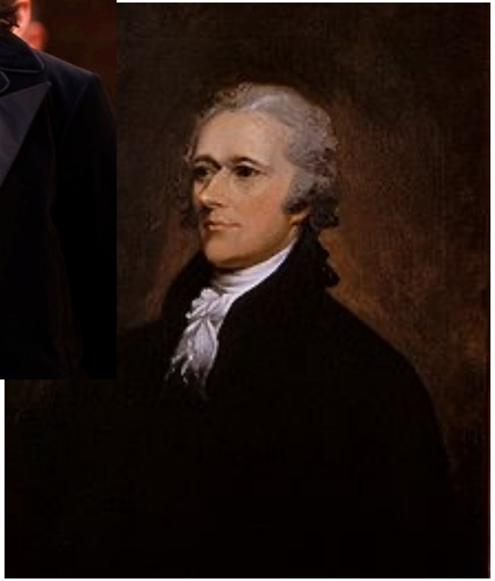
Classification in Logistic Regression

Logistic Regression

- Important analytic tool in natural and social sciences
- Baseline supervised machine learning tool for classification
- Is also the foundation of a neural network

Classification Reminder

- Positive/negative sentiment
- Spam/not spam
- Authorship attribution (Hamilton or Madison?)


Alexander Hamilton

Text Classification: definition

- *Input*:
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

- *Output*: a predicted class $c \in C$

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 $\hat{y}^i \in \{0,1\}$

Examples of inputs and features: spam

- X is an email
  - X_1 is “count of the word *sale* in the email
  - X_2 is “count of mentions of drugs”
  - X_3 is “number of URLS in the email”
  - X_4 is “length of email in bytes”
  - X_5 is “uses phrase ‘Prestigious Non-Accredited Universities’

Features in logistic regression

- For each feature x_i, we'll have a weight w_i
- Weight w_i tells us how important feature x_i is the classification

  x_i =”1 if review contains ‘awesome’": w_i very positive +10

  x_j =”1 if review contains ‘abysmal’": w_j very negative -10

  x_k =“1 if review contains ‘mediocre’": w_k a little negative -2

Logistic Regression for one observation x

- Input observation: vector $x = [x_1, x_2, \ldots, x_n]$
- Weights: one per feature: $W = [w_1, w_2, \ldots, w_n]$
    - Sometimes we call the weights $\theta = [\theta_1, \theta_2, \ldots, \theta_n]$
- Output: a predicted class $\hat{y} \in \{0,1\}$

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

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:

