Curve

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

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

The ROC Curve plots this tradeoff.

- ROC stands for “Receiver Operating Characteristic.” [Wikipedia]
- We want high TPR, low FPR.



1. Which part of this curve corresponds to T = 0.9?
2. Which part of this curve corresponds to T = 0.1?

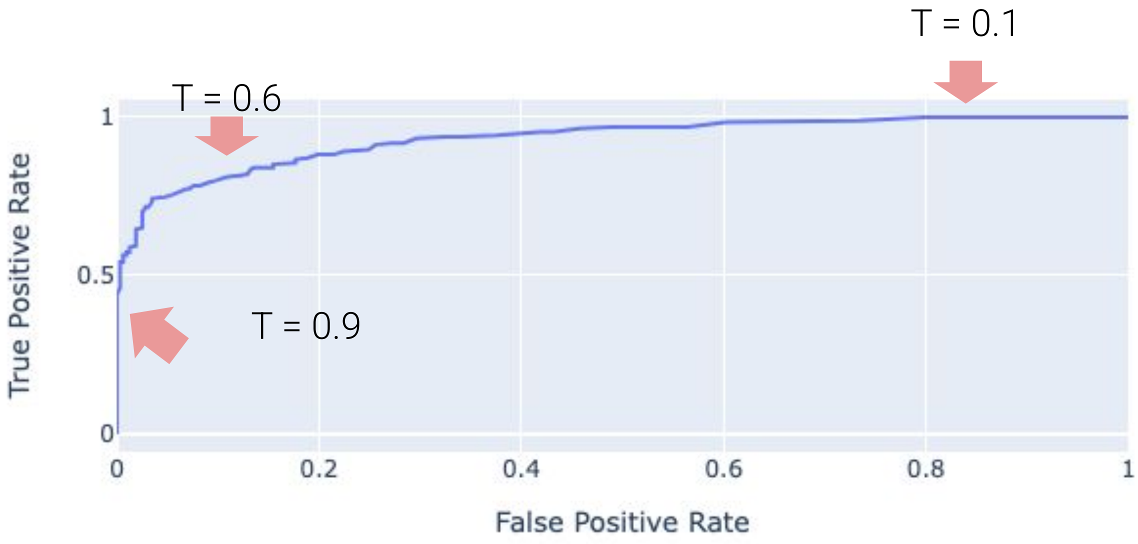
The ROC Curve

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

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

The ROC Curve plots this tradeoff.

- ROC stands for “Receiver Operating Characteristic.” [Wikipedia]
- We want high TPR, low FPR.



T = 0.1

T = 0.6

T = 0.9

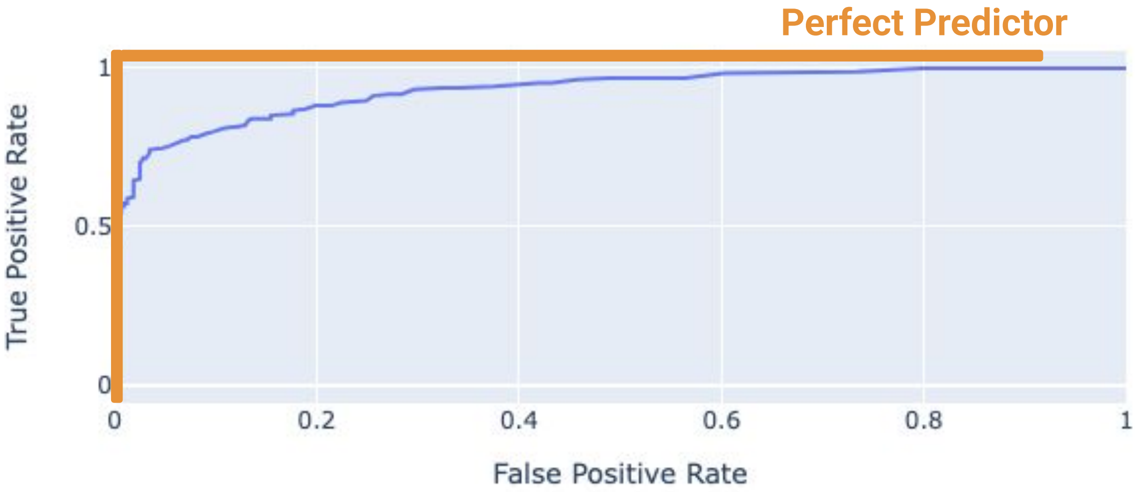
Demo

Demo

The Perfect Classifier

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.



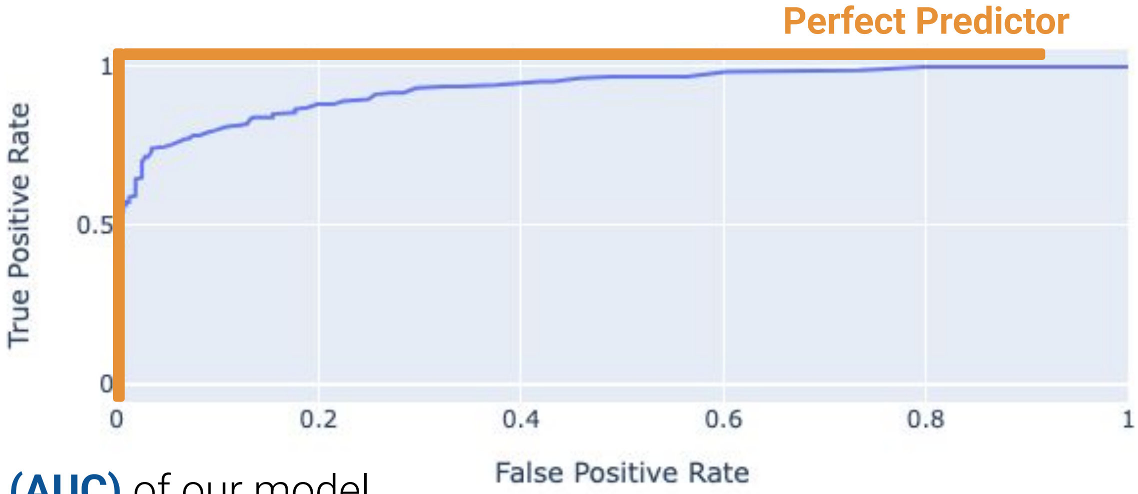
Performance Metric: Area Under Curve (AUC)

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

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

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

- We want our 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.

