18-661 Introduction to Machine Learning

Logistic Regression

Spring 2025

ECE – Carnegie Mellon University

Announcements

- Recitation on Friday will cover Naïve Bayes and Logistic Regression. You can also ask questions about mini-exam 1.
- Mini-exam 1 will be held in the first half of class on **Monday, February 10**.
  - The exam will cover all topics through the January 29 lecture (linear algebra, MLE/MAP, linear and ridge regression, overfitting and the bias-variance tradeoff).
  - Expect a mix of true/false, multiple choice, and descriptive questions.
  - Make sure you attend class in person that day. No makeups barring extraordinary special circumstances.
- Homework 2 is due on Friday, February 21. After this lecture, you should be able to attempt most of the problems (except those on SVMs).

Outline

1. Review of Naïve Bayes

2. Logistic Regression Model

3. Loss Function and Parameter Estimation

4. Non-linear Decision Boundaries

5. Evaluating Classification Methods

Review of Naïve Bayes

Naïve Bayes Model for Identifying Spam

- **Class label**: binary
  - $y = \{$ spam, ham $\}$
- **Features**: word counts in the document (bag-of-words)
  - $\mathbf{x} = \{($‘free’, 100$)$, $($‘lottery’, 5$)$, $($‘money’, 10$)\}$
  - Each pair is in the format of $(\text{word}_i, \#\text{word}_i)$, namely, a unique word in the dictionary, and the number of times it shows up
- **Model**

$$p(\mathbf{x}|spam) = p(\text{‘}free'|spam)^{100} p(\text{‘}lottery'|spam)^{5} p(\text{‘}money'|spam)^{10} \cdots$$

  - Choose the “most likely” option: $p(\mathbf{x}|spam)p(spam)$ vs. $p(\mathbf{x}|ham)p(ham)$

These conditional probabilities are the parameters we need to estimate

Why Is This Naïve?

- Strong assumption of conditional independence:

$$p(\text{word}_i, \text{word}_j|y) = p(\text{word}_i|y)p(\text{word}_j|y)$$

- Previous example:

$$p(\mathbf{x}|spam) = p(`free'|spam)^{100}p(`lottery'|spam)^{5}p(`money'|spam)^{10}\cdots$$

- Independence across different words as well as multiple occurrences of the same word
- This assumption makes estimation much easier (as we'll see)

Bayes Classification Rule

MAP rule: For any document $\mathbf{x}$, we want to compare

$$p(\text{spam}|\mathbf{x}) \text{ versus } p(\text{ham}|\mathbf{x})$$

Recall that by Bayes rule we have:

$$p(\text{spam}|\mathbf{x}) = \frac{p(\mathbf{x}|\text{spam})p(\text{spam})}{p(\mathbf{x})}$$

$$p(\text{ham}|\mathbf{x}) = \frac{p(\mathbf{x}|\text{ham})p(\text{ham})}{p(\mathbf{x})}$$

Denominators are same, and easier to compute logarithms, so instead we compare:

$$\log[p(\mathbf{x}|\text{spam})p(\text{spam})] \quad \text{versus} \quad \log[p(\mathbf{x}|\text{ham})p(\text{ham})]$$

Classifier in Linear Form

$$\begin{aligned}\log[p(\mathbf{x}|\text{spam})p(\text{spam})] &= \log\left[\prod_i p(\text{word}_i|\text{spam})^{\#\text{word}_i}p(\text{spam})\right]\\ &= \sum_i(\#\text{word}_i)\log p(\text{word}_i|\text{spam}) + \log p(\text{spam})\end{aligned}$$

Similarly, we have

$$\log[p(\mathbf{x}|\text{ham})p(\text{ham})] = \sum_i(\#\text{word}_i)\log p(\text{word}_i|\text{ham}) + \log p(\text{ham})$$

We're back to the idea of comparing weighted sums of word occurrences!

