LECTURE 33

Logistic Regression

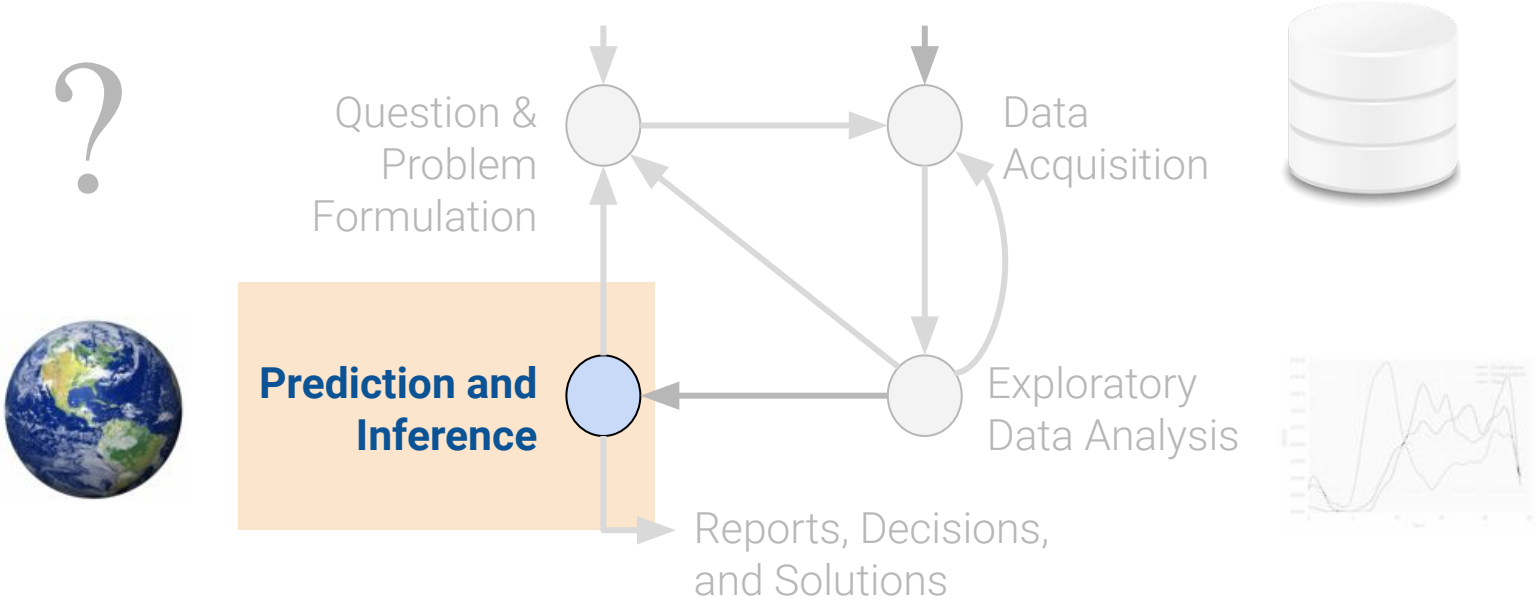
Moving from regression to classification.

CSCI 3022

Announcements

- Project Part 2 - due Thursday at 11:59pm MT (no late submissions accepted)
  - Please make sure you are using V2
- Last day of class: Wednesday (Review Day)
- Final Exam:
  - Section 1: Tuesday Dec 19th: 4:30-7pm (in our classroom)
  - **Section 2: Wednesday Dec 20th: 1:30-4pm (in our classroom)**
    - You are allowed a 2 sided crib sheet
    - You are allowed a calculator
    - Bring your Buff One card

A Different Modeling Problem

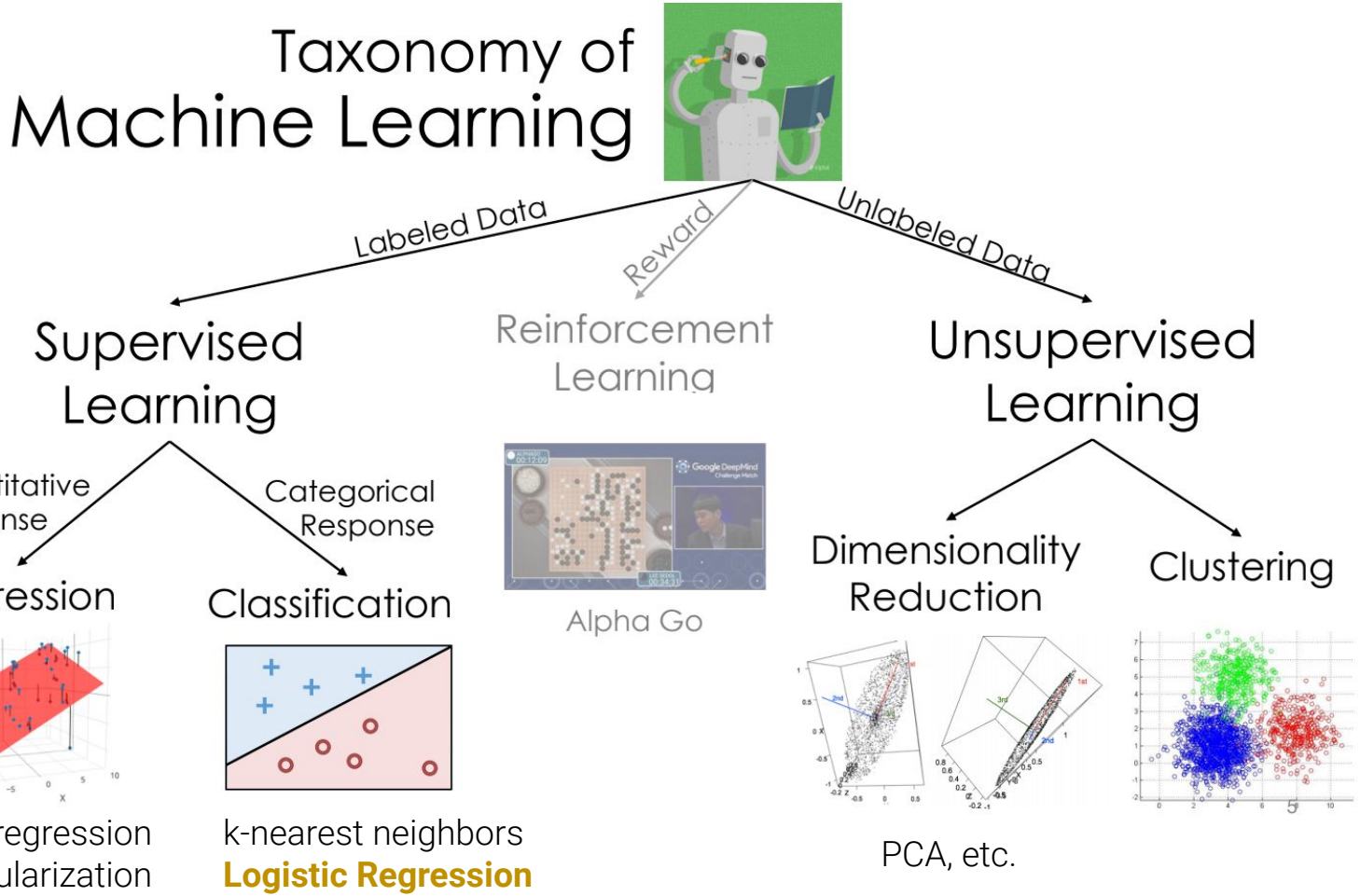


(today)

Logistic Regression
The Model
Cross-Entropy Loss
The Probabilistic View

Taxonomy of Machine Learning

Regression and classification are both forms of **supervised learning**.



Kinds of Classification

We are interested in predicting some **categorical variable**, or **response**, y.

Binary classification [today]

- Two classes
- **Responses** y are either 0 or 1

win or lose

disease or no disease

spam or ham

Multiclass classification

- Many classes
- Examples:
  - Image labeling (tuxedo cat, black/white yarn, and moon crescent),
  - Next word in a sentence.

Structured prediction tasks

- Multiple related classification predictions.
- Examples: Translation, voice recognition, etc.

Maximum Likelihood Estimation

Regression vs. Classification

- **Maximum Likelihood Estimation**

The No-Input Binary Classifier

Suppose you observed some outcomes of a coin (1 = Heads, 0 = Tails):

$$\{0, 0, 1, 1, 1, 1, 0, 0, 0, 0\}$$

Training data has only responses $\mathbb{Y}$ (no features $\mathbb{X}$)

For the next flip, do you predict heads or tails?

The No-Input Binary Classifier

Suppose you observed some outcomes of a coin (1 = Heads, 0 = Tails):

$$\{0, 0, 1, 1, 1, 1, 0, 0, 0, 0\}$$

Training data has only responses $\mathbb{Y}$ (no features $\mathbb{X}$)

For the next flip, do you predict heads or tails?

A reasonable model is to **assume all flips are IID** (i.e., same coin; same prob. of heads θ).

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

Prediction: 1 or 0

1. Of the options below, which is the best theta θ? Why?

A. 0.8 B. 0.5 C. 0.4
D. 0.2 E. Something else

2. For the next flip, would you predict 1 or 0?

The No-Input Binary Classifier

Suppose you observed some outcomes of a coin (1 = Heads, 0 = Tails):

$$\{0, 0, 1, 1, 1, 1, 0, 0, 0, 0\}$$

Training data has only responses $\mathbb{Y}$ (no features $\mathbb{X}$)

For the next flip, do you predict heads or tails?

A reasonable model is to **assume all flips are IID** (i.e., same coin; same prob. of heads θ).

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

Prediction: 1 or 0

1. Of the options below, which is the best theta θ? Why?

A. 0.8 B. 0.5 C. 0.4
D. 0.2 E. Something else

2. For the next flip, would you predict 1 or 0?

The No-Input Binary Classifier

Suppose you observed some outcomes of a coin (1 = Heads, 0 = Tails):

$$\{0, 0, 1, 1, 1, 1, 0, 0, 0, 0\}$$

Training data has only responses $\mathbb{Y}$ (no features $\mathbb{X}$)

For the next flip, do you predict heads or tails?

A reasonable model is to **assume all flips are IID** (i.e., same coin; same prob. of heads θ).

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

Prediction:
1 or 0

1. Of the below, which is the best theta θ? Why?

A. 0.8 **B.** 0.5 **C.** 0.4
D. 0.2 **E.** Something else

The No-Input Binary Classifier

Suppose you observed some outcomes of a coin (1 = Heads, 0 = Tails):

$$\{0, 0, 1, 1, 1, 1, 0, 0, 0, 0\}$$

Training data has only responses $\mathbb{Y}$ (no features $\mathbb{X}$)

For the next flip, do you predict heads or tails?

A reasonable model is to **assume all flips are IID** (i.e., same coin; same prob. of heads θ).

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

Prediction:
1 or 0

1. Of the below, which is the best theta θ? Why?

A. 0.8 **B.** 0.5 **C. 0.4**
D. 0.2 **E.** Something else

0.4 is the most "intuitive" for two reasons:
1. Frequency of heads in our data.
2. Maximizes the **likelihood** of our data.

The No-Input Binary Classifier

Suppose you observed some outcomes of a coin (1 = Heads, 0 = Tails):

$$\{0, 0, 1, 1, 1, 1, 0, 0, 0, 0\}$$

Training data has only responses $\mathbb{Y}$ (no features $\mathbb{X}$)

For the next flip, do you predict heads or tails?

A reasonable model is to **assume all flips are IID** (i.e., same coin; same prob. of heads θ).

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

Prediction:
1 or 0

1. Of the below, which is the best theta θ? Why?

A. 0.8 **B.** 0.5 **C. 0.4**
D. 0.2 **E.** Something else

2. For the next flip, would you predict 1 or 0?

The most frequent outcome in the sample which is tails.

The No-Input Binary Classifier

Suppose you observed some outcomes of a coin (1 = Heads, 0 = Tails):

$$\{0, 0, 1, 1, 1, 1, 0, 0, 0, 0\}$$

Training data has only responses $\mathbb{Y}$ (no features $\mathbb{X}$)

For the next flip, do you predict heads or tails?

Recall Hw 2 # 4

Question 4). Applying Those Prereqs: A Maximum Likelihood Estimate

In this problem we're going to apply your calculus and discrete math prerequisite knowledge to introduce a data science concept called a maximum likelihood estimate.

Data scientists use coin tossing as a visual image for sampling at random with replacement from a binary population.