- 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.

[Extra] What is the “worst” AUC and why is it 0.5?

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

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

This slide was added post-lecture to clarify closing comments.

Best possible AUC = 1. Terrible AUC = 0.5.

- Random predictors have an AUC of around 0.5. Why?

A **random predictor** randomly predicts P(Y = 1 | x) to be uniformly between 0 and 1.

On average, if your dataset is size n_1+n_0 (with n_1 true class 1’s and n_0 true class 0’s):

If T = 0.5:

	0	1
0	TN = 0.5 n_0	FP = 0.5 n_0
1	FN = 0.5 n_1	TP = 0.5 n_1

FPR = 0.5 n_0/((0.5 + 0.5)n_0) = 0.5

TPR = 0.5 n_1/((0.5 + 0.5)n_1) = 0.5

Point on ROC curve is (0.5, 0.5).

If T = 0.8:

	0	1
0	TN = 0.8 n_0	FP = 0.2 n_0
1	FN = 0.8 n_1	TP = 0.2 n_1

FPR = 0.2 n_0/((0.2 + 0.8)n_0) = 0.2

TPR = 0.2 n_1/((0.2 + 0.8)n_1) = 0.2

Point on ROC curve is (0.2, 0.2).

If T = 0.3:

	0	1
0	TN = 0.3 n_0	FP = 0.7 n_0
1	FN = 0.3 n_1	TP = 0.7 n_1

FPR = 0.7 n_0/((0.7 + 0.3)n_0) = 0.7

TPR = 0.7 n_1/((0.7 + 0.3)n_1) = 0.7

Point on ROC curve is (0.7, 0.7).

[Extra] What is the “worst” AUC and why is it 0.5?

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

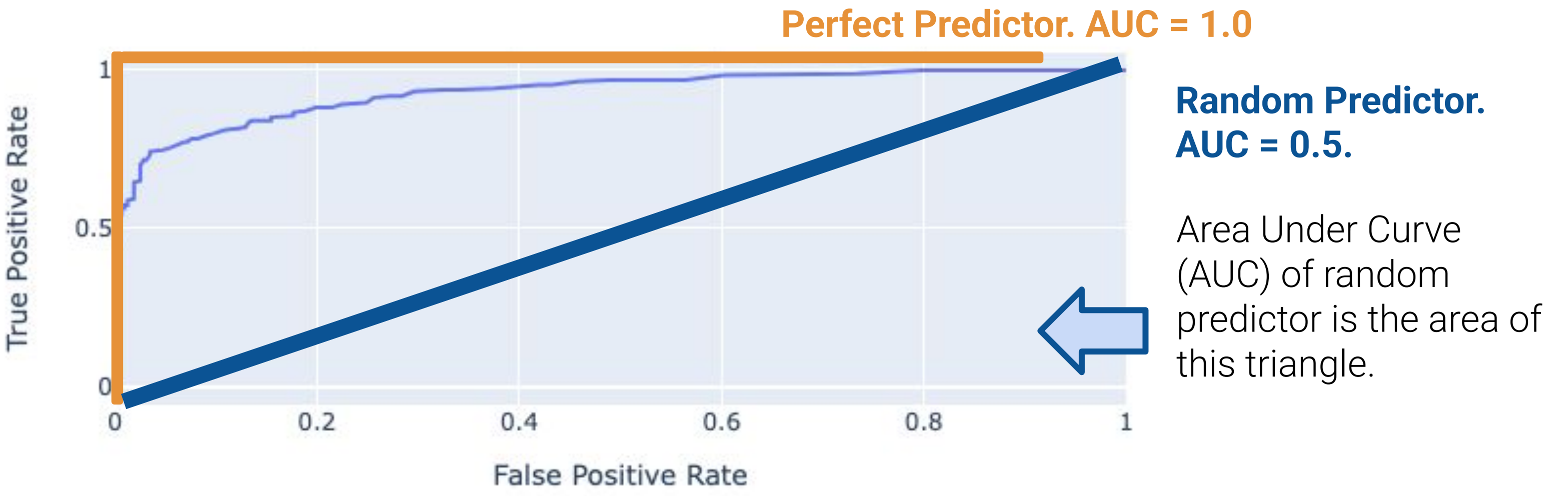
$$\text{FPR} = \frac{FP}{FP + TN}$$

Best possible AUC = 1. Terrible AUC = 0.5.

- Random predictors have an AUC of around 0.5. Why?

A **random predictor** randomly predicts P(Y = 1 | x) to be uniformly between 0 and 1.

This slide was added post-lecture to clarify closing comments.



Area Under Curve (AUC) of random predictor is the area of this triangle.