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

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

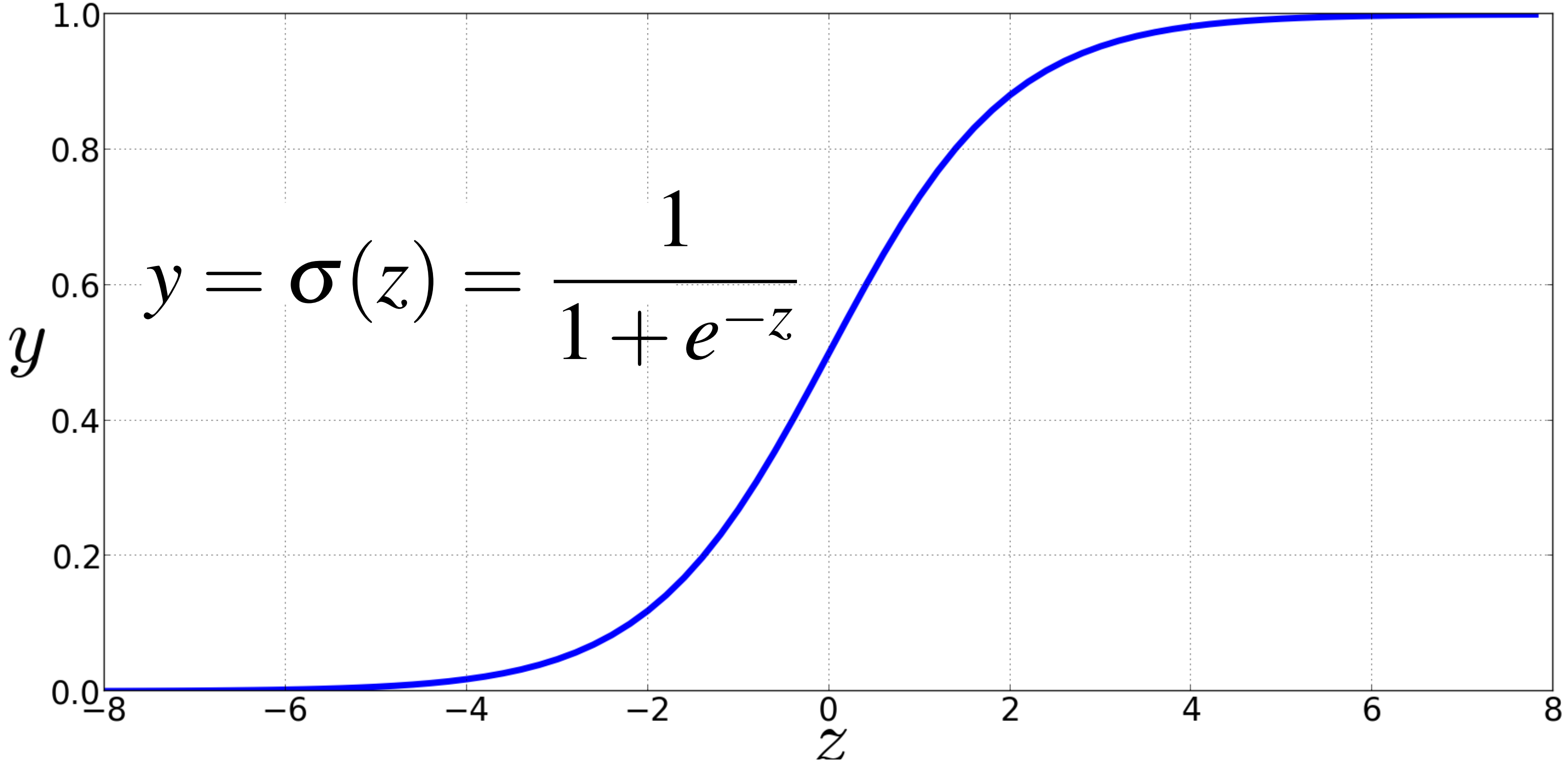
The problem: z isn't a probability, it's just a number!