Question 4a)

A coin that lands heads with chance 0.7 is tossed six times. What is the chance of the sequence HHHTHT? Assign your answer to the variable `p_HHHTHT`.

```
p_HHHTHT = ...
p_HHHTHT
```

```
grader.check("4a")
```

Question 4b)

I have a coin that lands heads with an unknown probability p.

Suppose I toss it 10 times and get the sequence TTTHTHHTTH.

If you toss this coin 10 times, the chance that you get the sequence above is a function of p. That function is called the *likelihood* of the sequence TTTHTHHTTH, so we will call it $L(p)$.

What is $L(p)$ for the sequence TTTHTHHTTH?

Write your answer using LaTeX below (i.e. your answer should be of the form: $L(p)$=some function of p)

Question 4e)

Let's prove what you observed graphically above. That is, let's use calculus to find $\hat{p}$.

But wait before you start trying to find the value p where $L'(p) = 0$ (trust us, the algebra is not pretty...)

TIP:
The value $\hat{p}$ at which the function L attains its maximum is the same as the value at which the function $\log(L)$ attains its maximum. To clarify, $\log(L)$ is the composition of $\log$ and L: $\log(L)$ at p is $\log(L(p))$. Even though it doesn't make a difference for this problem, $\log$ is now and forevermore the $\log$ to the base e, not to the base 10.

This tip is hugely important in data science because many probabilities are products and the $\log$ function turns products into sums. It's much simpler to work with a sum than with a product.

Armed with that tip use calculus to find $\hat{p}$. You don't have to check that the value you've found produces a max and not a min – we'll spare you that step.

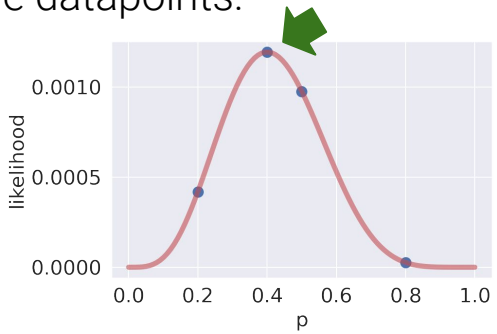
Likelihood of Data; Definition of Probability

A Bernoulli random variable Y with parameter p has distribution: $P(Y = y) = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{if } y = 0 \end{cases}$

Given that all flips are IID from the same coin (probability of heads = p), **likelihood** of our data is **proportional to** the probability of observing the datapoints.

Training data: [0, 0, 1, 1, 1, 1, 0, 0, 0, 0]

Data likelihood: L(p) = $p^4(1-p)^6$.



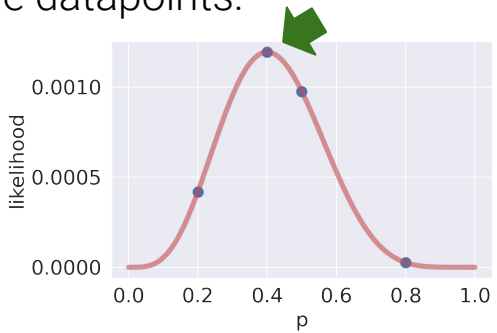
Likelihood of Data

A Bernoulli random variable Y with parameter p has distribution: $P(Y = y) = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{if } y = 0 \end{cases}$

Given that all flips are IID from the same coin (probability of heads = p), the **likelihood** of our data is **proportional to** the probability of observing the datapoints.

Training data: [0, 0, 1, 1, 1, 1, 0, 0, 0, 0]

Data likelihood: $p^4(1-p)^6$.



An example of a bad estimate is parameter $p = 0.1$ since the likelihood of observing the training data is going to be :

$$(0.1)^4(0.9)^6 = 0.000053$$

An example of a good estimate is parameter $p = 0.4$ since the likelihood of observing the training data is going to be :

$$(0.4)^4(0.6)^6 = 0.001194$$

Generalization of the Coin Demo

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:

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

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?

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

data (1's and 0's) — likelihood

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

$$P(Y = y) = p^y (1 - p)^{1-y}$$

For P(Y = **1**), only this term stays

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?

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

data (1's and 0's) — likelihood

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

$$P(Y = y) = p^y (1-p)^{1-y}$$

For P(Y = **1**), only this term stays

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} → $p^4(1-p)^6$

Generalized Likelihood of Binary Data

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

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

data (1's and 0's) — likelihood

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

$$P(Y = y) = p^y (1-p)^{1-y}$$

For P(Y = **1**), only this term stays

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

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

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

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 (since we need to take the first derivative):

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

(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 (since we need to take the first derivative):

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).)

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

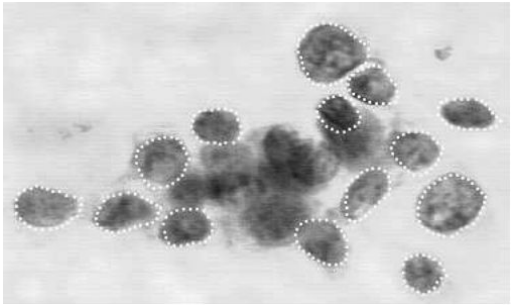
minimize $p_1, p_2, \ldots, p_n$ $-\frac{1}{n}\sum_{i=1}^{n} \left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$

Regression vs. Classification

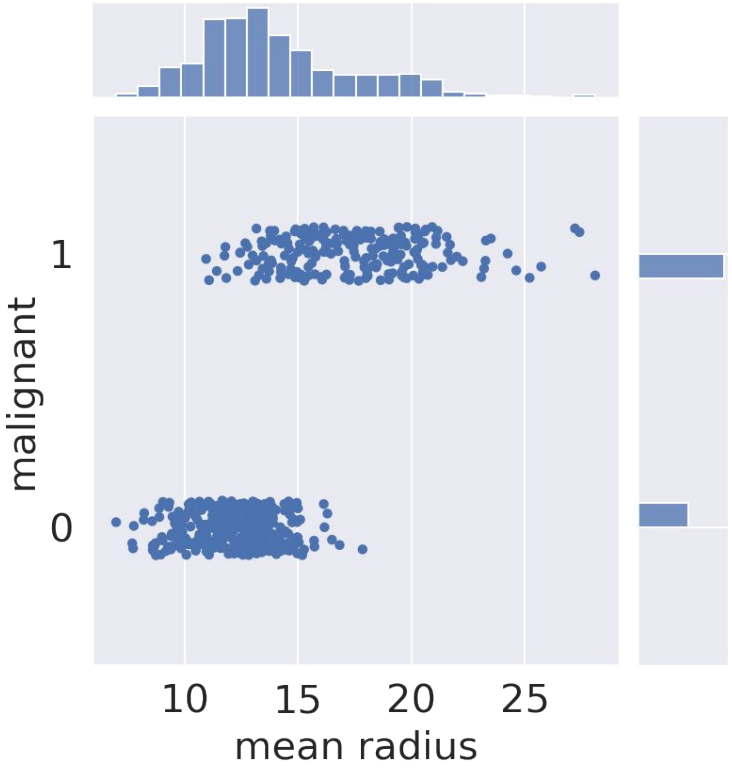
The Breast Cancer Wisconsin Dataset

Input x: mean radius of breast tumor cells
Response y: 1 if malignant, 0 if benign

mean radius	malignant
17.99	1
20.57	1
19.69	1
11.42	1
20.29	1
...	...
21.56	1
20.13	1
16.60	1
20.60	1
7.76	0



512 training observations, 57 test



Given the (single) **input feature**,
how can we predict the **label class**?

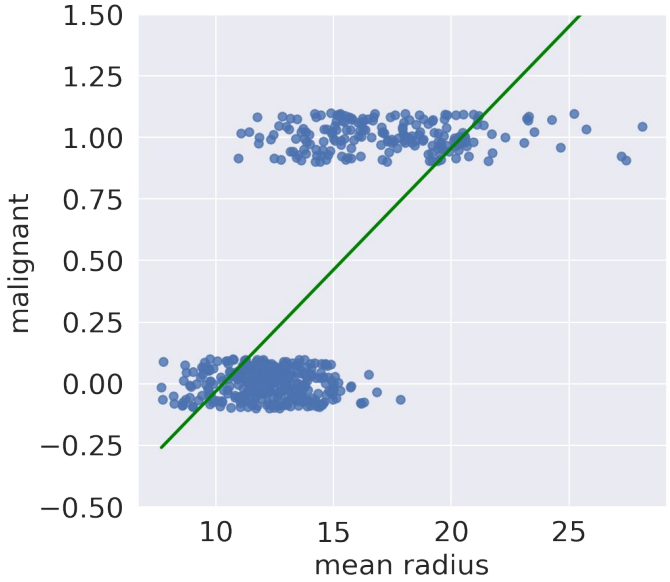
Classification labels are jittered to avoid overplotting.

Why Not Use Least Squares Linear Regression?

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

This is a valid model…

- It assumes y is continuous.
- It is the line that minimizes MSE for the training data.



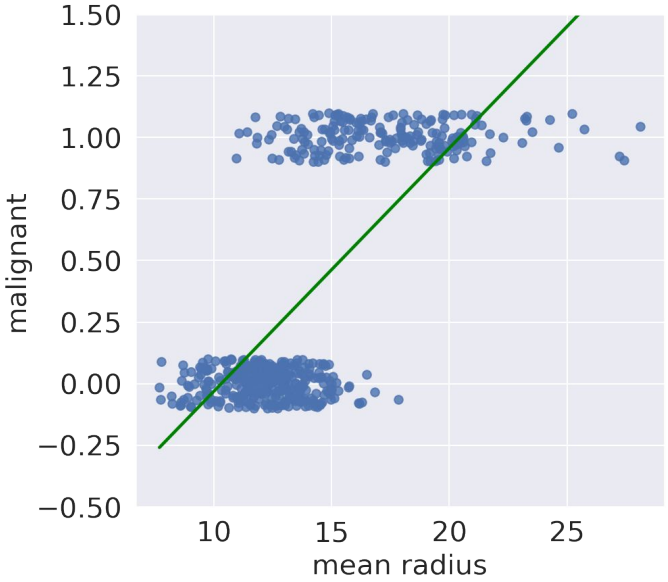
Demo

Why Not Use Least Squares Linear Regression?

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

This is a valid model…

- It assumes y is continuous.
- It is the line that minimizes MSE for the training data.



…but not a *good* model:

- The output $\hat{y}$ can be outside of the label range {0, 1}.
- What does a response of value of -2 mean?

Possible classification: assign $\hat{y}$ > 0.5 to 1, and 0 otherwise.

- Boundary very sensitive to outliers.

Demo

Why Not Use Least Squares Linear Regression?

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

This is a valid model…

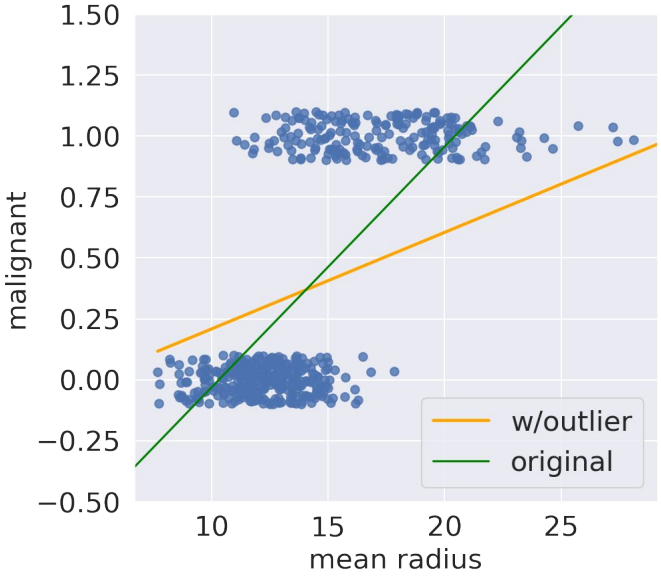
- It assumes y is continuous.
- It is the line that minimizes MSE for the training data.

…but not a *good* model:

- The output $\hat{y}$ can be outside of the label range {0, 1}.
- What does a response of value of -2 mean?