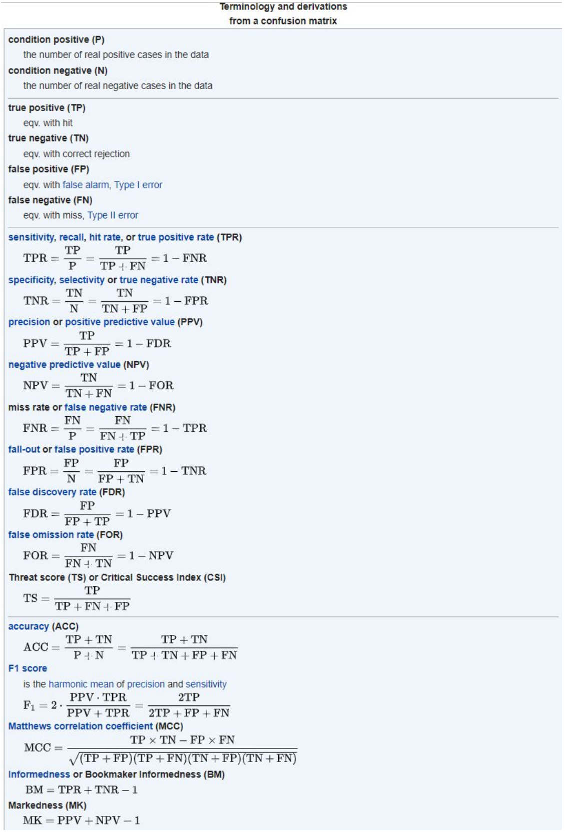
Common techniques for evaluating classifiers

Numerical assessments:

- **Accuracy, precision, recall/TPR, FPR.**
- Area under curve (AUC), for ROC curves.

Visualizations:

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



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.

Extra Slides

Lecture 22, Data 100 Spring 2022

Logistic Regression Model, continued

- sklearn demo
- Maximum Likelihood Estimation: high-level (live), detailed (recorded)

Linear separability and Regularization

Performance Metrics

- Accuracy
- Imbalanced Data, Precision, Recall

Adjusting the Classification Threshold

- A case study
- ROC curves, and AUC

[Extra] Detailed MLE, Gradient Descent, PR curves

Video: link

Out of scope, but useful for understanding where cross-entropy loss comes from.

[Extra] Detailed Maximum Likelihood Estimation

Lecture 22, Data 100 Spring 2022

Logistic Regression Model, continued

- sklearn demo
- Maximum Likelihood Estimation: high-level (live), detailed (recorded)

Linear separability and Regularization

Performance Metrics

- Accuracy
- Imbalanced Data, Precision, Recall

Adjusting the Classification Threshold

- A case study
- ROC curves, and AUC

[Extra] Detailed MLE, Gradient Descent, PR curves

The Modeling Process

1. Choose a model ✅

2. Choose a loss function

3. Fit the model

4. Evaluate model performance

Regression $(y \in \mathbb{R})$

Linear Regression

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

Squared Loss or Absolute Loss

Regularization
Sklearn/Gradient descent

R^2, Residuals, etc.

Classification $(y \in \{0, 1\})$

Logistic Regression

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

Average Cross-Entropy Loss

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

Wherefore use cross-entropy?

atio
Sklearn/Gradient de

??
(next time)

Shakespeare
[Wikipedia]

Why Use Cross-Entropy Loss?

This section will not be directly tested, but you will understand why we minimize cross-entropy loss for logistic regression.

Two common explanations:

- [Information Theory] KL Divergence (textbook)
- [Probability] Maximum Likelihood Estimation (this lecture)

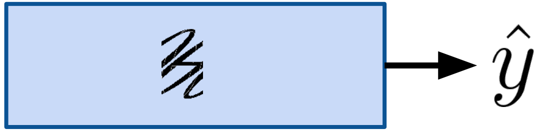
Recall the Coin Demo (No-Input Classification)

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

0.4 is the most “intuitive” θ for two reasons:

1. Frequency of heads in our data
2. Maximizes the **likelihood** of our data: (**proportional to** the probability of our data)

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



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

Prediction: 1 or 0

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

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

data (1’s and 0’s)

likelihood

A Compact Representation of the Bernoulli Probability Distribution

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

$$\underbrace{\{y_1, y_2, \ldots, y_n\}}_{\text{data (1's and 0's)}} \Longrightarrow \hat{\theta} = \underset{\theta}{\operatorname{argmax}} \underbrace{(???)}_{\text{likelihood}}$$

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

$$\boxed{P(Y = y) = \underbrace{p^y}_{\text{For P(Y = 1), only this term stays}} \underbrace{(1-p)^{1-y}}_{\text{For P(Y = 0), only this term stays}}}$$

(long, non-compact form):

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

Generalized Likelihood of Binary Data

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

$$\underbrace{\{y_1, y_2, \ldots, y_n\}}_{\text{data (1's and 0's)}} \Rightarrow \hat{\theta} = \underset{\theta}{\operatorname{argmax}} \underbrace{(???)}_{\text{likelihood}}$$

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

$$P(Y = y) = \underbrace{p^y}_{\text{For P(Y = 1), only this term stays}} \underbrace{(1 - p)^{1-y}}_{\text{For P(Y = 0), only this term stays}}$$

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

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

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

likelihood
vs. probability

$$\binom{10}{4} p^4 (1 - p)^6$$

Generalized Likelihood of Binary Data

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

$$\underbrace{\{y_1, y_2, \ldots, y_n\}}_{\text{data (1's and 0's)}} \Rightarrow \hat{\theta} = \underset{\theta}{\operatorname{argmax}} \underbrace{(???)}_{\text{likelihood}}$$

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

$$P(Y = y) = \underbrace{p^y}_{\text{For P(Y = 1), only this term stays}} \underbrace{(1-p)^{1-y}}_{\text{For P(Y = 0), only this term stays}}$$

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

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

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

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

(spoiler: for logistic regression, $p_i = \sigma(X_i^T\theta)$)

Maximum Likelihood Estimation (MLE)