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

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

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

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

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

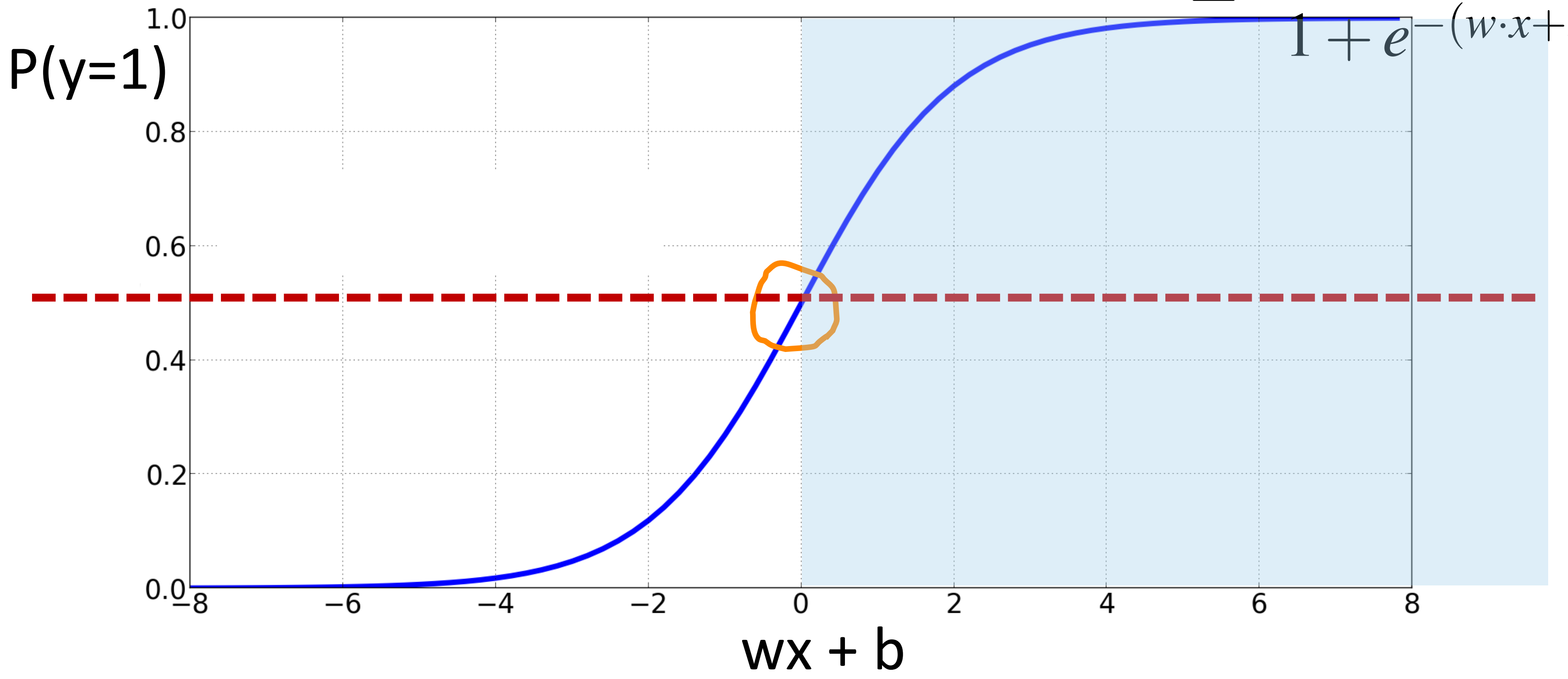
$$
\begin{aligned}
P(y=0) &= 1-\sigma(w\cdot x+b) \\
&= 1-\frac{1}{1+e^{-(w\cdot x+b)}} \\
&= \frac{e^{-(w\cdot x+b)}}{1+e^{-(w\cdot x+b)}}
\end{aligned}
$$

Turning a probability into a classifier: the decision boundary

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

The probabilistic classifier

$$P(y=1) = \sigma(w \cdot x + b)$$
$$= \frac{1}{1+e^{-(w \cdot x+b)}}$$



Logistic Regression: A text example

Sentiment example: does $y=1$ or $y=0$?

It's hokey, there are virtually no surprises, and the writing is second-rate. So why was it so enjoyable? For one thing, the cast is great. Another nice touch is the music. I was overcome with the urge to get off the couch and start dancing. It sucked me in, and it'll do the same to you.

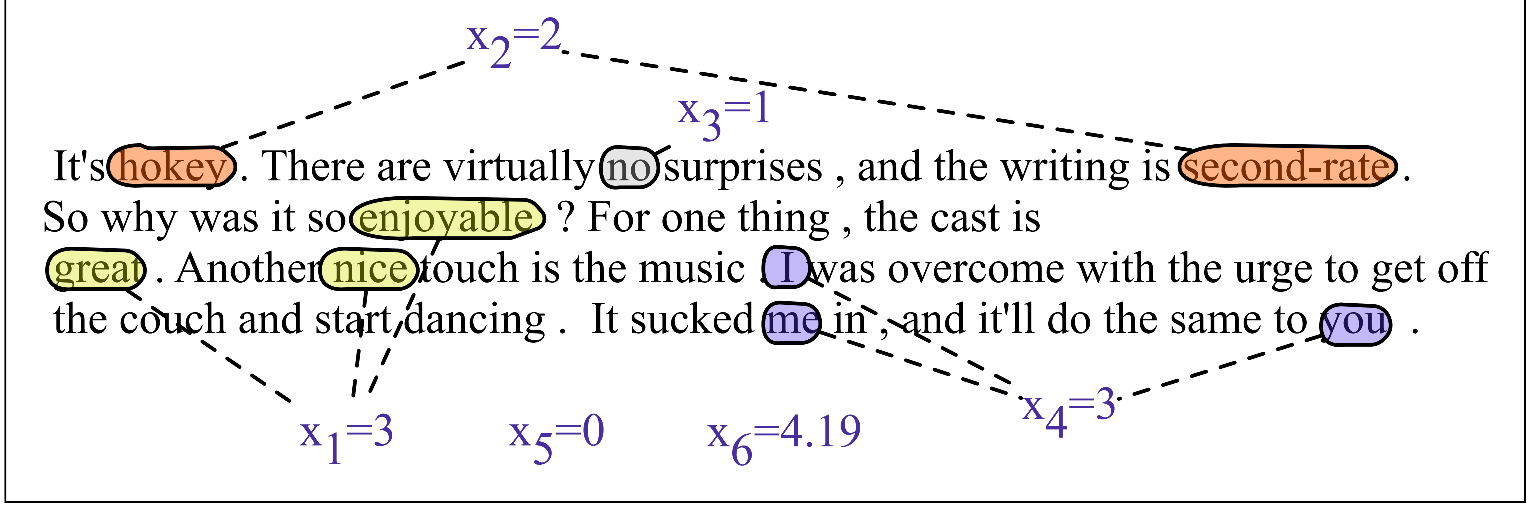
It's hokey, there are virtually no surprises, and the writing is second-rate. So why was it so enjoyable? For one thing, the cast is great. Another nice touch is the music. I was overcome with the urge to get off the couch and start dancing. It sucked me in, and it'll do the same to you.