$\log p(\mathit{spam})$ *and* $\log p(\mathit{ham})$ *are called "priors" (in our initial example we did not include them but they are important!)*

Estimating the Conditional and Prior Probabilities

- Collect a lot of ham and spam emails as training examples
- Estimate the "prior"

$$p(\text{ham}) = \frac{\#\text{of ham emails}}{\#\text{of emails}}, \quad p(\text{spam}) = \frac{\#\text{of spam emails}}{\#\text{of emails}}$$

- Estimate the weights, e.g., $p(\text{funny\_word}|\text{ham})$

$$p(\text{funny\_word}|\text{ham}) = \frac{\#\text{of funny\_word in ham emails}}{\#\text{of words in ham emails}}$$

$$p(\text{funny\_word}|\text{spam}) = \frac{\#\text{of funny\_word in spam emails}}{\#\text{of words in spam emails}}$$

- Use Laplacian smoothing to avoid these probabilities being 0 for any word

Missing Features: Some Words Never Occur in Ham Emails

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	0	2	3	Ham
Email 4	1	0	3	2	Ham

Find ML estimates of parameters π_c and θ_{ck}

In this training phase, we can just use all available values and ignore missing values

$$
\begin{aligned}
\theta_{spam,free} &= Pr(free|spam) = 9/18 & \theta_{ham,free} &= Pr(free|ham) = 3/13 \\
\theta_{spam,bank} &= Pr(bank|spam) = 5/18 & \theta_{ham,bank} &= Pr(bank|ham) = 0/13 \\
\theta_{spam,meet} &= Pr(meet|spam) = 2/18 & \theta_{ham,meet} &= Pr(meet|ham) = 5/13 \\
\theta_{spam,time} &= Pr(time|spam) = 2/18 & \theta_{ham,time} &= Pr(time|ham) = 5/13 \\
\pi_{spam} &= Pr(spam) = 2/4 & \pi_{ham} &= Pr(ham) = 2/4
\end{aligned}
$$

Dealing with Missing Features: Solution 2

Use Laplacian smoothing: Pretend you've seen each word 1 extra time for each class (spam and ham). This is called a 'pseudo-count'.

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	0	2	3	Ham
Email 4	1	0	3	2	Ham

$$\begin{aligned}
\theta_{ham,free} &= Pr(free|ham) = (3+1)/(13+4) \\
\theta_{ham,bank} &= Pr(bank|ham) = (0+1)/(13+4) \\
\theta_{ham,meet} &= Pr(meet|ham) = (5+1)/(13+4) \\
\theta_{ham,time} &= Pr(time|ham) = (5+1)/(13+4) \\
\pi_{ham} &= Pr(ham) = 2/4
\end{aligned}$$

Dealing with Missing Features: Solution 2

More generally, pretend you've seen each word $\alpha \geq 1$ extra times.

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	0	2	3	Ham
Email 4	1	0	3	2	Ham

$$p(\text{funny\_word}|\text{spam}) = \frac{\#\text{of funny\_word in spam emails} + \alpha}{\#\text{of words in spam emails} + \alpha \times \#\text{of unique words}}$$

Laplacian Smoothing: History and Effect on ML Estimate

History: What is the prob. that the sun will rise tomorrow?

- Given a large sample of days with the rising sun, we still can not be completely sure that the sun will still rise tomorrow

$$\Pr(\text{sun rising tomorrow}|\text{it rose } t \text{ times}) = \frac{t+1}{t+2}$$

Effect on the ML estimate

- Laplace smoothing biases the ML estimate
- Equivalent to performing MAP estimation with a Dirichlet (multi-variate Beta) prior
- As training data size grows, the effect of Laplacian smoothing disappears

Classification Rule

Given an unlabeled point $\mathbf{x} = \{x_k, k = 1, 2, \cdots, \mathrm{K}\}$, how to label it?

$$\begin{aligned} y^* &= \arg\max_{c \in [\mathrm{C}]} P(y = c|\mathbf{x}) \\ &= \arg\max_{c \in [\mathrm{C}]} P(y = c)P(\mathbf{x}|y = c) \\ &= \arg\max_c [\log \pi_c + \sum_k x_k \log \theta_{ck}] \end{aligned}$$

Choose class c that maximizes the log-likelihood of an observed email

Outline

1. Review of Naïve Bayes
2. Logistic Regression Model
3. Loss Function and Parameter Estimation
4. Non-linear Decision Boundaries
5. Evaluating Classification Methods

Logistic Regression Model

How Does Naïve Bayes Work?