Our **maximum likelihood estimation** problem:

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

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

Maximum Likelihood Estimation (MLE)

Our **maximum likelihood estimation** problem:

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

Find $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n$ that **maximize**

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

Equivalent, simplifying optimization problems:

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

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

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

Maximum Likelihood Estimation (MLE)

Our **maximum likelihood estimation** problem:

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

Find $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n$ that **maximize**

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

Equivalent, simplifying optimization problems:

maximize $p_1, p_2, \ldots, p_n$

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

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

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

minimize $p_1, p_2, \ldots, p_n$

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

Argmax property:
x that maximizes f(x) will
minimize -f(x)

Maximizing Likelihood == Minimizing Average Cross-Entropy

$$\underset{p_1, p_2, \dots, p_n}{\textbf{argmax}} \quad \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}$$

Log is increasing;
max/min properties

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

Average Cross-Entropy Loss!!

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

$$\underset{\theta}{\textbf{argmin}} \quad -\frac{1}{n}\sum_{i=1}^{n}\Big(y_i \log(\sigma(X_i^T\theta) + (1-y_i)\log(1-\sigma(X_i^T\theta))\Big)$$

Average Cross-Entropy Loss for Logistic Regression!!

[High-Level] Maximum Likelihood Estimation

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

Assumption: all data are independent Bernoulli random variables.

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

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

$$\underset{p_1, p_2, \ldots, p_n}{\textbf{argmax}} \quad \prod_{i=1}^{n} \underbrace{p_i^{y_i}(1-p_i)^{(1-y_i)}}_{\text{Prob. that i-th response is yi}}$$

[High-Level] Maximum Likelihood Estimation

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

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

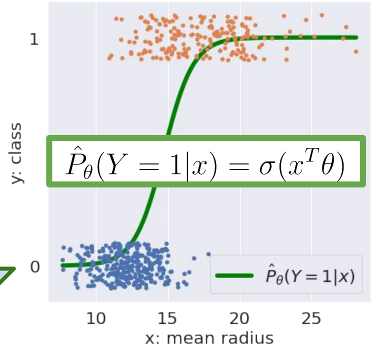
For logistic regression, let $p_i = \sigma(X_i^T\theta)$

Assumption: all data are independent Bernoulli random variables

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

Prob. that i-th response is yi

Main takeaway: The optimal theta that minimizes mean cross-entropy loss “pushes” all probabilities in the direction of the true class.



Want

$\sigma(x^T\theta) \to 1$

Want

$\sigma(x^T\theta) \to 0$

[High-Level] Maximum Likelihood Estimation

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

Assumption: all data are independent Bernoulli random variables

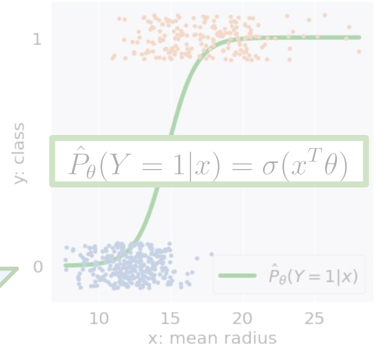
It turns out that many of the model + loss combinations we've seen can be motivated using MLE.

- OLS, Ridge Regression, etc.
- You will study MLE further in probability and ML classes. But now you know it exists.

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

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

$$\underset{p_1, p_2, \ldots, p_n}{\textbf{argmax}} \quad \prod_{i=1}^{n} \underbrace{p_i^{y_i}(1-p_i)^{(1-y_i)}}_{\text{Prob. that i-th response is yi}}$$



Want $\sigma(x^T\theta) \to 1$

Want $\sigma(x^T\theta) \to 0$

We Did it!

1. Choose a model ✓

2. Choose a loss function ✓

3. Fit the model

4. Evaluate model performance

Regression $(y \in \mathbb{R})$

Linear Regression

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

Squared Loss or Absolute Loss

Regularization
Sklearn/Gradient descent

R^2, Residuals, etc.

Classification $(y \in \{0, 1\})$

Logistic Regression

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

Average Cross-Entropy Loss

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

rizatio
ent d

??
(next time)

That which we call a rose would by any other name smell as sweet.

Shakespeare
[Wikipedia]

Reference slides. Out of scope.

[Extra] Gradient Descent for Logistic Regression

Lecture 22, Data 100 Spring 2022

Logistic Regression Model, continued

- sklearn demo
- Maximum Likelihood Estimation: high-level (live), detailed (recorded)