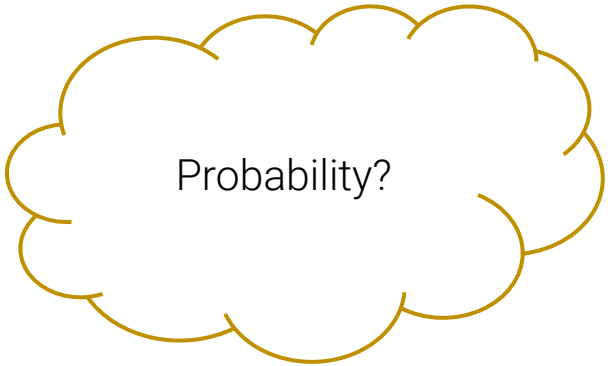
Possible classification: assign $\hat{y} > 0.5$ to 1 and 0 otherwise.

- Boundary very sensitive to outliers.



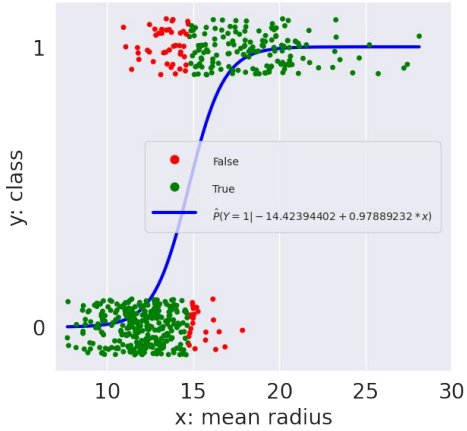
Demo

The Modeling Process

	Regression ($y \in \mathbb{R}$)	Classification ($y \in \{0, 1\}$)
1. Choose a model	Linear Regression $\hat{y} = f_\theta(x) = x^T\theta$	
2. Choose a loss function	Squared Loss or Absolute Loss	
3. Fit the model	Use built-in Python functions	Use built-in Python functions
4. Evaluate model performance	R^2, Residuals, etc.	??

The Sigmoid Function

We need some type of S-shaped curve



1600133

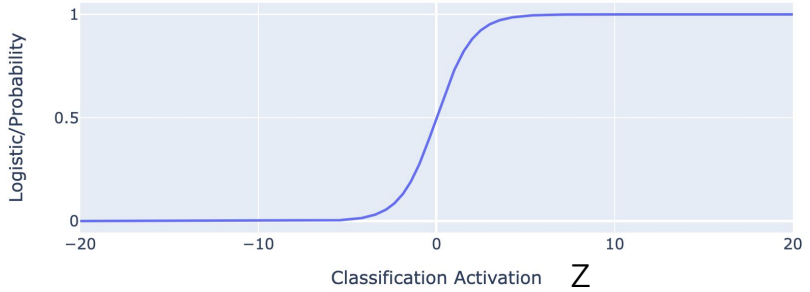
Enter: the **sigmoid function**.

$$p = \frac{1}{1+e^{-z}}$$

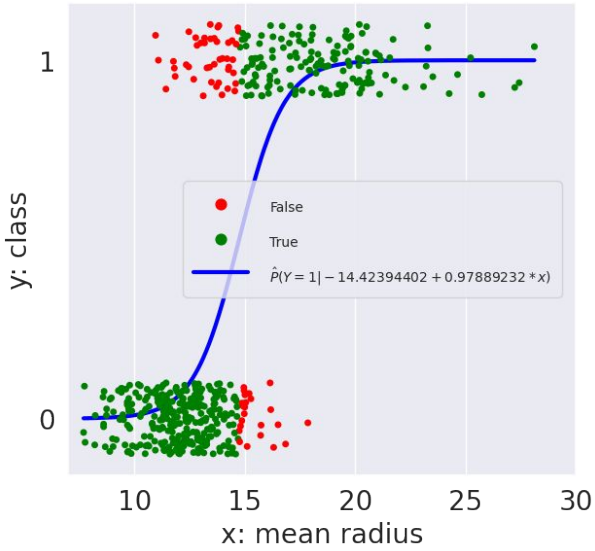
$$\text{sigm}(z) = \frac{1}{1+e^{-z}}$$

Has **awesome** properties:

- Behaves like a probability
- Distinguishes between points
- Really smooth



The Logistic Regression Model



The **logistic regression model** for the probability of a datapoint belonging to Class 1 is given by:: (See Appendix for Derivation)

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

To predict a probability:

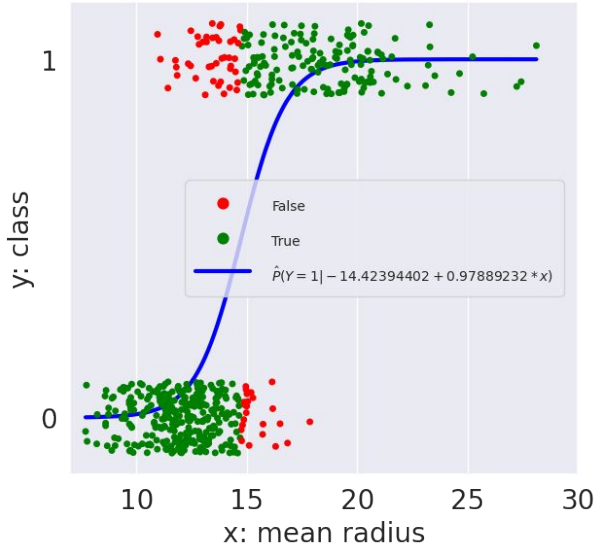
- Compute $-(\theta_0+\theta_1 x)$
- Apply the **sigmoid function** $p=\frac{1}{1+e^{-z}}$

The Logistic Regression Model

The **logistic regression model** for the probability of a datapoint belonging to Class 1 is given by

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

1600133




```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(fit_intercept=True)
model.fit(X, Y); # X, Y are training data

model.intercept_, model.coef_
```

```
(array([-14.42394402]), array([[0.97889232]]))
```

Properties of the Logistic Model

Consider a logistic regression model with one feature and an intercept term (Desmos): 1600133

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

Properties:

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

This generalizes to more than one feature

Suppose we're trying to use statistics about teams to predict who will win a game

1600133

GAME_ID	TEAM_NAME	MATCHUP	REB	FTM	TOV	GOAL_DIFF	WON
21700001	Boston Celtics	BOS @ CLE	46	19	12	-0.049	0
21700002	Golden State Warriors	GSW vs. HOU	41	19	17	0.053	0
21700003	Charlotte Hornets	CHA @ DET	47	23	17	-0.030	0
21700004	Indiana Pacers	IND vs. BKN	47	25	14	0.041	1
21700005	Orlando Magic	ORL vs. MIA	50	22	15	0.042	1

Input: numeric features

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

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

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

Decision rule

Output: **class**

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

Example Calculation

Suppose I want to predict the probability that a tumor is malignant, given **mean radius** (first feature) and **mean smoothness** (second feature).

Suppose I fit a logistic regression model (with no intercept) using my training data, and somehow estimate the optimal parameters:

$$= [0.1 \quad -0.5]$$

Now, you encounter a new breast tumor image:

$$= [15 \quad 1]$$

---

1. What is the probability that the tumor is malignant (Y = 1)?
2. What would you predict as response? 1 or 0?

Example Calculation

Suppose I want to predict the probability that a tumor is malignant, given **mean radius** (first feature) and **mean smoothness** (second feature).

Suppose I fit a logistic regression model (with no intercept) using my training data, and somehow estimate the optimal parameters:

$$\hat{\theta}^T = [0.1 \qquad -0.5]$$

Now, you encounter a new breast tumor image:

$$x^T = [15 \qquad 1]$$

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

$$= \frac{1}{1+e^{-1}}$$
$$\approx 0.7311$$

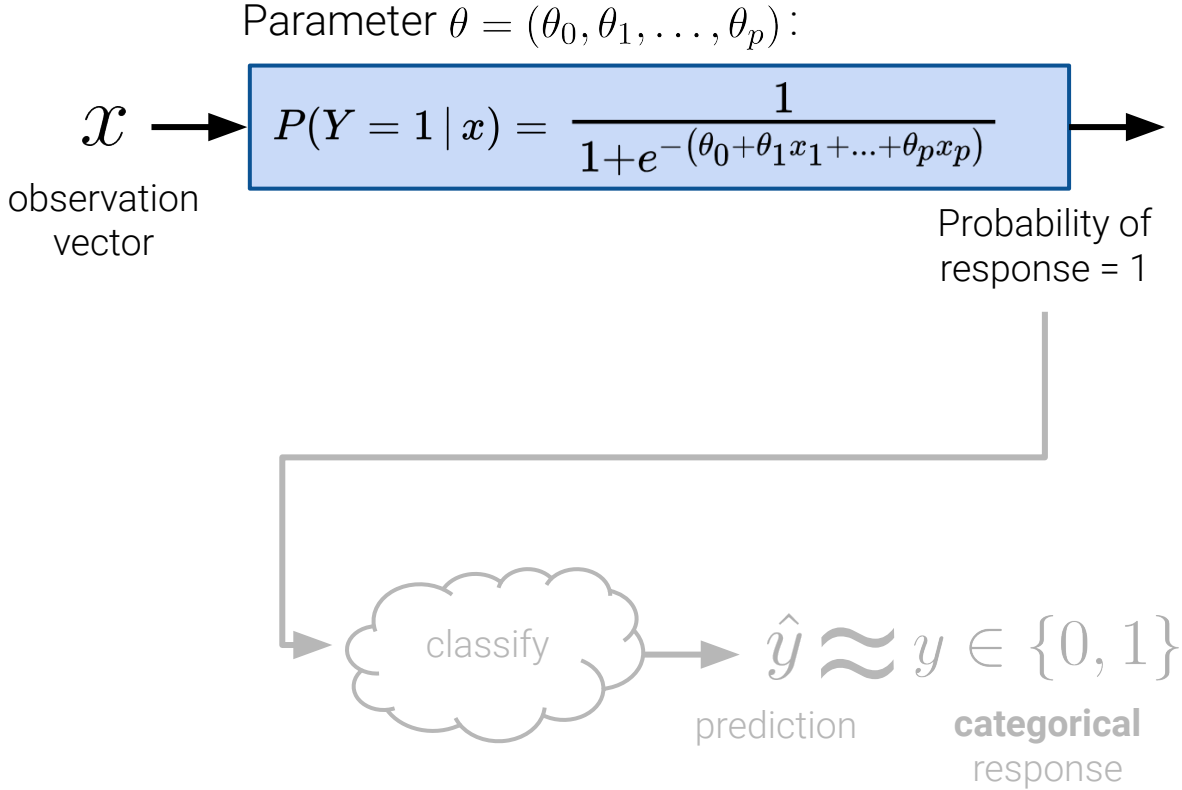
Because the response is more likely to be 1 than 0, a reasonable prediction is

$$\hat{y} = 1$$

Engineering, Part 1: Deciding a Model

Feature Engineering:

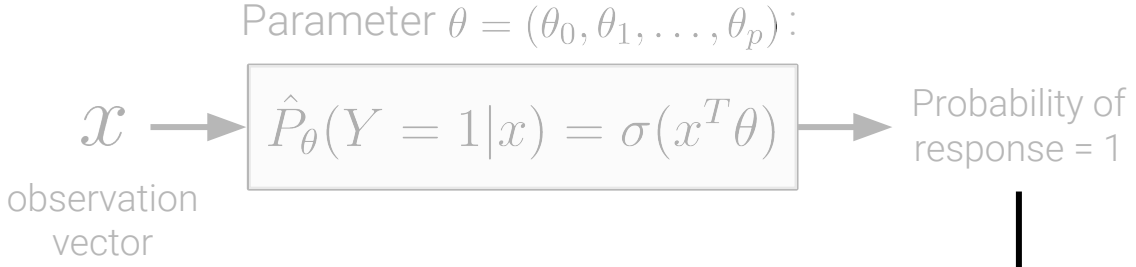
What are the features x that generate great probabilities for prediction?