Positive sentiment words

Negative sentiment words

"no"

i/me/mine/you/your/yours


x2=2
x3=1
It's hokey . There are virtually no surprises , and the writing is second-rate .
So why was it so enjoyable ? For one thing , the cast is
great . Another nice touch is the music . I was overcome with the urge to get off
the couch and start dancing . It sucked me in , and it'll do the same to you .
x1=3
x5=0
x6=4.19
x4=3

Var	Definition	Value in Fig. 5.2
x_1	$\text{count}(\text{positive lexicon}) \in \text{doc})$	3
x_2	$\text{count}(\text{negative lexicon}) \in \text{doc})$	2
x_3	$\begin{cases} 1 & \text{if “no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	$\text{count}(\text{1st and 2nd pronouns} \in \text{doc})$	3
x_5	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	$\log(\text{word count of doc})$	$\ln(66) = 4.19$

Classifying sentiment for input x

Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if ``no''} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if ``!''} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(66) = 4.19$

Suppose $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$

$b = 0.1$

Classifying sentiment for input x

$$\begin{aligned} p(+|x) = P(Y=1|x) &= \sigma(w \cdot x + b) \\ &= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1) \\ &= \sigma(.833) \\ &= 0.70 \end{aligned} \tag{5.6}$$

$$\begin{aligned} p(-|x) = P(Y=0|x) &= 1 - \sigma(w \cdot x + b) \\ &= 0.30 \end{aligned}$$

We can build features for logistic regression for any classification task: period disambiguation

End of sentence

This ends in a period.
The house at 465 Main St. is new.

Not end

$$x_1 = \begin{cases} 1 & \text{if ``}Case(w_i) = \text{Lower''} \\ 0 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if ``}w_i \in \text{AcronymDict''} \\ 0 & \text{otherwise} \end{cases}$$

$$x_3 = \begin{cases} 1 & \text{if ``}w_i = \text{St. \& } Case(w_{i-1}) = \text{Cap''} \\ 0 & \text{otherwise} \end{cases}$$

Logistic Regression: The Whole Picture

Classification in (binary) logistic regression: summary

Given:

- a set of classes: (+ sentiment,- sentiment)
- a vector **x** of features `[x1, x2, …, xn]`
    - x1= count("awesome")
    - x2 = log(number of words in review)
- A vector **w** of weights `[w1, w2, …, wn]`
    - w_i for each feature f_i

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

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

Wait, where did the W's come from?

Training: given a training set of M observations (example x, class y)

- Learn the parameters of the model

Test: Given a test example *x* and class $y \in \{0,1\}$

- return the higher probability class

Components of a probabilistic (supervised) machine learning classifier

A **corpus** of M observation input/output pairs, $(x^{(i)},y^{(i)})$

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

- a vector of **features** $[x_1, x_2, ..., x_n]$

A **classification function** computing $\hat{y}$, via $p(y|x)$

- *sigmoid*
- *softmax*

For learning

- A **loss function** (cross-entropy loss)
- An **optimization algorithm** (stochastic gradient descent)

Learning in Logistic Regression

- How are parameters of the model (w and b) learned?