Linear separability and Regularization

Performance Metrics

- Accuracy
- Imbalanced Data, Precision, Recall

Adjusting the Classification Threshold

- A case study
- ROC curves, and AUC

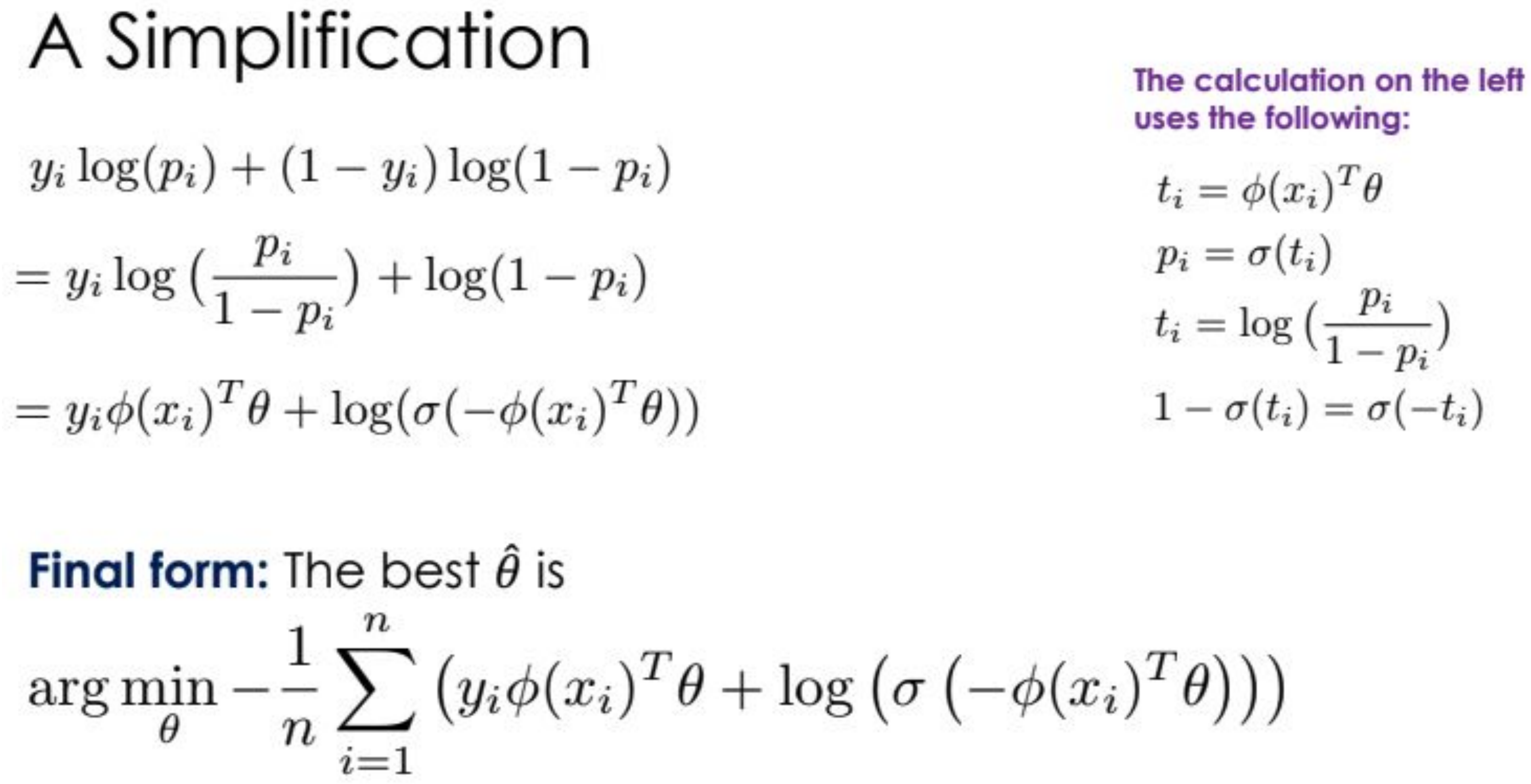
[Extra] Detailed MLE, **Gradient Descent**, PR curves

[Extra] Gradient Descent for Logistic Regression

	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	Average Cross-Entropy Loss $-\frac{1}{n}\sum_{i=1}^{n}(y_i \log(\sigma(X_i^T\theta) + (1 - y_i)\log(1 - \sigma(X_i^T\theta)))$
3. Fit the model	Regularization Sklearn/Gradient descent	Regularization Sklearn/**Gradient descent**
4. Evaluate model performance ✓	R^2, Residuals, etc.	Accuracy, Precision, Recall, ROC Curves

Simplifying Average Cross-Entropy Loss

Slides from Spring 2020.



A Simplification

$$y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

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

$$= y_i \phi(x_i)^T \theta + \log(\sigma(-\phi(x_i)^T \theta))$$

The calculation on the left uses the following:

$$t_i = \phi(x_i)^T \theta$$

$$p_i = \sigma(t_i)$$

$$t_i = \log\left(\frac{p_i}{1 - p_i}\right)$$

$$1 - \sigma(t_i) = \sigma(-t_i)$$

Final form: The best $\hat{\theta}$ is

$$\arg\min_{\theta} -\frac{1}{n} \sum_{i=1}^{n} \left(y_i \phi(x_i)^T \theta + \log\left(\sigma\left(-\phi(x_i)^T \theta\right)\right)\right)$$

Gradient of Average Cross-Entropy Loss

Slides from Spring 2020.

- Want to minimize

$$\mathbf{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log(\sigma(-\phi(x_i)^T\theta))\right)$$

- Take Derivative:

$$\nabla_\theta \mathbf{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\nabla_\theta y_i\phi(x_i)^T\theta + \nabla_\theta \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)$$

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

$$\nabla_\theta \mathbf{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i\phi(x_i) + \frac{1}{\sigma\left(-\phi(x_i)^T\theta\right)}\nabla_\theta\sigma\left(-\phi(x_i)^T\theta\right)$$