Engineering, Part 2: Deciding a Classification Threshold

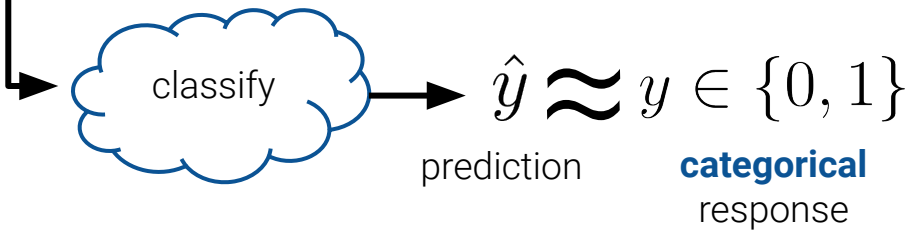
Classification:

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



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

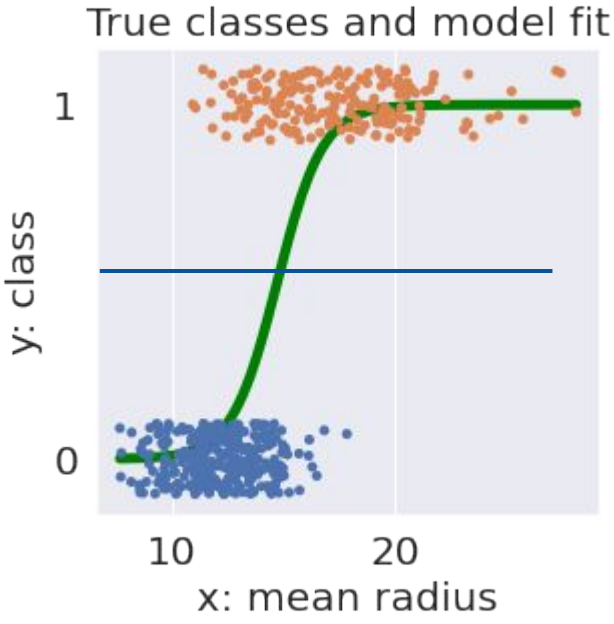
sklearn's model.predict() uses fixed 0.5



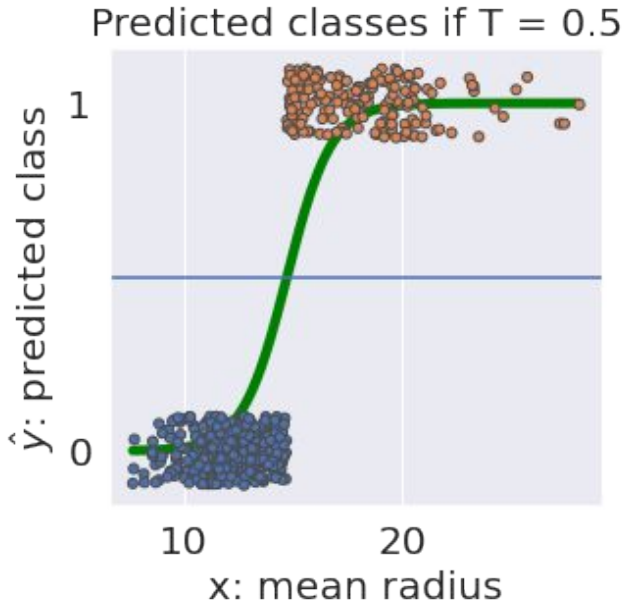
Classification Threshold

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

The default threshold in sklearn is T = 0.5.



	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

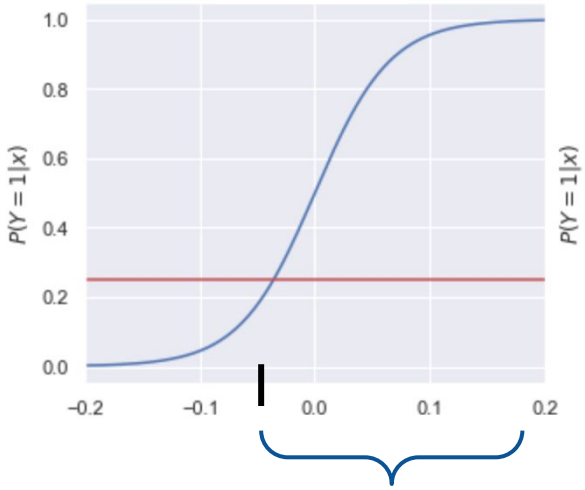


Classification Threshold

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

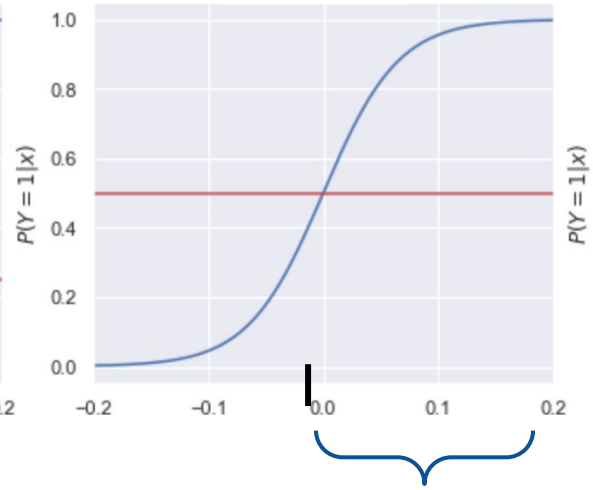
As we increase the threshold T, we “raise the standard” of how confident our classifier needs to be to predict 1 (i.e., “positive”).

T = 0.25

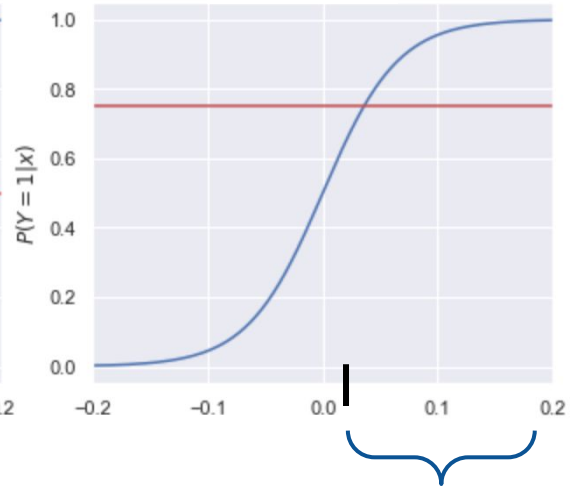


These x will all predict 1

T = 0.50



T = 0.75



Fewer positives

The Modeling Process

	Regression ($y \in \mathbb{R}$)	Classification ($y \in \{0, 1\}$)
1. Choose a model ✅	Linear Regression $\hat{y} = f_\theta(x) = x^T\theta$	Logistic Regression $\frac{1}{1+e^{-(\theta_0+\theta_1 x_1+\ldots+\theta_p x_p)}}$
2. Choose a loss function	Squared Loss or Absolute Loss	??
3. Fit the model	Regularization Sklearn/Gradient descent	Regularization Sklearn/Gradient descent
4. Evaluate model performance	R^2, Residuals, etc.	Accuracy, Precision, Recall, ROC Curves

The Modeling Process

	Regression ($y \in \mathbb{R}$)	Classification ($y \in \{0, 1\}$)
1. Choose a model ✅	Linear Regression $\hat{y} = f_\theta(x) = x^T\theta$	Logistic Regression $\frac{1}{1+e^{-(\theta_0+\theta_1 x_1+\ldots+\theta_p x_p)}}$
2. Choose a loss function	Squared Loss or Absolute Loss	Can squared loss still work?
3. Fit the model	Sklearn	Sklearn
4. Evaluate model performance	R^2, Residuals, etc.	???

Pitfalls of Squared Loss

Regression vs. Classification

Deriving the Logistic Regression Model

- The Sigmoid (Logistic) Function

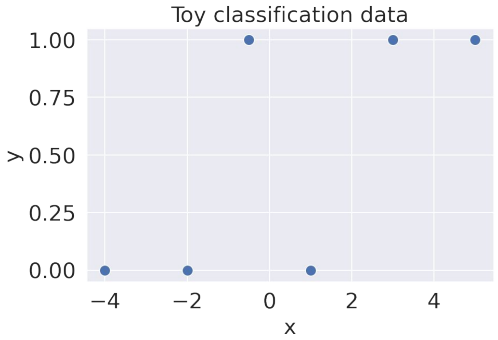
The Logistic Regression Model

- Comparison to Linear Regression

Parameter Estimation

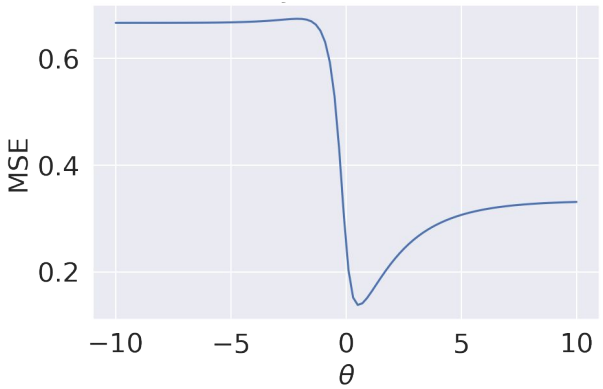
- **Pitfalls of Squared Loss**
- Cross-Entropy Loss
- Maximum Likelihood Estimation

Convexity Proof By Picture

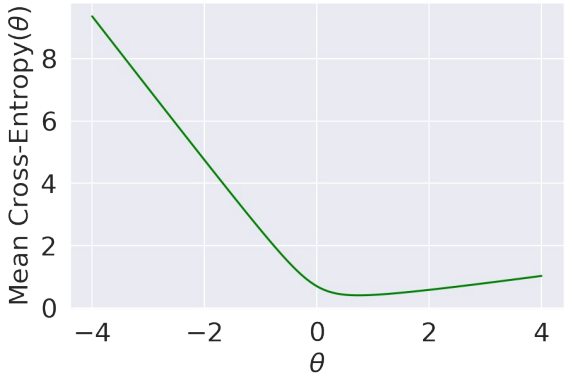


	x	y
0	-4.0	0
1	-2.0	0
2	-0.5	1
3	1.0	0
4	3.0	1
5	5.0	1

Squared Loss Surface

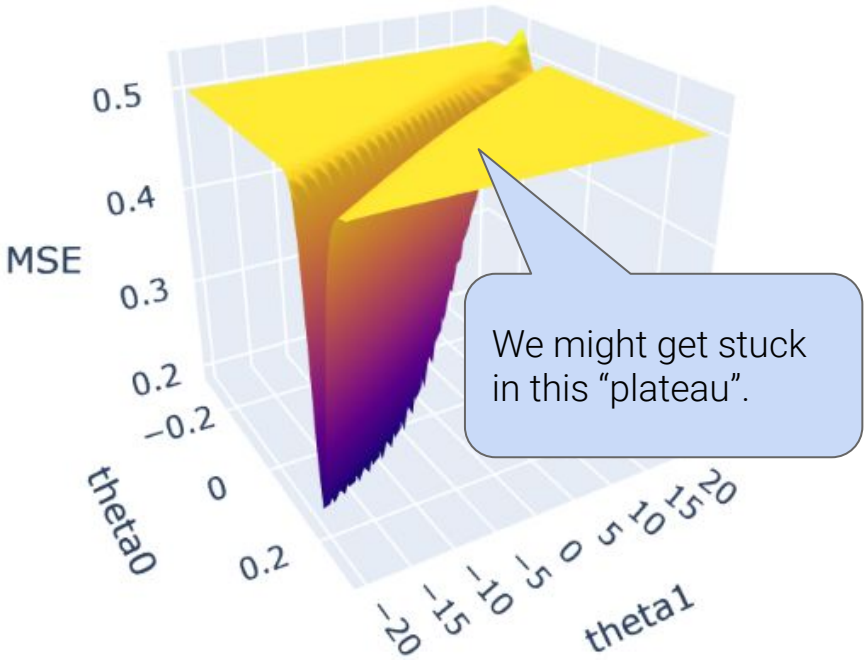


"BETTER" Loss Surface (Using Cross Entropy - stay tuned)

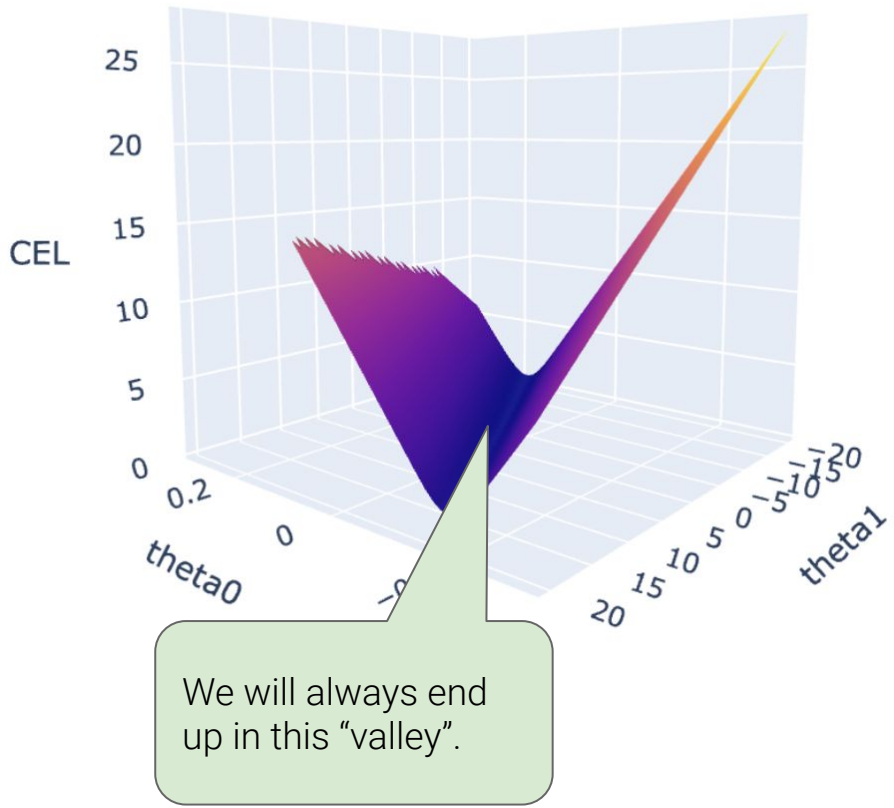


Convexity Proof By Picture (more features)

Squared Loss Surface



Cross-Entropy Loss Surface



A better loss function!