Examine the classification rule for Naïve Bayes

$$y^* = \arg\max_c \left(\log \pi_c + \sum_k x_k \log \theta_{ck} \right)$$

For binary classification, we thus determine the label based on the sign of

$$\log \pi_1 + \sum_k x_k \log \theta_{1k} - \left(\log \pi_2 + \sum_k x_k \log \theta_{2k} \right)$$

This is just a linear function of the features (word-counts) $\{x_k\}$

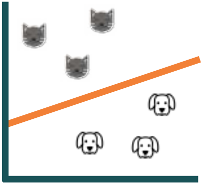
$$w_0 + \sum_k x_k w_k$$

where we "absorb" $w_0 = \log \pi_1 - \log \pi_2$ and $w_k = \log \theta_{1k} - \log \theta_{2k}$.

Intuition: Logistic Regression

Learn the equation of the decision boundary $\mathbf{w}^\top \mathbf{x} = 0$ such that

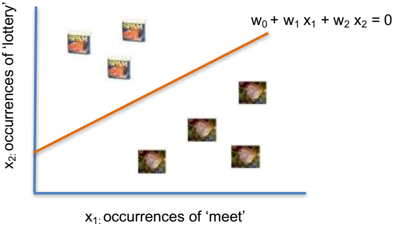
- If $\mathbf{w}^\top \mathbf{x} \geq 0$ declare $y = 1$ (cat)
- If $\mathbf{w}^\top \mathbf{x} < 0$ declare $y = 0$ (dog)



$y = 0$ for dog, $y = 1$ for cat

Back to Spam vs. Ham Classification...

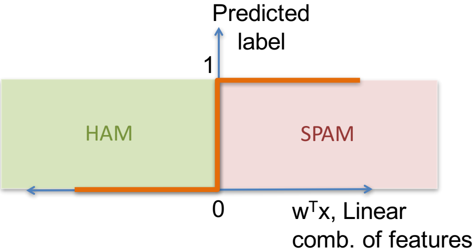
- $x_1 = \#$ of times 'meet' appears in an email
- $x_2 = \#$ of times 'lottery' appears in an email
- Define feature vector $\mathbf{x} = [1, x_1, x_2]$
- Learn the decision boundary $w_0 + w_1 x_1 + w_2 x_2 = 0$ such that
  - If $\mathbf{w}^\top \mathbf{x} \geq 0$ declare $y = 1$ (spam)
  - If $\mathbf{w}^\top \mathbf{x} < 0$ declare $y = 0$ (ham)


$w_0 + w_1 x_1 + w_2 x_2 = 0$
x_2: occurrences of 'lottery'
x_1: occurrences of 'meet'

Key Idea: If 'meet' appears few times and 'lottery' appears many times than the email is spam

Visualizing a Linear Classifier

- x_1 = # of times 'lottery' appears in an email
- x_2 = # of times 'meet' appears in an email
- Define feature vector $\mathbf{x} = [1, x_1, x_2]$
- Learn the decision boundary $w_0 + w_1 x_1 + w_2 x_2 = 0$ such that
  - If $\mathbf{w}^\top \mathbf{x} \geq 0$ declare $y = 1$ (spam)
  - If $\mathbf{w}^\top \mathbf{x} < 0$ declare $y = 0$ (ham)



$y = 1$ for spam, $y = 0$ for ham

Your Turn

Suppose you see the following email:

```
CONGRATULATIONS!! Your email address have won you the
lottery sum of US$2,500,000.00 USD to claim your prize,
contact your office agent (Athur walter) via email
claims2155@yahoo.com.hk or call +44 704 575 1113
```