- This is an instance of supervised learning
  - We have labeled training examples

- We want model parameters such that
  - For training examples x
  - The prediction of the model $\hat{y}$
  - is as close as possible to the true y

Learning in Logistic Regression

- How are parameters of the model (w and b) learned?

- This is an instance of supervised learning
  - We have labeled training examples

- We want model parameters such that
  - For training examples x, the prediction of the model $\hat{y}$ is as close as possible to the true y
  - Or equivalently so that the distance between $\hat{y}$ and y is small

Ingredients required for training

- Loss function or cost function
  - A measure of distance between classifier prediction and true label for a given set of parameters

$$L(y, \hat{y}) = How\ much\ \hat{y}\ differs\ from\ the\ true\ y$$

- An algorithm to minimize this loss
  - Here we'll introduce stochastic gradient descent

Cross-entropy loss function

Maximize $L(y, \hat{y}) = \log p(y|x) = \log[\hat{y}^y (1 - \hat{y}^{1-y})]$

Negative log-likelihood

Minimize $L_{CE}(w, \hat{b}) = -[y \log \sigma(w \cdot x + b) +$
$(1 - y) \log(1 - (w \cdot x + b))]$

Gradient Descent

- Goal:
  - find parameters $\theta = w, b$
  - Such that

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{CE}(y^{(i)}, x^{(i)}; \theta)$$

- For logistic regression, the loss is **convex**

Gradient Descent

- The gradient indicates the direction of greatest increase of the cost/loss function.

- Gradient descent finds parameters (w,b) that decrease the loss by taking a step in the opposite direction of the gradient.

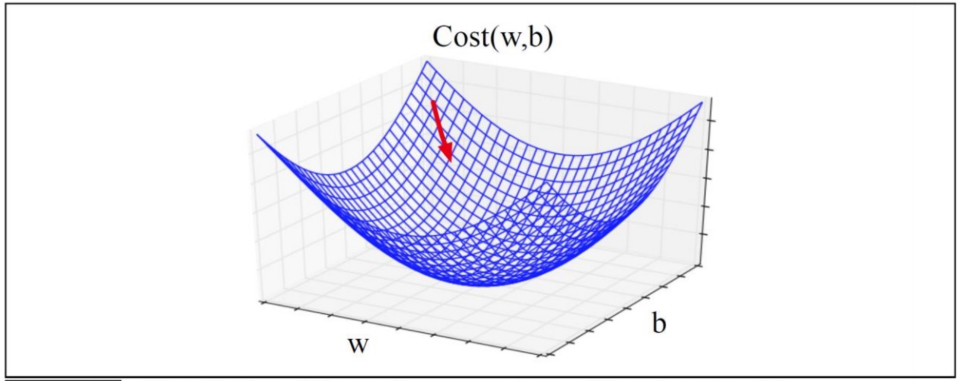


Figure 5.4 Visualization of the gradient vector in two dimensions *w* and *b*.

The gradient for logistic regression

$$L_{CE}(w,b) = -[y \log \sigma(w \cdot x + b) + (1-y)\log(1-\sigma(w \cdot x + b))]$$

$$\frac{\partial L_{CE}(w,b)}{\partial w_j} = [\sigma(w \cdot x + b) - y]x_j$$

Difference between the model prediction and the correct answer y

Feature value for dimension j

Note: the detailed derivation is available in the reading (SLP3 Chapter 5, section 5.8)

Logistic Regression and Naive Bayes

Generative and Discriminative Classifiers

- Naïve 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?

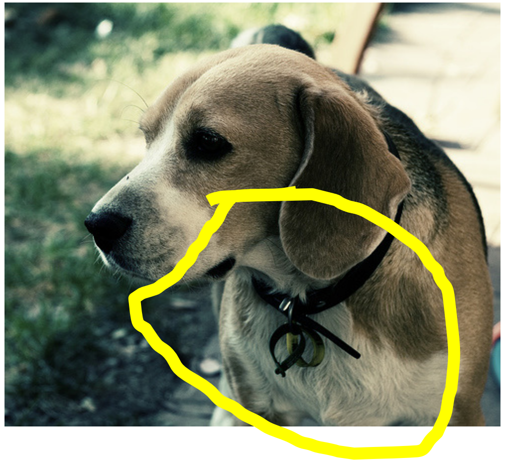
Also build a model for dog images

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

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

Bayes vs regression

- Naive Bayes assumes conditional independence. Regression does not.
- When there are many correlated features, logistic regression will assign a more accurate probability than naive Bayes.
- Naive Bayes works well on small datasets or short documents.
- Naive Bayes is easy to implement and fast to train.