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

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

$$-\left(y\log(p)+(1-y)\log(1-p)\right)$$

Matches the probabilistic modeling of logistic regression if $\frac{1}{1+e^{-(\theta_0+\theta_1 x_1+\ldots+\theta_p x_p)}}$

Cross-Entropy loss addresses the 3 pitfalls of squared loss.

✅ Convex. No local minima for logistic regression.

✅ A good measure of model error. Strongly penalizes “off” predictions.

3. Conceptually sound - Based on Maximum Likelihood Estimation!

The Modeling Process

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

Shakespeare [Wikipedia]

Why Use Cross-Entropy Loss?

Two common explanations:

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

Maximizing Likelihood == Minimizing Average Cross-Entropy

maximize $p_1, p_2, \ldots, p_n$ $\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}$

Log is increasing; max/min properties

minimize $p_1, p_2, \ldots, p_n$ $-\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$

$p_i = \sigma(X_i^T\theta) = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1+\ldots+\theta_p x_p)}}$

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

Cross-Entropy Loss!!

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

- We are choosing the model parameters that are "most likely", given this data.

Assumption: all data drawn **independently** from the same logistic regression model with parameter θ

- 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.

[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.

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

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

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

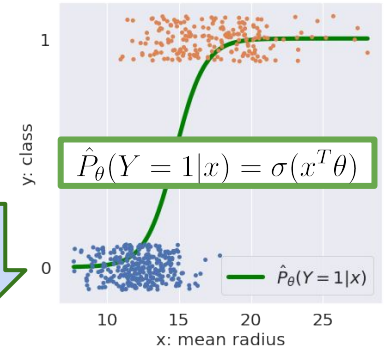
For logistic regression, let $p_i = \sigma(X_i^T\theta)$

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

Prob. that i-th response is yi

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**.

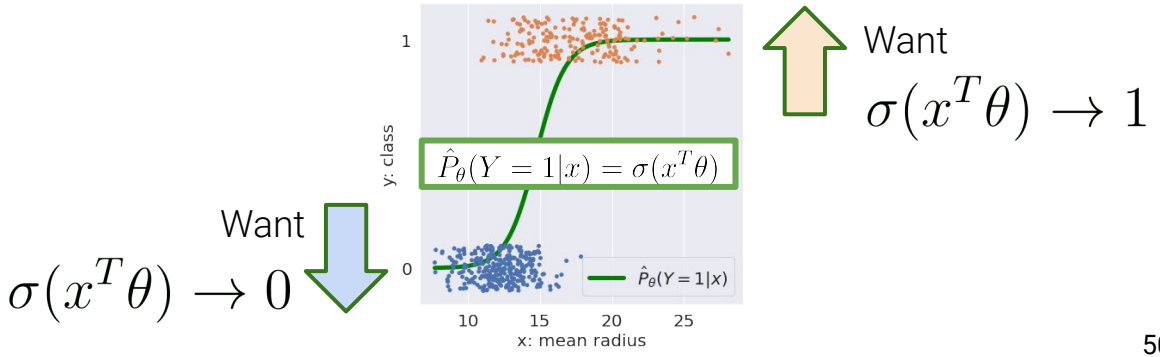
Assumption: all data are independent Bernoulli random variables

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

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

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

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



Modeling Process

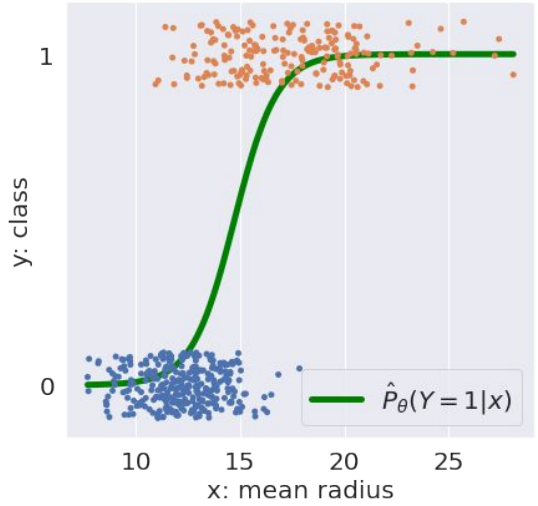
	Regression ($y \in \mathbb{R}$)	Classification ($y \in \{0, 1\}$)
		$\sigma(x^T\theta) = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1+\ldots+\theta_p x_p)}}$
1. Choose a model ✅	Linear Regression $\hat{y} = f_\theta(x) = x^T\theta$	Logistic Regression $\hat{P}_\theta(Y = 1\|x) = \sigma(x^T\theta)$
2. Choose a loss function	Squared Loss or Absolute Loss	Average Cross-Entropy Loss $-\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\sigma(X_i^T\theta) + (1-y_i)\log(1-\sigma(X_i^T\theta))\right)$
3. Fit the model	Regularization Sklearn/Gradient descent	Sklearn
4. Evaluate model performance	R^2, Residuals, etc.	Let's do it!

Demo

Logistic Regression with sklearn (documentation)

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

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



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

Demo

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)

Demo

See notebook for code snippets.

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…

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

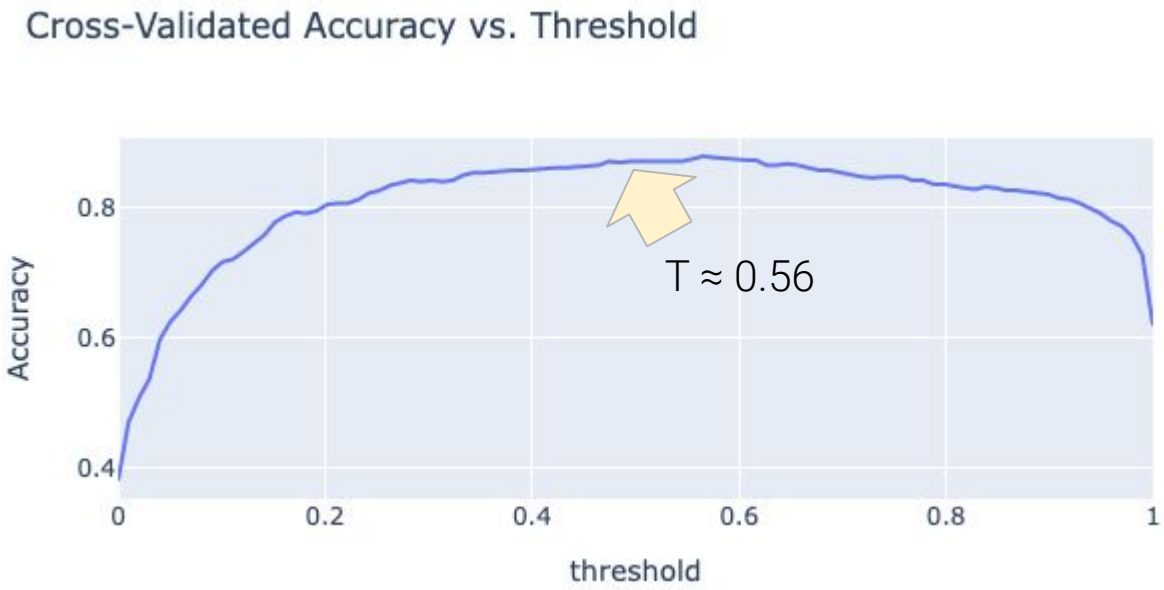
```
malignant
0          317
1          195
dtype: int64
```

Tune Thresholds with Cross Validation

The threshold should typically be tuned using cross validation.

For a threshold T:

$$\text{cross\_val\_acc} = (1/k)\left[\text{Model fit to train set 1, Acc on val set 1} + \ldots + \text{Model fit to train set k, Acc on val set k}\right]$$



Demo

See notebook for code snippets.
documentation

Pitfalls of Accuracy: A Case Study

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

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!!!

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)**.

See Appendix for more on these!

Demo

Appendix: Deriving The Logistic Regression Model

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

The `games` Dataset

New modeling task, new dataset.

1600133

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

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

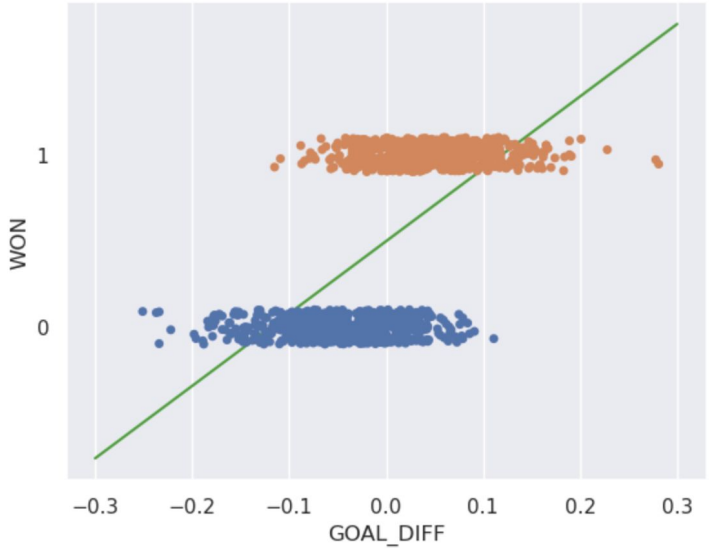
Difference in field goal success rate between teams

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

Why Not Least Squares Linear Regression?

1600133

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



Problems:

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

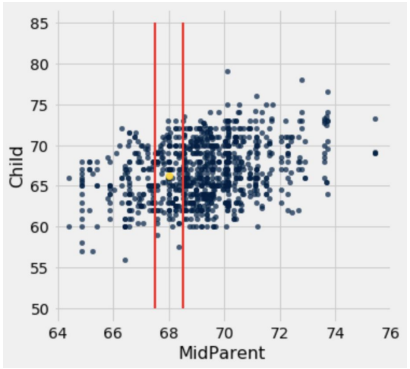
Demo

Back to Basics: the Graph of Averages

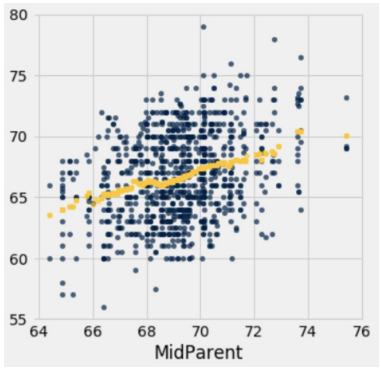
Clearly, least squares regression won't work here. We need to start fresh. 1600133

In Data 8, you built up to the idea of linear regression by considering the **graph of averages**.

