# Logistic Regression

## Background: Generative and Discriminative Classifiers

# Logistic Regression

Important analytic tool in natural and social sciences

Baseline supervised machine learning tool for classification

Is also the foundation of neural networks

# Generative and Discriminative Classifiers

Naive Bayes is a **generative** classifier

by contrast:

Logistic regression is a **discriminative** classifier

# Generative and Discriminative Classifiers

Suppose we're distinguishing cat from dog images

imagenet

imagenet

# Generative Classifier:

- Build a model of what's in a cat image
  - Knows about whiskers, ears, eyes
  - Assigns a probability to any image:
    - how cat-y is this image?

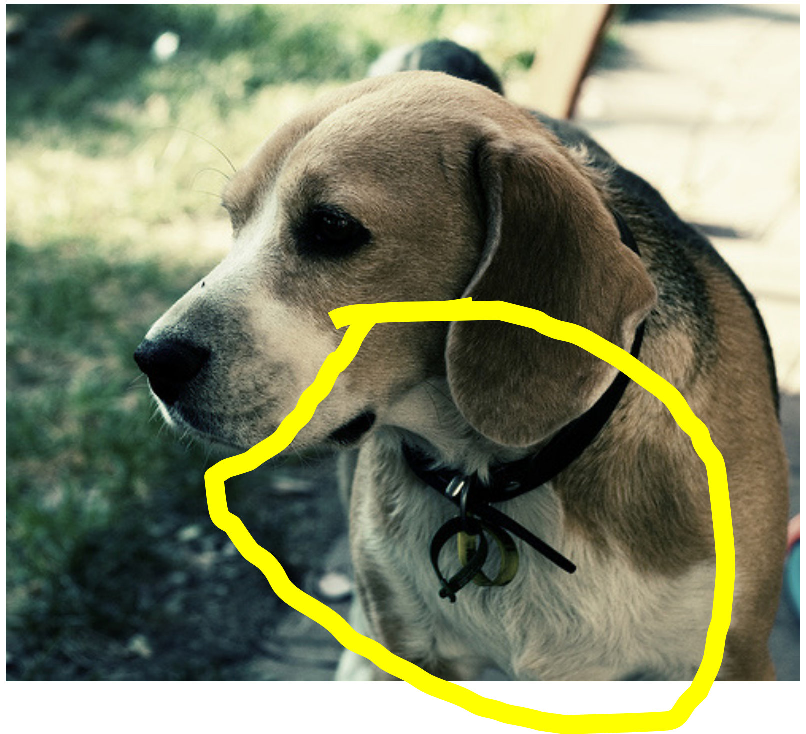
Also build a model for dog images

Now given a new image:

**Run both models and see which one fits better**

# Discriminative Classifier

Just try to distinguish dogs from cats



Oh look, dogs have collars!

Let's ignore everything else

# Finding the correct class c from a document d in Generative vs Discriminative Classifiers

## Naive Bayes

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \; \overbrace{P(d|c)}^{\text{likelihood}} \; \overbrace{P(c)}^{\text{prior}}$$

## Logistic Regression

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \; \overset{\text{posterior}}{P(c|d)}$$

# Components of a probabilistic machine learning classifier

Given *m* input/output pairs $(x^{(i)},y^{(i)})$:

1. A **feature representation** of the input. For each input observation $x^{(i)}$, a vector of features $[x_1, x_2, ... , x_n]$. Feature $j$ for input $x^{(i)}$ is $\mathrm{x}_\mathrm{j}$, more completely $x_j^{(i)}$, or sometimes $f_j(x)$.
2. A **classification function** that computes $\hat{y}$, the estimated class, via $p(y|x)$, like the **sigmoid** or **softmax** functions.
3. An objective function for learning, like **cross-entropy loss**.
4. An algorithm for optimizing the objective function: **stochastic gradient descent**.

# The two phases of logistic regression

**Training**: we learn weights *w* and *b* using **stochastic gradient descent** and **cross-entropy loss**.

**Test**: Given a test example *x* we compute $p(y|x)$ using learned weights *w* and *b*, and return whichever label ($y = 1$ or $y = 0$) is higher probability