Keywords are [lottery, prize, office, email]
The given weight vector is $\mathbf{w} = [0.3, 0.3, -0.1, -0.04]^\top$

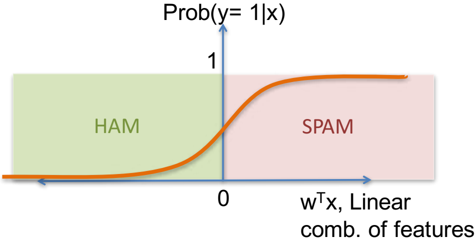
Will we predict that the email is spam or ham?

$$\mathbf{x} = [1, 1, 1, 2]^\top$$
$$\mathbf{w}^\top \mathbf{x} = 0.3 * 1 + 0.3 * 1 - 0.1 * 1 - 0.04 * 2 = 0.42 > 0$$

so we predict spam!

Intuition: Logistic Regression

- Suppose we want to output the probability of an email being spam/ham instead of just 0 or 1
- This gives information about the confidence in the decision
- Use a function $\sigma(\mathbf{w}^\top \mathbf{x})$ that maps $\mathbf{w}^\top \mathbf{x}$ to a value between 0 and 1



Probability that predicted label is 1 (spam)

Key Problem: Finding optimal weights $\mathbf{w}$ that accurately predict this probability for a new email

Formal Setup: Binary Logistic Classification

- Input/features: $\boldsymbol{x} = [1, x_1, x_2, \dots x_D] \in \mathbb{R}^{D+1}$
- Output: $y \in \{0, 1\}$
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \dots, N\}$
- Model:

$$p(y = 1 | \boldsymbol{x}; \boldsymbol{w}) = \sigma[g(\boldsymbol{x})]$$

where

$$g(\boldsymbol{x}) = w_0 + \sum_d w_d x_d = \boldsymbol{w}^\top \boldsymbol{x}$$

and $\sigma[\cdot]$ stands for the sigmoid function

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

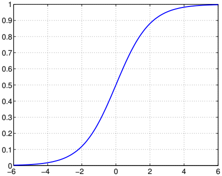
Why the Sigmoid Function?

What does it look like?

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

where

$$a = \boldsymbol{w}^\top \boldsymbol{x}$$

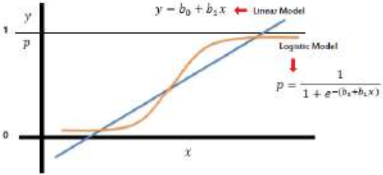


Sigmoid properties

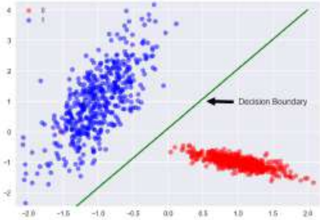
- Bounded between 0 and 1 ← thus, interpretable as probability
- Monotonically increasing ← thus, usable to derive classification rules
  - $\sigma(a) \geq 0.5$, positive (classify as '1')
  - $\sigma(a) < 0.5$, negative (classify as '0')
- Nice computational properties ← as we will see soon

Comparison to Linear Regression

Sigmoid function returns values in [0,1]



Decision boundary is linear: linear classifier



Loss Function and Parameter Estimation

How Do We Optimize the Weight Vector w?

Learn from experience

- get a lot of spams
- get a lot of hams

But what to optimize?