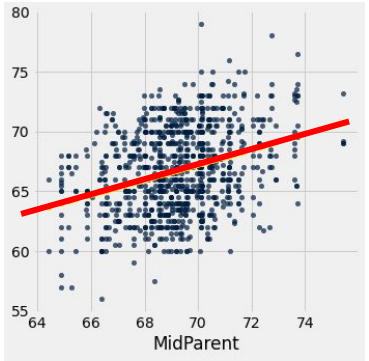
> For an input x, compute the **average value of y for all nearby x**, and predict that.
> [Data 8 textbook]



Bucket x-axis into bins



Average y vs. x bins is approximately linear



Parametric linear model

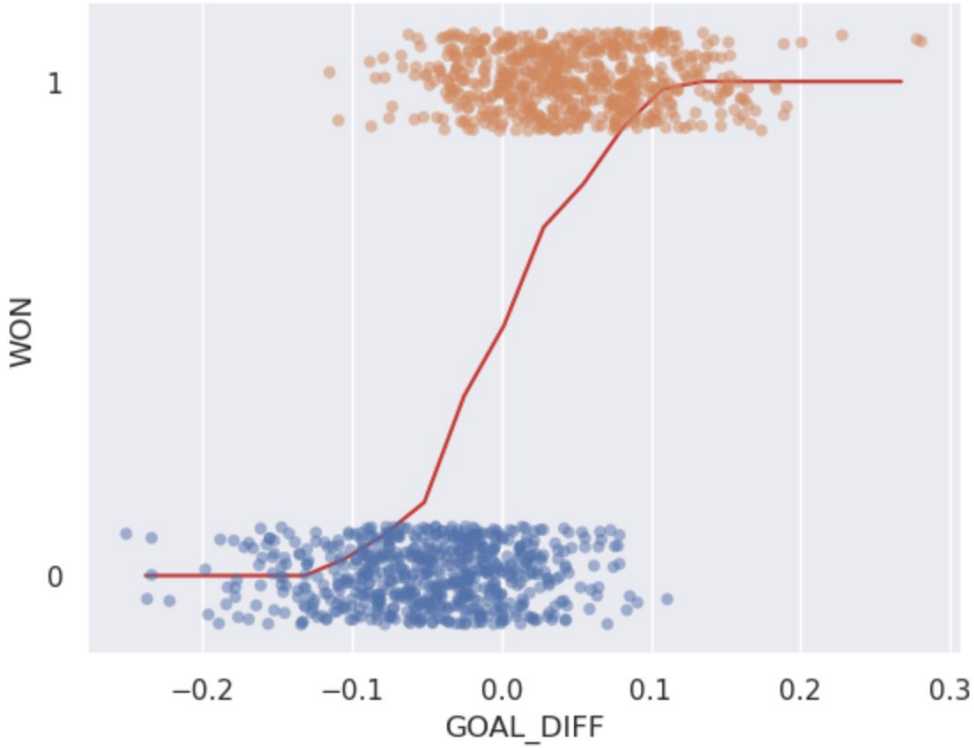
$$\hat{y} = x^T\theta$$

Graph of Averages for Our Classification Task

1600133

> For an input x, compute the **average value of y for all nearby x**, and predict that.

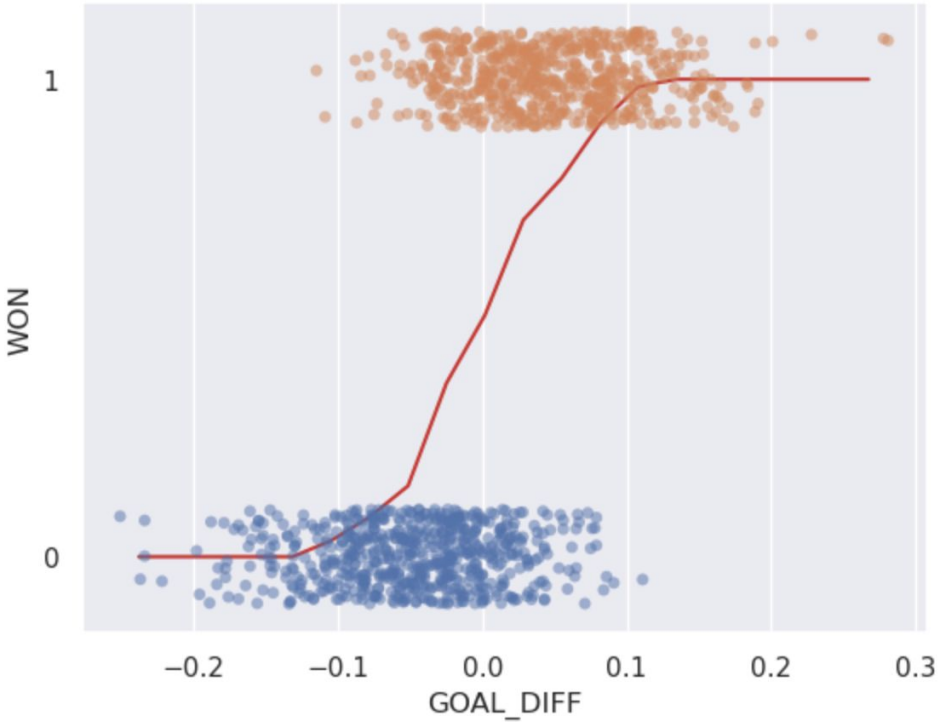
[Data 8 textbook]



Demo

Graph of Averages for Our Classification Task

1600133



Looking a lot better!

Some observations:

- All predictions are between 0 and 1.
- Our predictions are **non-linear.**
  - Specifically, we see an “S” shape.

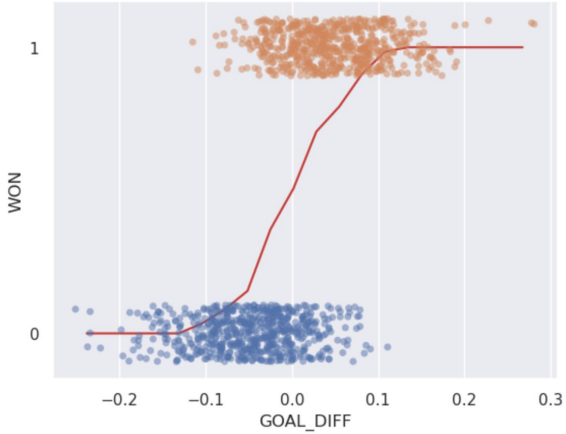
    This will be important soon.

Graph of Averages for Our Classification Task

Thinking more deeply about what we've just done.

To compute the average of a bin, we:

- Counted the number of wins in the bin.
- Divided by the total number of datapoints in the bin.



1600133

$$\frac{1(\#\ Y{=}1 \text{ in bin}) + 0(\#\ Y{=}0 \text{ in bin})}{\#\ \text{datapoints in bin}} = \frac{\#\ Y{=}1 \text{ in bin}}{\#\ \text{datapoints in bin}} = P(Y = 1 \mid \text{bin})$$

This is the probability that Y is 1 in that bin!

Predicting Probabilities

Our curve is modeling the *probability* that Y = 1 for a particular value of x.

1600133

$$p = P(Y = 1 \mid x)$$

- This matches our observation that all predictions are between 0 and 1 – the predictions are representing probabilities.

New modeling goal: model the *probability* of a datapoint belonging to Class 1.

Handling the Non-Linear Output

1600133

New modeling goal: model the *probability* of a datapoint belonging to Class 1.

We still seem to have a problem: our probability curve is non-linear, but we only know linear modeling techniques.

Good news: we've seen this before! To capture non-linear relationships, we:

1. Applied transformations to linearize the relationship.
2. Fit a linear model to the transformed variables.
3. Transformed the variables back to find their underlying relationship.

We'll use the exact same process here.

Step 1: Linearize the Relationship

Our S-shaped probability curve doesn't match any relationship we've seen before. The bulge diagram won't help here.

1600133

To understand what transformation to apply, think about our eventual goal: assign each datapoint to its *most likely* class.

How do we decide which class is more likely? One way: check which class has the higher predicted probability.

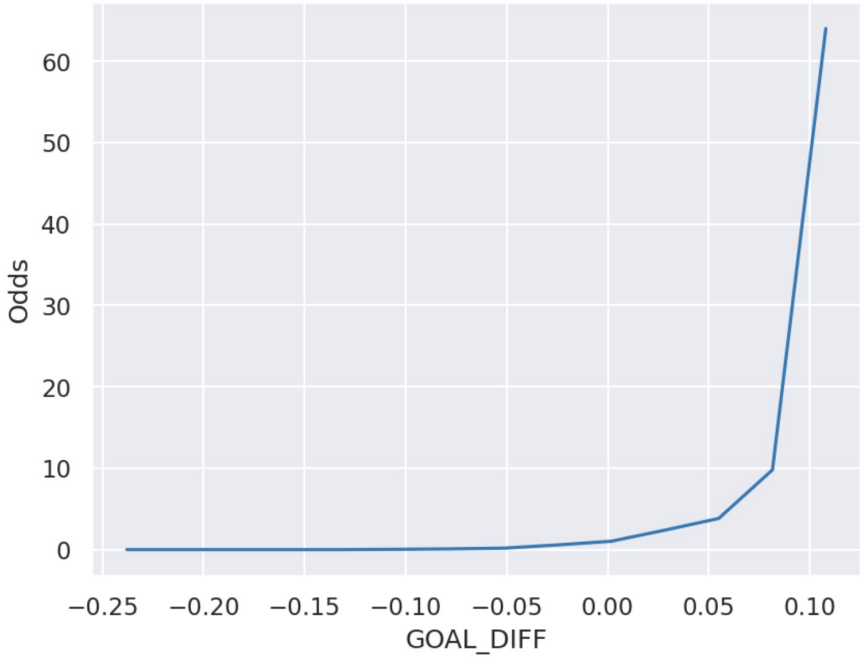
"Odds" is defined as the ratio of the probability of Y being Class 1 to the probability of Y being Class 0

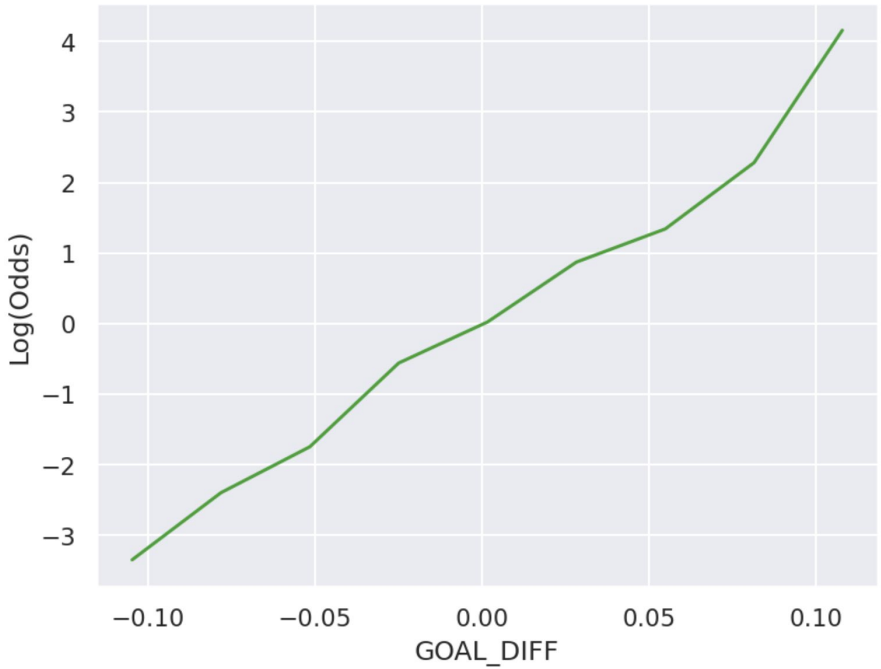
$$\text{odds} = \frac{P(Y=1 \mid x)}{P(Y=0 \mid x)} = \frac{p}{1-p}$$

Step 1: Linearize the Relationship

The odds curve looks roughly exponential. To linearize it, we'll take the logarithm*.

1600133



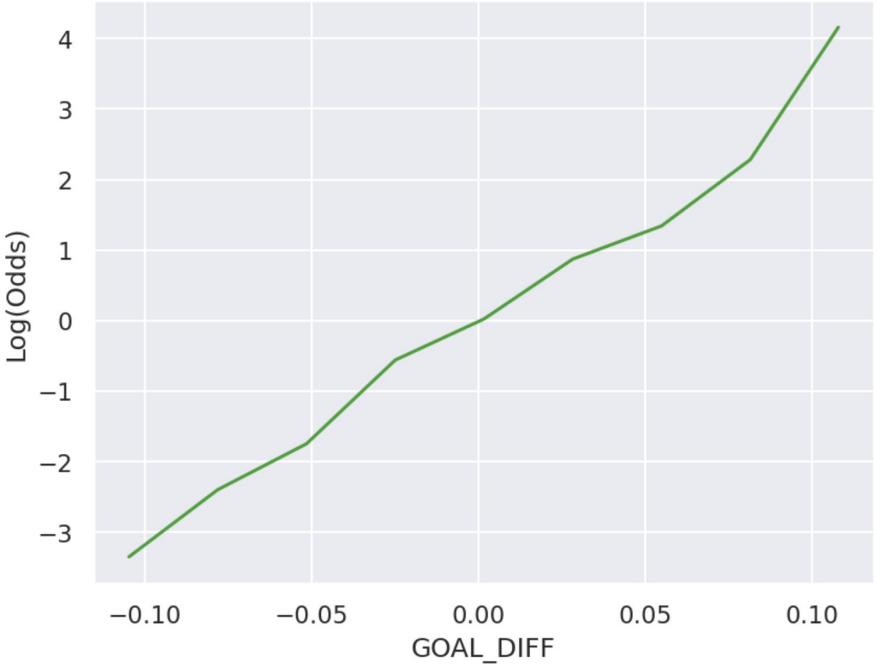


*In Data 100: assume that "log" means base e natural log unless told otherwise.