Multinomial Logistic Regression

Multinomial Logistic Regression

Often we need more than 2 classes

- Positive/negative/neutral
- Parts of speech (noun, verb, adjective, adverb, preposition, etc.)
- Classify emergency SMSs into different actionable classes

If >2 classes we use **multinomial logistic regression**

- "logistic regression" will just mean binary (2 output classes)

  = Softmax regression

  = Maximum entropy modeling

  = Maxent

  = Multinomial logit

Multinomial Logistic Regression

The probability of everything must still sum to 1

```
P(positive|doc) + P(negative|doc) + P(neutral|doc) = 1
```

Need a generalization of the sigmoid called the **softmax**

- Takes a vector *z* = [*z*1, *z*2, ..., *zk*] of *k* arbitrary values
- Outputs a probability distribution
  - each value in the range [0,1]
  - all the values summing to 1

The softmax function

Turns a vector $z = [z_1, z_2, \ldots, z_k]$ of k arbitrary values into probabilities

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad 1 \leq i \leq k$$

The denominator $\sum_{i=1}^{k} e^{z_i}$ is used to normalize all the values into probabilities.

$$\text{softmax}(z) = \left[\frac{e^{z_1}}{\sum_{i=1}^{k} e^{z_i}}, \frac{e^{z_2}}{\sum_{i=1}^{k} e^{z_i}}, \ldots, \frac{e^{z_k}}{\sum_{i=1}^{k} e^{z_i}} \right]$$

The softmax classifier

- Turns a vector $z = [z_1, z_2, \ldots, z_k]$ of k arbitrary values into probabilities

$$\text{softmax}(z) = \left[\frac{e^{z_1}}{\sum_{i=1}^{k} e^{z_i}}, \frac{e^{z_2}}{\sum_{i=1}^{k} e^{z_i}}, \ldots, \frac{e^{z_k}}{\sum_{i=1}^{k} e^{z_i}} \right]$$

$$z = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]$$

$$[0.055, 0.090, 0.0067, 0.10, 0.74, 0.010]$$

Softmax in multinomial logistic regression

$$p(y=c|x) = \frac{e^{w_c \cdot x + b_c}}{\sum_{j=1}^{k} e^{w_j \cdot x + b_j}}$$

Features in (binary) logistic regression

$$f_1 = \begin{cases} 1 & \text{if "!" in doc} \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = 2.5$$

Features in softmax regression distinct weights for each value!

Var	Definition	Wt
$f_1(0,x)$	$\begin{cases} 1 & \text{if ``!''} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	-4.5
$f_1(+,x)$	$\begin{cases} 1 & \text{if ``!''} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	2.6
$f_1(-,x)$	$\begin{cases} 1 & \text{if ``!''} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1.3

Binary versus multinomial logistic regression

Multinomial is obviously applicable to many more classification tasks

- Softmax is important for neural net classifiers

Binary logistic regression is simpler to model

Summary: Logistic Regression

Classification in (binary) logistic regression: summary

- Given:
  - a set of classes: (+ sentiment,- sentiment)
  - a vector **x** of features `[x1, x2, …, xn]`
    - x1= count("awesome")
    - x2 = log(number of words in review)
  - A vector **w** of weights `[w1, w2, …, wn]`
    - w_i for each feature f_i

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

Components of a probabilistic (supervised) machine learning classifier

- A **corpus** of M observation input/output pairs, $(x^{(i)},y^{(i)})$
- For each input observation $x^{(i)}$
  - a vector of **features** $[x_1, x_2, ..., x_n]$
- A **classification function** computing $\hat{y}$, via $p(y|x)$
  - *sigmoid*
  - *softmax*
- For learning
  - A **loss function** (cross-entropy loss)
  - An **optimization algorithm** (stochastic gradient descent)