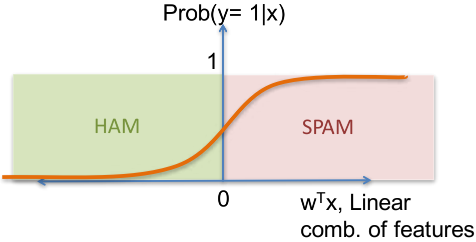
Likelihood Function

Probability of a single training sample $(\boldsymbol{x}_n, y_n)$...

$$p(y_n|\boldsymbol{x}_n; \boldsymbol{w}) = \begin{cases} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) & \text{if} \quad y_n = 1 \\ 1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) & \text{otherwise} \end{cases}$$

Simplify, using the fact that y_n is either 1 or 0

$$p(y_n|\boldsymbol{x}_n; \boldsymbol{w}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)^{y_n}[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]^{1-y_n}$$



Probability that predicted label is 1 (spam)

Log Likelihood or Cross Entropy Error

Log-likelihood of the whole training data $\mathcal{D}$

$$P(\mathcal{D}) = \prod_{n=1}^{N} p(y_n | \boldsymbol{x}_n; \boldsymbol{w}) = \prod_{n=1}^{N} \left\{ \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)^{y_n} [1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]^{1-y_n} \right\}$$

$$\log P(\mathcal{D}) = \sum_n \{ y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)] \}$$

It is convenient to work with its negation, which is called the cross-entropy error function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{ y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)] \}$$

Cross-Entropy as a Loss Function

We aim to build a function $h(\boldsymbol{x})$ to predict the true value y associated with $\boldsymbol{x}$. If we make a mistake, we incur a loss

$$\ell(h(\boldsymbol{x}), y)$$

Cross-entropy is also a sum over all data samples:

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

where $h(\boldsymbol{x}) = \sigma\left(\boldsymbol{w}^T \boldsymbol{x}\right)$.

What is the loss function?

$$\ell(h(\boldsymbol{x}_n), y) = -\{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

How to Find the Optimal Parameters for Logistic Regression?

We will minimize the cross-entropy function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_{n}\{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

However, this function is complex and we cannot find the simple solution as we did in Naïve Bayes. So we need to use numerical methods.

- Numerical methods are messier, in contrast to cleaner closed-form solutions.
- In practice, we often have to tune a few optimization parameters — patience is necessary.
- A popular method: gradient descent and its variants.

Gradient Descent Update for Logistic Regression

Finding the gradient of $\mathcal{E}(\boldsymbol{w})$ looks very hard, but it turns out to be simple and intuitive.

Let's start with the derivative of the sigmoid function $\sigma(a)$:

$$\begin{aligned}
\frac{d}{d\,a}\sigma(a) &= \frac{d}{d\,a}\left(1+e^{-a}\right)^{-1} \\
&= \frac{-1}{(1+e^{-a})^2}\frac{d}{d\,a}(1+e^{-a}) \\
&= \frac{e^{-a}}{(1+e^{-a})^2} \\
&= \frac{1}{1+e^{-a}}\frac{e^{-a}}{1+e^{-a}} \\
&= \frac{1}{1+e^{-a}}\frac{1+e^{-a}-1}{1+e^{-a}} \\
&= \sigma(a)[1-\sigma(a)]
\end{aligned}$$

Gradients of the Cross-entropy Function

Cross-entropy Error Function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

$$\frac{d}{d\,a}\sigma(a) = \sigma(a)[1 - \sigma(a)]$$

Computing the gradient

$$\begin{aligned}
\frac{\partial \mathcal{E}(\boldsymbol{w})}{\partial \boldsymbol{w}} &= -\sum_n \Big\{ y_n \frac{\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]}{\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)} \boldsymbol{x}_n \\
&\qquad - (1 - y_n) \frac{\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]}{1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)} \boldsymbol{x}_n \Big\} \\
&= -\sum_n \{ y_n [1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)] \boldsymbol{x}_n - (1 - y_n) \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) \boldsymbol{x}_n \} \\
&= \sum_n \underbrace{\{ \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) - y_n \}}_{\text{Error of the } n\text{th training sample.}} \boldsymbol{x}_n
\end{aligned}$$

Numerical Optimization

Gradient descent for logistic regression

- Choose a proper step size $\eta > 0$
- Iteratively update the parameters following the negative gradient to minimize the error function