# Logistic Regression

## Background: Generative and Discriminative Classifiers

Logistic Regression

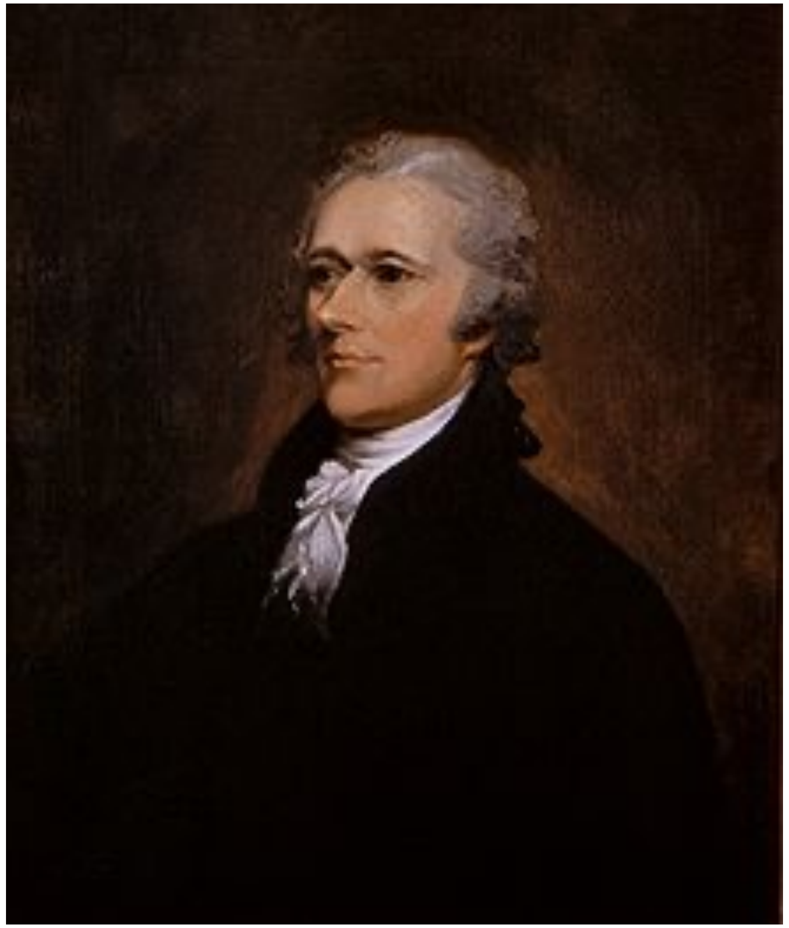
# Classification in Logistic Regression

# Classification Reminder

Positive/negative sentiment

Spam/not spam

Authorship attribution (Hamilton or Madison?)



Alexander Hamilton

# Text Classification: definition

*Input*:

- a document $x$
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

*Output*: a predicted class $\hat{y} \in C$

# Binary Classification in Logistic Regression

Given a series of input/output pairs:

- $(x^{(i)}, y^{(i)})$

For each observation $x^{(i)}$

- We represent $x^{(i)}$ by a **feature vector** $[x_1, x_2, \ldots, x_n]$
- We compute an output: a predicted class $\hat{y}^{(i)} \in \{0,1\}$

# Features in logistic regression

- For feature $x_i$, weight $w_i$ tells is how important is $x_i$
  - $x_i$ ="review contains 'awesome'": $w_i = +10$
  - $x_j$ ="review contains 'abysmal'": $w_j = -10$
  - $x_k$ ="review contains 'mediocre'": $w_k = -2$

# Logistic Regression for one observation x

Input observation: vector $x = [x_1, x_2,..., x_n]$

Weights: one per feature: $W = [w_1, w_2,..., w_n]$

- Sometimes we call the weights $\theta = [\theta_1, \theta_2,..., \theta_n]$

Output: a predicted class $\hat{y} \in \{0,1\}$

(multinomial logistic regression: $\hat{y} \in \{0, 1, 2, 3, 4\}$)

# How to do classification

For each feature $x_i$, weight $w_i$ tells us importance of $x_i$

- (Plus we'll have a bias b)

We'll sum up all the weighted features and the bias

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

$$z = w \cdot x + b$$

If this sum is high, we say y=1; if low, then y=0

# But we want a probabilistic classifier

We need to formalize “sum is high”.

We’d like a principled classifier that gives us a probability, just like Naive Bayes did

We want a model that can tell us:

$p(y=1|x; \theta)$

$p(y=0|x; \theta)$

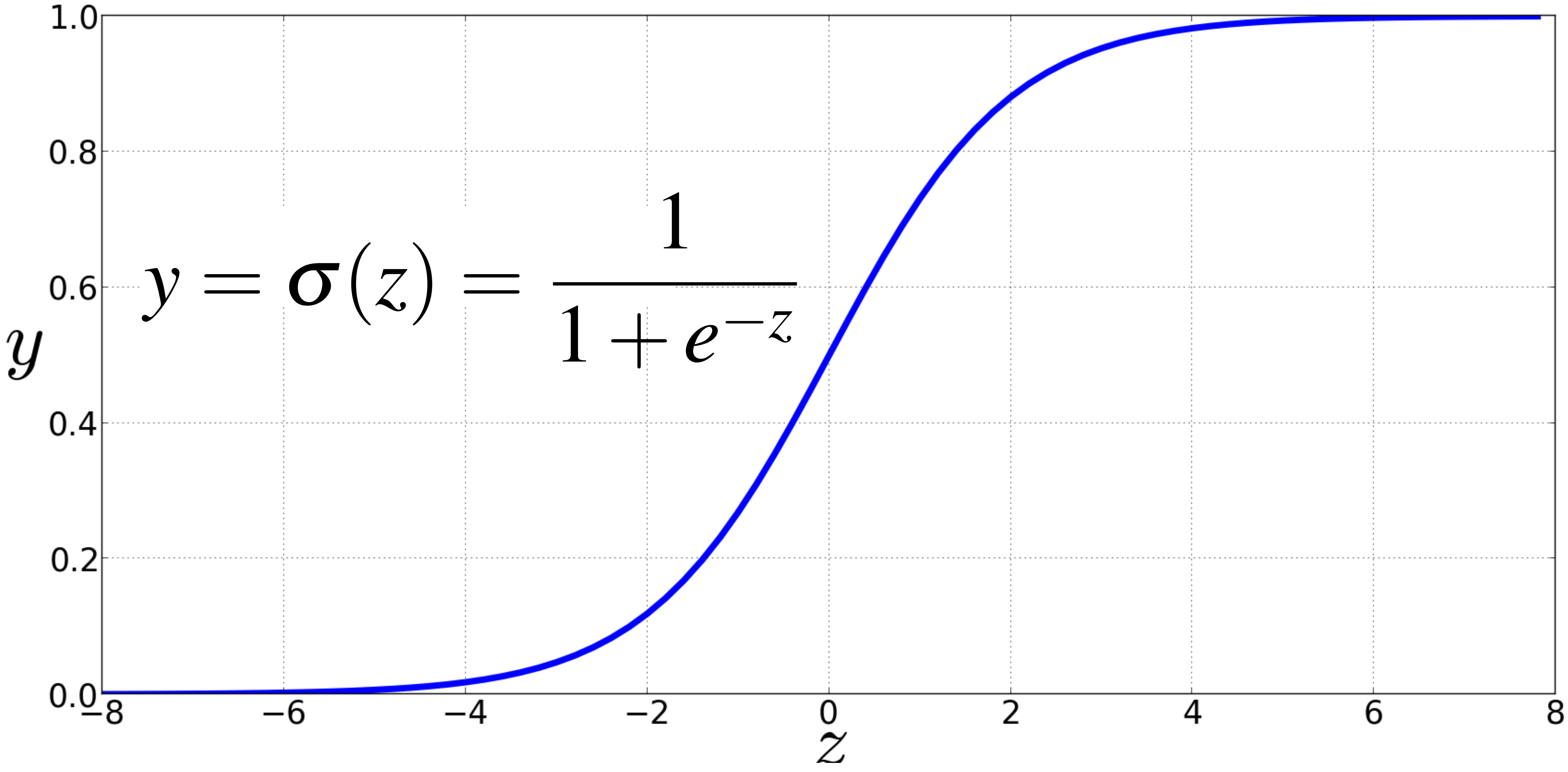
# The problem: z isn't a probability, it's just a number!

$$z = w \cdot x + b$$

Solution: use a function of z that goes from 0 to 1

$$y = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-z)}$$

# The very useful sigmoid or logistic function



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

# Idea of logistic regression

We’ll compute $w{\cdot}x+b$

And then we’ll pass it through the sigmoid function:

$$\sigma(w{\cdot}x+b)$$

And we'll just treat it as a probability

# Making probabilities with sigmoids

$$
\begin{aligned}
P(y=1) &= \sigma(w\cdot x+b) \\
&= \frac{1}{1+\exp\left(-(w\cdot x+b)\right)}
\end{aligned}
$$

$$
\begin{aligned}
P(y=0) &= 1-\sigma(w\cdot x+b) \\
&= 1-\frac{1}{1+\exp\left(-(w\cdot x+b)\right)} \\
&= \frac{\exp\left(-(w\cdot x+b)\right)}{1+\exp\left(-(w\cdot x+b)\right)}
\end{aligned}
$$

# By the way:

$$P(y=0) = 1-\sigma(w\cdot x+b) \qquad = \sigma(-(w\cdot x+b))$$

$$= 1-\frac{1}{1+\exp(-(w\cdot x+b))}$$

Because

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

$$= \frac{\exp(-(w\cdot x+b))}{1+\exp(-(w\cdot x+b))}$$

# Turning a probability into a classifier

$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

0.5 here is called the **decision boundary**