Step 2: Find a Linear Combination of the Features

We now have a linear relationship between our transformed variables. We can represent this relationship as a linear combination of the features.

1600133



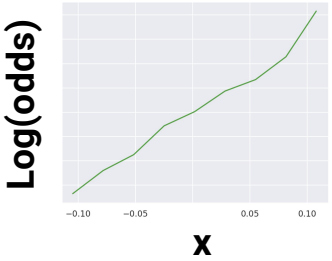
Our linear combination

$$z = x^\top \theta = \theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p$$

Remember that our goal is to predict the *probability* of Class 1. This linear combination isn't our final prediction! We use z instead of y_hat to remind ourselves.

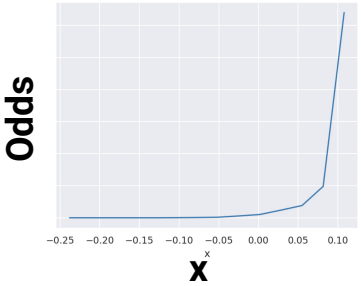
$$z = \log(\text{odds}) = \log\left(\frac{p}{1-p}\right)$$

Step 3: Transform Back to Original Variables

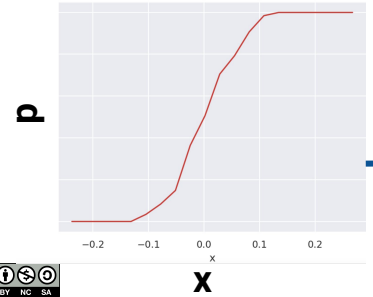


Solve for p:

$$\log\left(\frac{p}{1-p}\right) = x^T\theta$$



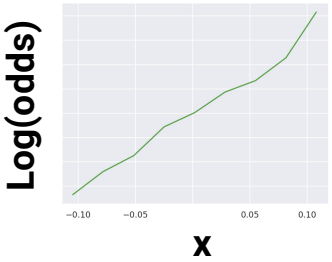
$$\frac{p}{1-p} = e^{x^T\theta}$$



$$p = \frac{1}{1+e^{-x^T\theta}}$$

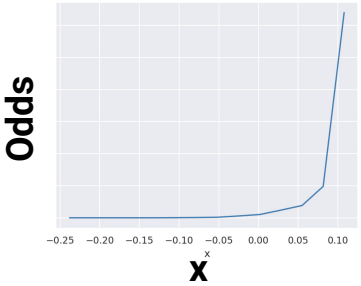
This is called the **logistic function**, $\sigma(\ )$.

Step 3: Transform Back to Original Variables



Solve for p:

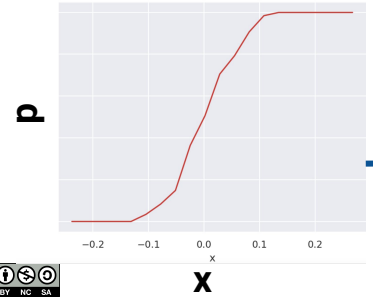
$$\log\left(\frac{p}{1-p}\right) = x^T\theta$$



$$\frac{p}{1-p} = e^{x^T\theta}$$

$$p = e^{x^T\theta} - pe^{x^T\theta}$$

$$p = \frac{e^{x^T\theta}}{1+e^{x^T\theta}}$$

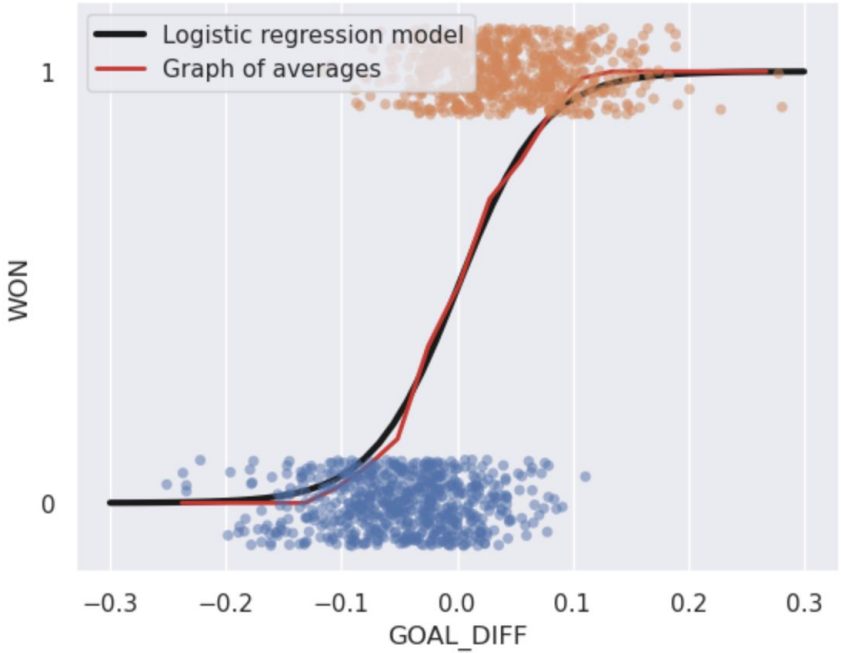


$$p = \frac{1}{1+e^{-x^T\theta}}$$

This is called the **logistic function**, $\sigma(\ )$.

Arriving at the Logistic Regression Model

We have just derived the **logistic regression model** for the probability of a datapoint belonging to Class 1. 1600133



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

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

To predict a probability:

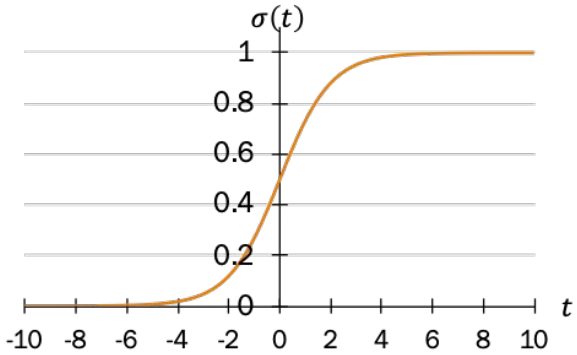
- Compute a linear combination of the features, $x^\top\theta$
- Apply the **sigmoid function** $\sigma(x^\top\theta)$

The Sigmoid Function

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

1600133

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



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

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

Domain $-\infty < t < \infty$

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

Derivative

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

The Sigmoid Converts Numerical Features to Probabilities

GAME_ID	TEAM_NAME	MATCHUP	REB	FTM	TOV	GOAL_DIFF	WON
21700001	Boston Celtics	BOS @ CLE	46	19	12	-0.049	0
21700002	Golden State Warriors	GSW vs. HOU	41	19	17	0.053	0
21700003	Charlotte Hornets	CHA @ DET	47	23	17	-0.030	0
21700004	Indiana Pacers	IND vs. BKN	47	25	14	0.041	1
21700005	Orlando Magic	ORL vs. MIA	50	22	15	0.042	1

Input: numeric features

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

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

1600133

Next lecture:

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

Decision rule

Output: **class**

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

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

Formalizing the Logistic Regression Model

Our main takeaways of this section:

1600133

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

Putting it all together:

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

The logistic regression model is most commonly written as follows:

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

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

Example Calculation

Suppose I want to predict the probability that a team wins a game, given **GOAL_DIFF** (first feature) and **number of free throws** (second feature). 1600133

I fit a logistic regression model (with no intercept) using my training data, and estimate the optimal parameters:

$$\hat{\theta}^T = [0.1 \quad -0.5]$$

Now, you want to predict the probability that a new team wins their game.

$$x^T = [15 \quad 1]$$

Example Calculation

Suppose I want to predict the probability that a team wins a game, given **GOAL_DIFF** (first feature) and **number of free throws** (second feature). 1600133

I fit a logistic regression model (with no intercept) using my training data, and estimate the optimal parameters:

$$\hat{\theta}^T = [0.1 \quad -0.5]$$

Now, you want to predict the probability that a new team wins their game.

$$x^T = [15 \quad 1]$$

$$\begin{aligned}\hat{P}_{\hat{\theta}}(Y = 1|x) &= \sigma(x^T\hat{\theta}) \\ &= \sigma(0.1 \cdot 15 + (-0.5) \cdot 1) \\ &= \sigma(1) \\ &= \frac{1}{1 + e^{-1}} \\ &\approx 0.7311\end{aligned}$$

Because the response is more likely to be 1 than 0, a reasonable prediction is

$$\hat{y} = 1$$

(more on this next lecture)

Appendix: Issues with MSE

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

Toy Dataset: L2 Loss

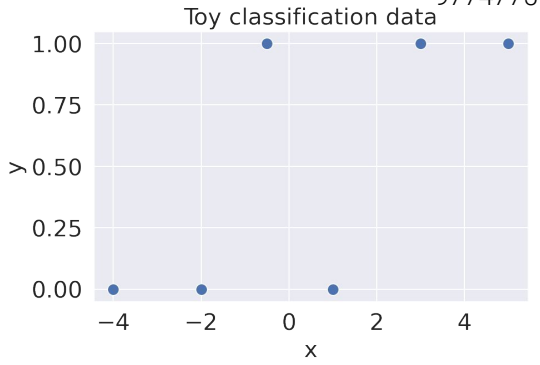
9774775

Logistic Regression model:

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

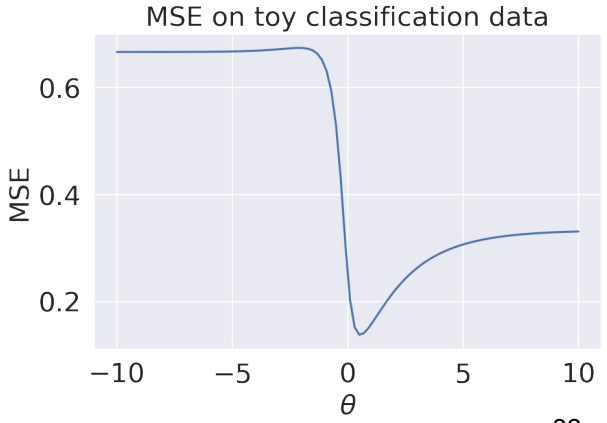
Assume no intercept.
So x, θ both scalars.

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



Mean Squared Error:

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



The MSE loss surface for logistic regression has many issues!

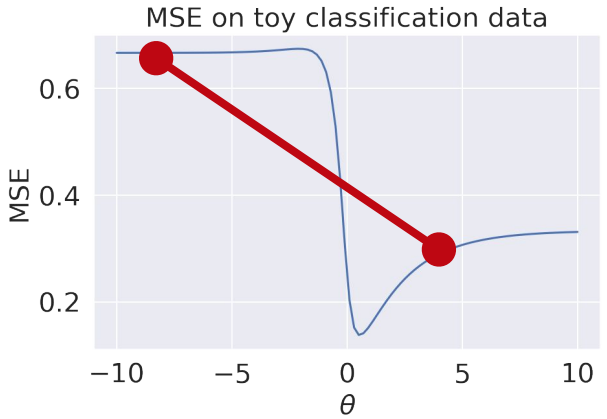
Demo

3 Pitfalls of Squared Loss

9774775

1. Non-convex. Gets stuck in local minima.

Secant line crosses function, so $R''(\theta)$ is not greater than 0 for all θ.