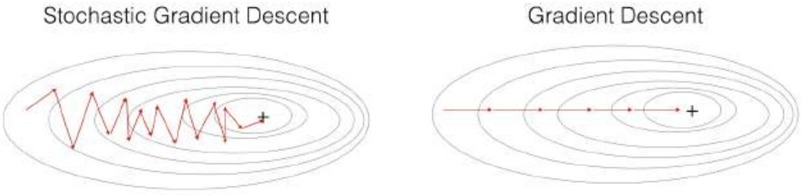
$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \sum_{n} \left\{ \sigma(\boldsymbol{w}^{(t)\top} \boldsymbol{x}_n) - y_n \right\} \boldsymbol{x}_n$$

Stochastic gradient descent for logistic regression

- Choose a proper step size $\eta > 0$
- Draw a sample *n* uniformly at random
- Iteratively update the parameters following the negative gradient to minimize the error function

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \left\{ \sigma(\boldsymbol{w}^{(t)\top} \boldsymbol{x}_n) - y_n \right\} \boldsymbol{x}_n$$

SGD versus Batch GD



Stochastic Gradient Descent

Gradient Descent

- SGD reduces per-iteration complexity since it considers fewer samples.
- But it is noisier and can take longer to converge.

Example: Spam Classification

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	1	2	3	Ham
Email 4	1	2	3	2	Ham

Perform gradient descent to learn weights **w**

- Feature vector for email 1: $\mathbf{x}_1 = \begin{bmatrix} 1 & 5 & 3 & 1 & 1 \end{bmatrix}^\top$
- Let $\boldsymbol{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \end{bmatrix}^\top$, the matrix of all feature vectors.
- Initial weights $\mathbf{w} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}^\top$
- Prediction $\begin{bmatrix} \sigma(\mathbf{w}^\top \mathbf{x}_1) & \sigma(\mathbf{w}^\top \mathbf{x}_2) & \sigma(\mathbf{w}^\top \mathbf{x}_3) & \sigma(\mathbf{w}^\top \mathbf{x}_4) \end{bmatrix}^\top$

$$= \begin{bmatrix} 0.996 & 0.989 & 0.989 & 0.989 \end{bmatrix}^\top$$

from computing $\mathbf{Xw}$ and then applying $\sigma(\cdot)$ entry-wise.

Example: Spam Classification, Batch Gradient Descent

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	1	2	3	Ham
Email 4	1	2	3	2	Ham

Perform gradient descent to learn weights **w**

- Prediction $\sigma(\boldsymbol{X}\mathbf{w}) = \begin{bmatrix} 0.996 & 0.989 & 0.989 & 0.989 \end{bmatrix}^\top$
- Difference from labels $\boldsymbol{y} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}^\top$ is

$$\sigma(\boldsymbol{X}\mathbf{w}) - \boldsymbol{y} = \begin{bmatrix} -0.004 & -0.011 & 0.989 & 0.989 \end{bmatrix}^\top$$

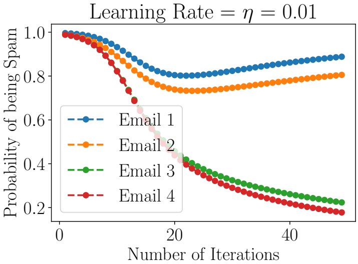
- Gradient of the first email,

$$\boldsymbol{g}_1 = (\sigma(\mathbf{w}^\top \mathbf{x}_1) - y_1)\mathbf{x}_1 = -0.004 \begin{bmatrix} 1 & 5 & 3 & 1 & 1 \end{bmatrix}^\top$$

- $\mathbf{w} \leftarrow \mathbf{w} - \underbrace{0.01}_{\text{learning rate}} \sum_n \boldsymbol{g}_n = \mathbf{w} - \eta \boldsymbol{X}^T(\sigma(\boldsymbol{X}\mathbf{w}) - \boldsymbol{y})$

Example: Spam Classification, Batch Gradient Descent

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	1	2	3	Ham
Email 4	1	2	3	2	Ham