Derivative

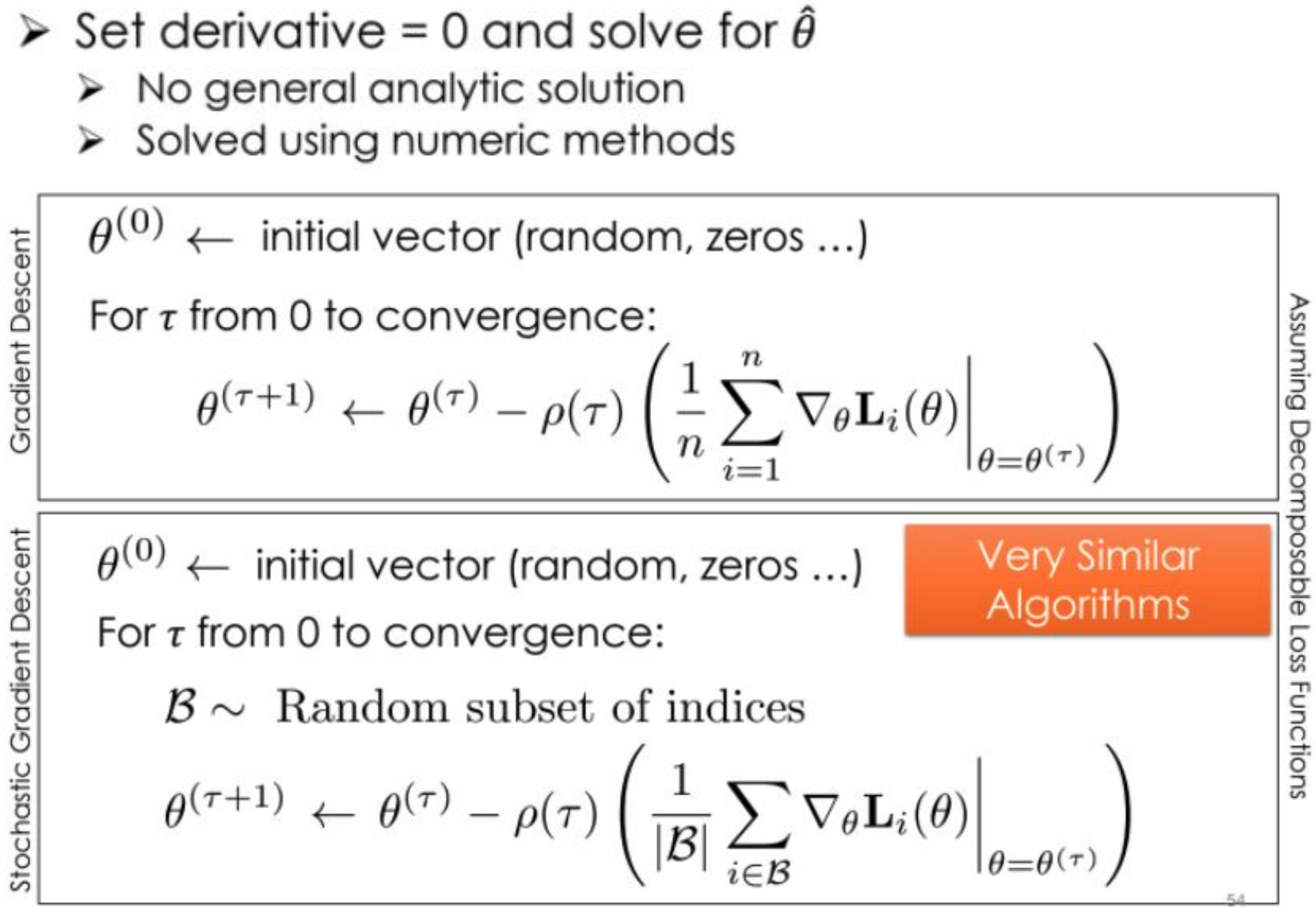
$$\frac{d}{dt}\sigma(t) = \sigma(t)\sigma(-t)$$

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

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

Gradient Descent Algorithms

Slides from Spring 2020.



- Set derivative = 0 and solve for $\hat{\theta}$
  - No general analytic solution
  - Solved using numeric methods

Gradient Descent

```
$\theta^{(0)} \leftarrow$ initial vector (random, zeros ...)
For $\tau$ from 0 to convergence:
    $\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau) \left( \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \mathbf{L}_i(\theta) \Big|_{\theta=\theta^{(\tau)}} \right)$
```

Stochastic Gradient Descent

```
$\theta^{(0)} \leftarrow$ initial vector (random, zeros ...)
For $\tau$ from 0 to convergence:
    $\mathcal{B} \sim$ Random subset of indices
    $\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau) \left( \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_\theta \mathbf{L}_i(\theta) \Big|_{\theta=\theta^{(\tau)}} \right)$
```

Very Similar Algorithms

Assuming Decomposable Loss Functions

54

Reference slides. Out of scope.

[Extra] Precision-Recall Curves

Lecture 22, Data 100 Spring 2022

Logistic Regression Model, continued

- sklearn demo
- Maximum Likelihood Estimation: high-level (live), detailed (recorded)

Linear separability and Regularization

Performance Metrics

- Accuracy
- Imbalanced Data, Precision, Recall

Adjusting the Classification Threshold

- A case study
- ROC curves, and AUC

[Extra] Detailed MLE, Gradient Descent, **PR curves**

Precision vs. threshold

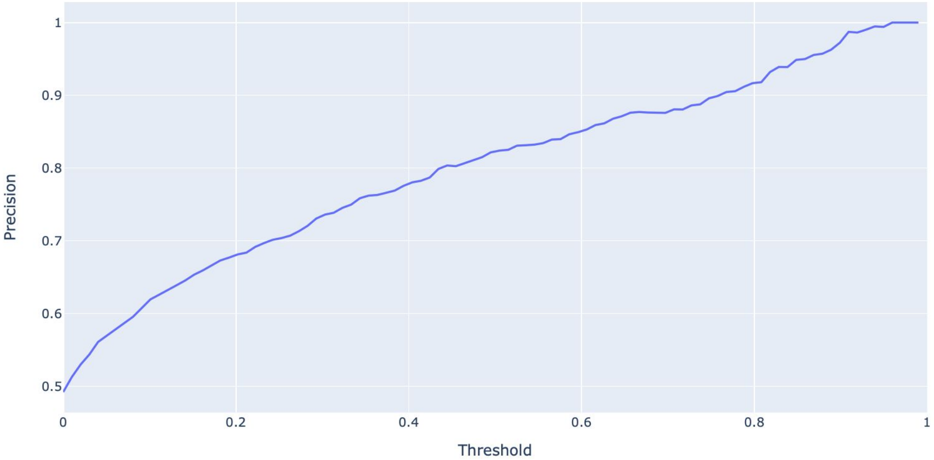
As we increase our threshold, we have fewer and fewer false positives.