Demo

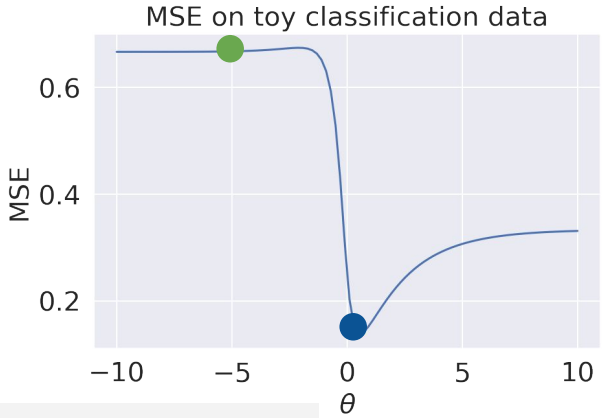
3 Pitfalls of Squared Loss

9774775

1. Non-convex. Gets stuck in local minima.

Secant line crosses function, so $R''(\theta)$ is not greater than 0 for all θ.

Gradient Descent: Different initial guesses will yield different optimal estimates.




```
from scipy.optimize import minimize

minimize(mse_loss_toy_nobias, x0 = 0)["x"][0]
```

```
0.5446601825581691
```

```
minimize(mse_loss_toy_nobias, x0 = -5)["x"][0]
```

```
-10.343653061026611
```

Demo

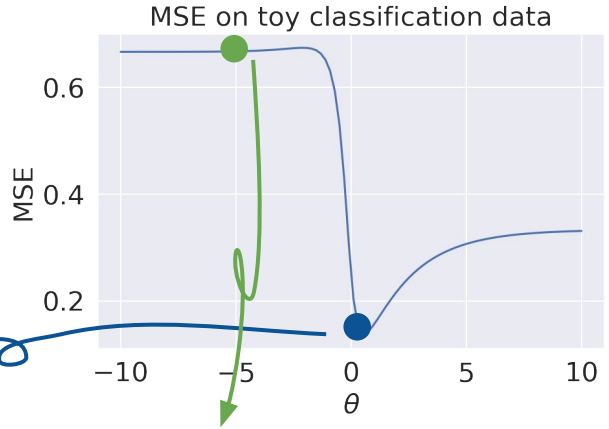
3 Pitfalls of Squared Loss

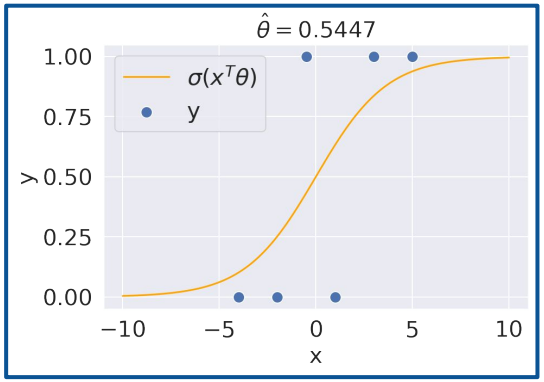
9774775

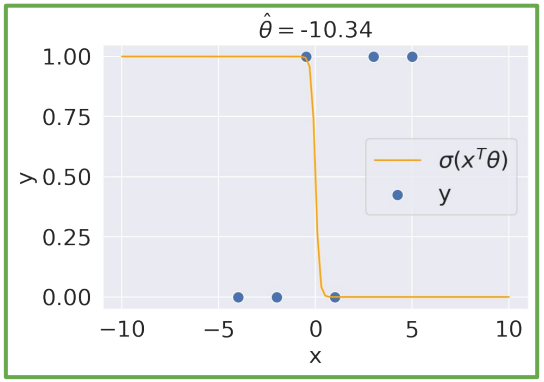
1. Non-convex. Gets stuck in local minima.

Secant line crosses function, so $R''(\theta)$ is not greater than 0 for all θ.

Gradient Descent: Different initial guesses will yield different optimal estimates.







Demo

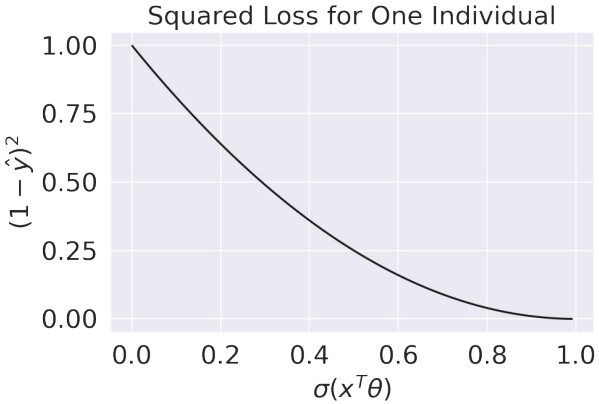
3 Pitfalls of Squared Loss

9774775

1. **Non-convex**. Gets stuck in local minima.

2. **Bounded**. Not a good measure of model error.

- We'd like loss functions to penalize "off" predictions.
- MSE never gets very large, because both response and predicted probability are bounded by 1.



Demo

3 Big Pitfalls of Squared Loss

9774775

1. **Non-convex**. Gets stuck in local minima.

2. **Bounded**. Not a good measure of model error.

3. **Conceptually questionable**.
   Tries to match probability to 0/1 class labels.

shutterstock.com · 406338748

MSE + classification is *occasionally* used in some neural network applications.
But overall, avoid.

Demo

Appendix: Model Evaluation

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

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.

Actual y \ Prediction $\hat{y}$	**0**	**1**
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**.

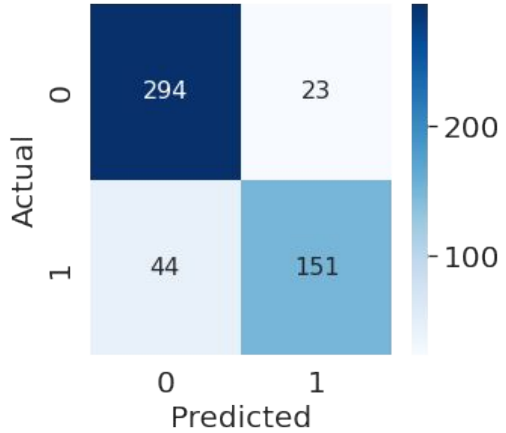
Types of Classification Successes/Errors: The Confusion Matrix

- **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.

Actual y \ Prediction $\hat{y}$	**0**	**1**
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.

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



Accuracy, Precision, and Recall

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

What proportion of points did our classifier classify correctly?

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

Accuracy, Precision, and Recall

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

What proportion of points did our classifier classify correctly?

Actual \ Prediction	0	1
0	TN	FP
1	FN	TP

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

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

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

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

Accuracy, Precision, and Recall

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

What proportion of points did our classifier classify correctly?

Actual \ Prediction	0	1
0	TN	FP
1	FN	TP

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

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

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

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

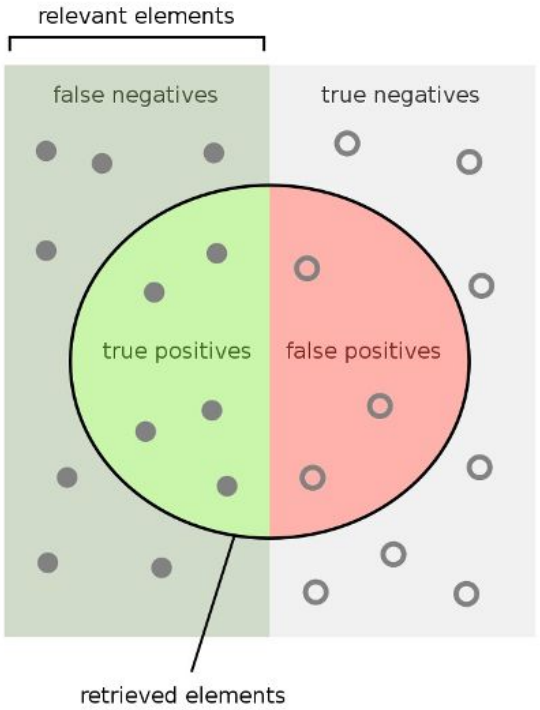
$$\text{recall} = \frac{TP}{TP + FN}$$

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

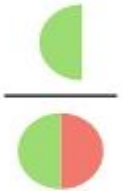
- How **sensitive** is our classifier to **positives**?
- Penalizes false negatives.

One of the Most Valuable Graphics on Wikipedia

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



How many retrieved items are relevant?

Precision = 

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

How many relevant items are retrieved?

Recall = 

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

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

[adapted from Wikipedia]

Back to the Spam

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

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

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

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

Your friend:

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

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

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

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

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

Back to the Spam

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

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

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

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

Your friend:

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

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

Never positive!

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

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

Your other friend ("Friend 2"):

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

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

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

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

Many false positives!

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

No false negatives!

Precision vs. Recall

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

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

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?

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.

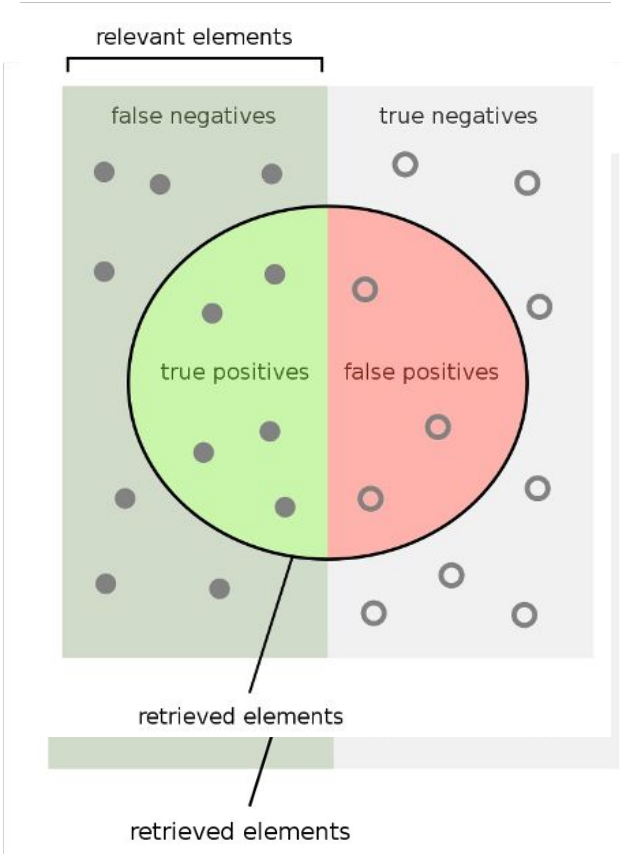
$$\text{FPR} = \frac{FP}{FP + TN}$$

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

Demo

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

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



How many retrieved items are relevant?

Precision =

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

Want close to 1.0

How many relevant items are retrieved?

Recall =

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

TPR

Want close to 1.0

How many irrelevant items are retrieved?

FPR =

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

Want close to 0.0

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

[adapted from Wikipedia]

Demo

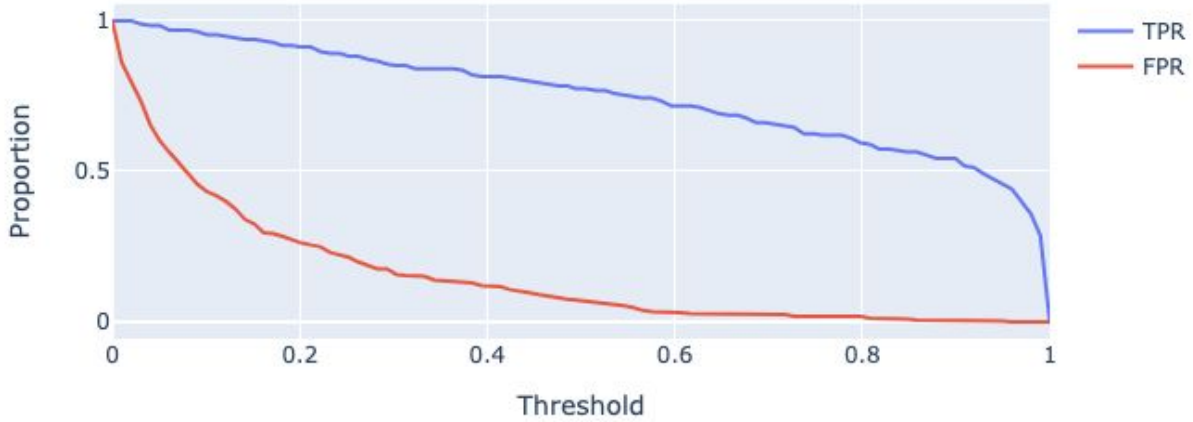
Not the ROC Curve, but Useful to Start with

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.

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

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



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).

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?

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.

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.