Predictions for Emails 3 and 4 are initially close to 1 (spam), but they converge towards the correct value 0 (ham)

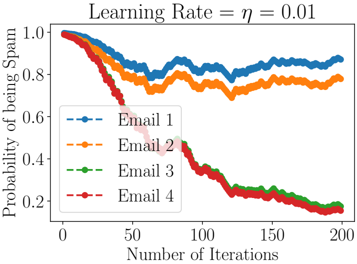
Example: Spam Classification, Stochastic Gradient Descent

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	1	2	3	Ham
Email 4	1	2	3	2	Ham

- Prediction $\sigma(\mathbf{w}^\top \mathbf{x}_r) = 0.996$ for a randomly chosen email r
- Difference from label $y = 1$ is -0.004
- Gradient is $\boldsymbol{g}_r = (\sigma(\mathbf{w}^\top \mathbf{x}_n) - y)\mathbf{x}_r = -0.004\mathbf{x}_r$
- $\mathbf{w} \leftarrow \mathbf{w} - 0.01\boldsymbol{g}_r$

Example: Spam Classification, Stochastic Gradient Descent

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	1	2	3	Ham
Email 4	1	2	3	2	Ham



Predictions for Emails 3 and 4 are initially close to 1 (spam), but they converge towards the correct value 0 (ham)

Example: Spam Classification, Test Phase

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	1	2	3	Ham
Email 4	1	2	3	2	Ham

- Final $\mathbf{w} = [0.187, 0.482, 0.179, -0.512, -0.524]^\top$ after 50 batch gradient descent iterations.
- Given a new email with feature vector $\mathbf{x} = [1, 1, 3, 4, 2]$, the probability of the email being spam is estimated as $\sigma(\mathbf{w}^\top \mathbf{x}) = \sigma(-1.889) = 0.13$.
- Since this is less than 0.5 we predict ham.

Contrast Naïve Bayes and Logistic Regression

Both classification models are linear functions of features

Joint vs. Conditional distribution
Naive Bayes models the joint distribution: $P(X, Y) = P(Y)P(X|Y)$

Logistic regression models the conditional distribution: $P(Y|X)$

Correlated vs. Independent features
Naive Bayes assumes independence of features and multiple occurences

Logistic Regression implicitly captures correlations when training weights

Generative vs. Discriminative
NB is a generative model, LR is a discriminative model

Generative Model vs. Discriminative Model

$\{x : P(Y = 1|X = x) = P(Y = 0|X = x)\}$ is called the decision boundary of our data.

Generative classifiers
Model the class-conditional densities $P(Y|X = x)$ explicitly:

$$P(Y = 1|X = x) = \frac{P(X = x|Y = 1)P(Y = 1)}{P(X = x|Y = 1)P(Y = 1) + P(X = x|Y = 0)P(Y = 0)}$$

This means we need to separately estimate both $P(X|Y)$ and $P(Y)$.

Discriminative classifier
Directly model the decision boundary and avoid estimating the conditional probabilities.

Logistic Regression vs. Linear Regression

	Logistic regression	**Linear regression**
Training data	$(\boldsymbol{x}_n, y_n), y_n \in \{0,1\}$	$(\boldsymbol{x}_n, y_n), y_n \in \mathbb{R}$
Loss function	cross-entropy	RSS
Interpretation of $y_n \| \boldsymbol{x}_n, \boldsymbol{w}$	$\sim \mathrm{Ber}(\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))$	$\sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2)$
Gradient per sample	$\left(\sigma(\boldsymbol{x}_n^\top \boldsymbol{w}) - y_n\right) \boldsymbol{x}_n$	$\left(\boldsymbol{x}_n^\top \boldsymbol{w} - y_n\right) \boldsymbol{x}_n$

Cross-entropy loss function (logistic regression):