- 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

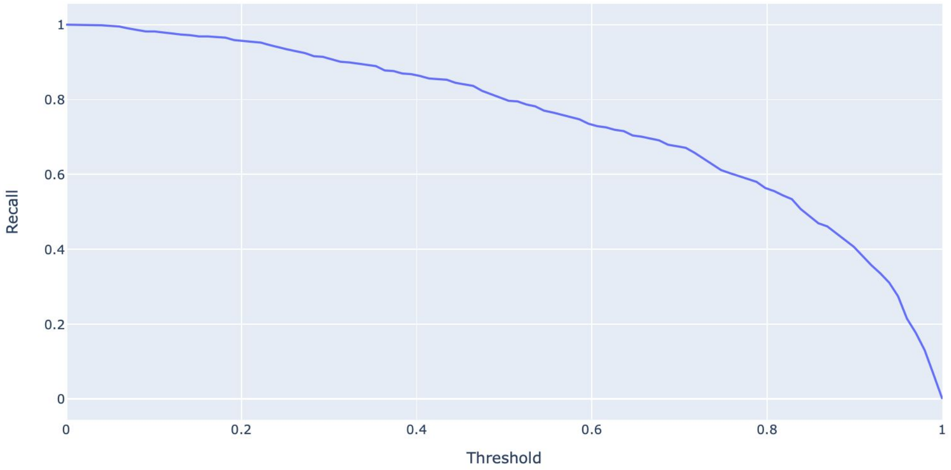
As we increase our threshold, we have more and more false negatives.

- Thus, recall tends to decrease.

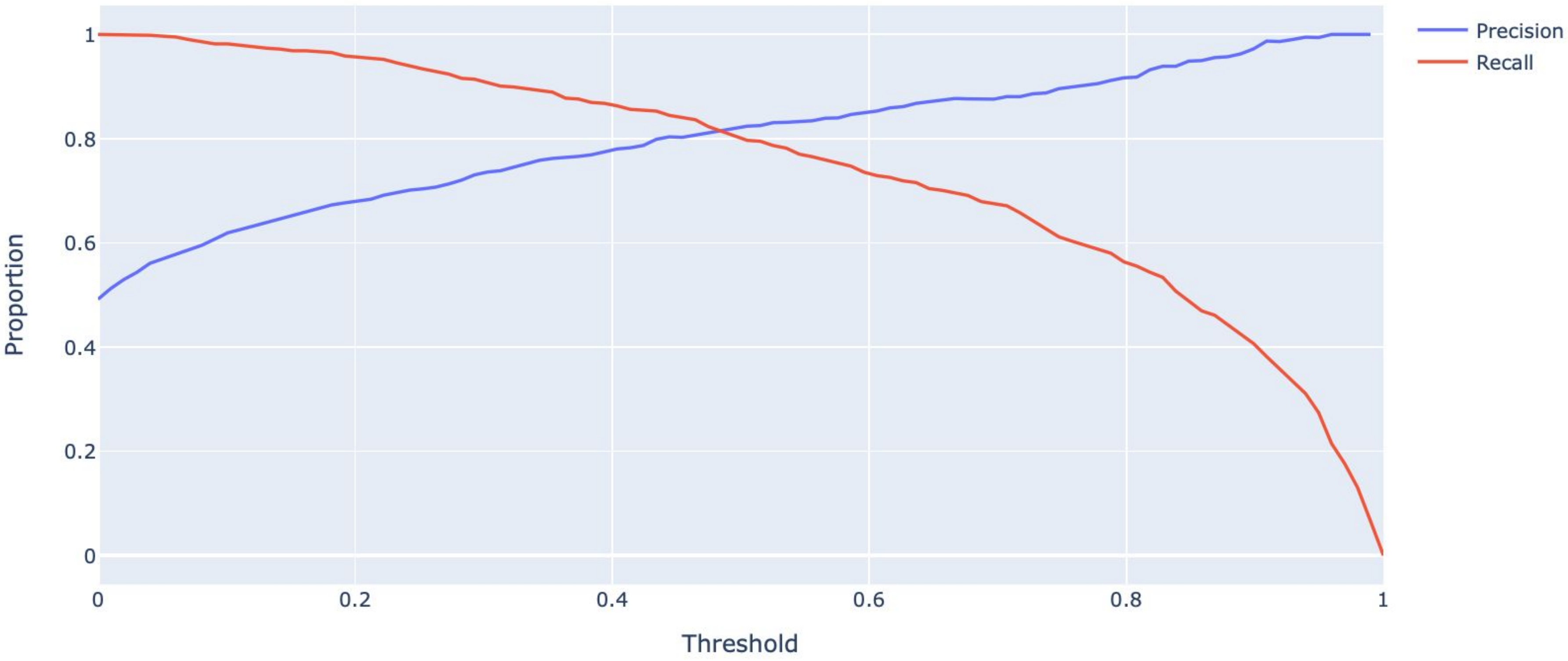
$$\textbf{Recall} = \frac{\textbf{True Positives}}{\textbf{True Positives} + \textbf{False Negatives}}$$

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

Recall strictly decreases as we increase our threshold. Why?



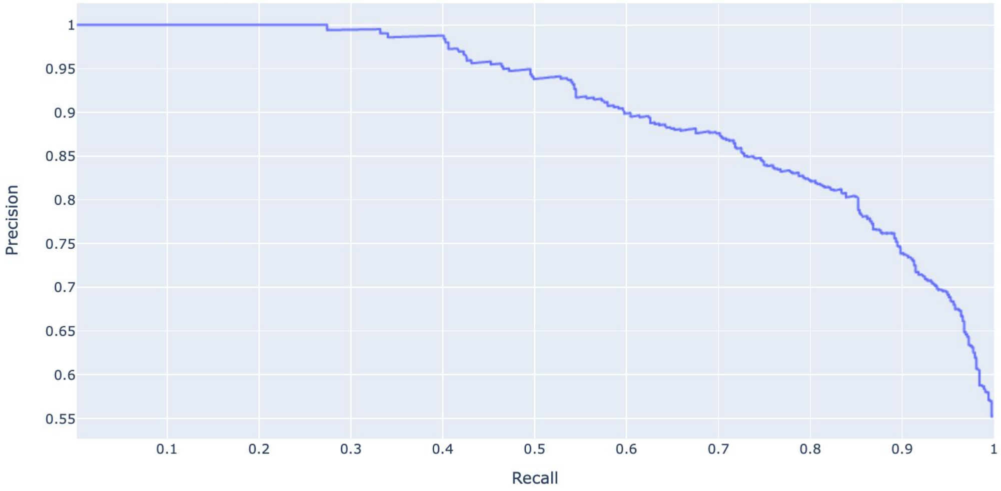
Precision and Recall vs. Threshold



Precision-recall curves

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

1. Which part of this curve corresponds to T = 0.9?
2. Which part of this curve corresponds to T = 0.1?

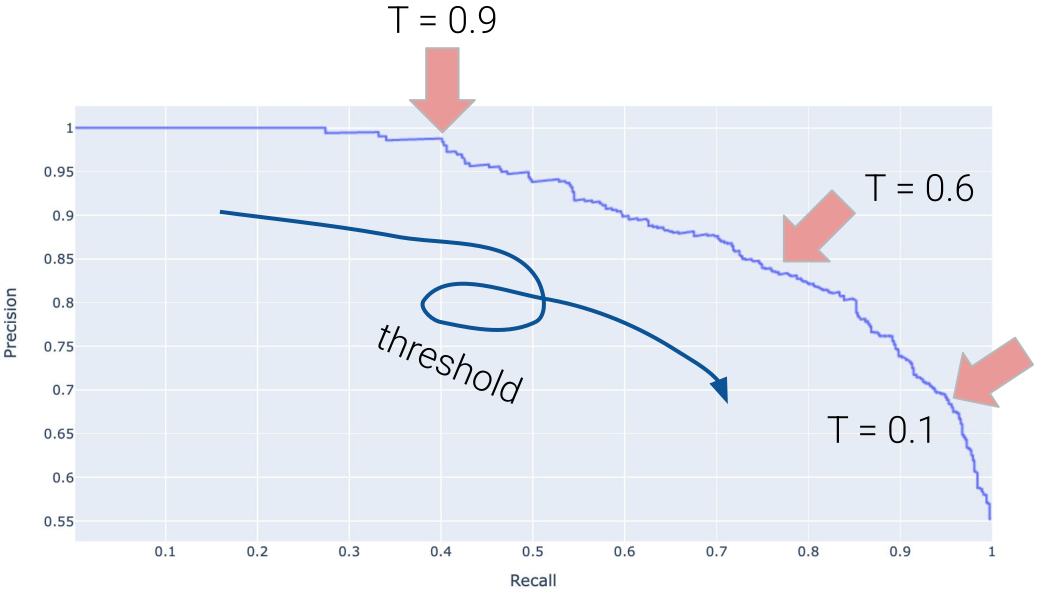


Precision-recall curves

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

Answer:

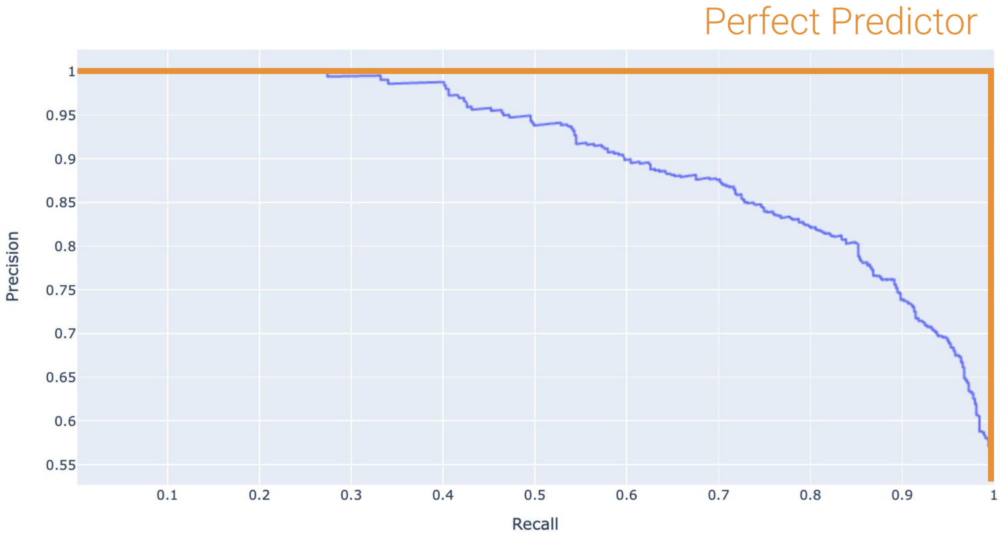
- Threshold decreases from the top left to the bottom right.
- In the notebook, there's an interactive version of this plot.



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.



LECTURE 22

Logistic Regression II

Content credit: Lisa Yan, Suraj Rampure, Ani Adhikari, Josh Hug, Joseph Gonzalez