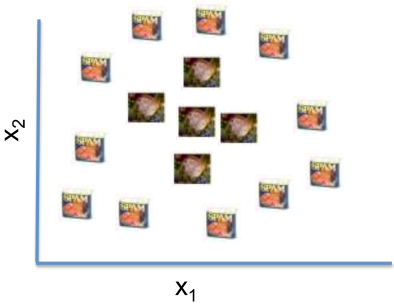
$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

RSS loss function (linear regression):

$$RSS(\boldsymbol{w}) = \frac{1}{2} \sum_n (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2$$

Non-linear Decision Boundaries

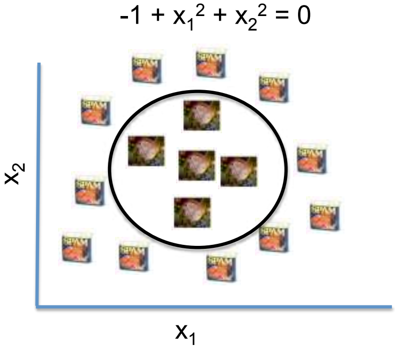
How to Handle More Complex Decision Boundaries?



- This data is not linearly separable...
- Use non-linear basis functions to add more features.

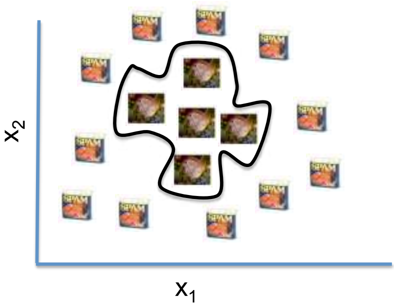
Adding Polynomial Features

- New feature vector is $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2]$
- $\Pr(y = 1|\mathbf{x}) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$
- If $\mathbf{w} = [-1, 0, 0, 1, 1]$, the boundary is $-1 + x_1^2 + x_2^2 = 0$
    - If $-1 + x_1^2 + x_2^2 \geq 0$ declare spam
    - If $-1 + x_1^2 + x_2^2 < 0$ declare ham



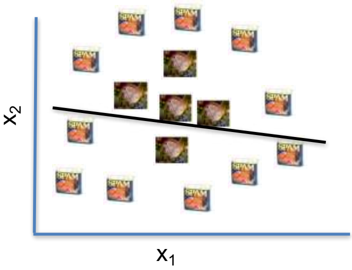
Adding Polynomial Features

- What if we add many more features and define $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3, \ldots]$?
- We get a complex decision boundary

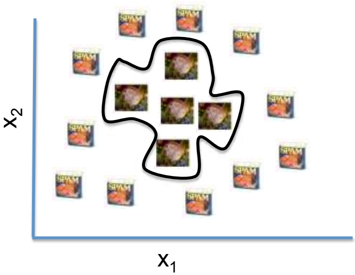


Can result in overfitting and bad generalization to new data points.

Concept-check: Bias-Variance Trade-off



high bias



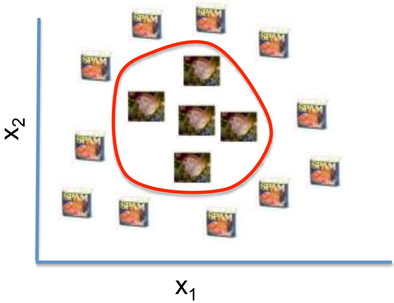
high variance

Solution to Overfitting: Regularization

- Add regularization term to be cross entropy loss function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_{n} \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1-y_n) \log[1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\} + \underbrace{\frac{1}{2}\lambda \|w\|_2^2}_{\text{regularization}}$$

- Perform gradient descent on this regularized function
- Often, we do NOT regularize the bias term w_0



Evaluating Classification Methods

Loss Functions

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

- Easy to optimize!
- Average loss over the (training, validation, test) dataset
- ...but what does it mean?

Interpretable Classification Metrics

True positive	False positive
False negative	True negative

- Measure the accuracy within each class
- Accounts for imbalance between classes

- Sensitivity: true positive rate

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

- Specificity: true negative rate

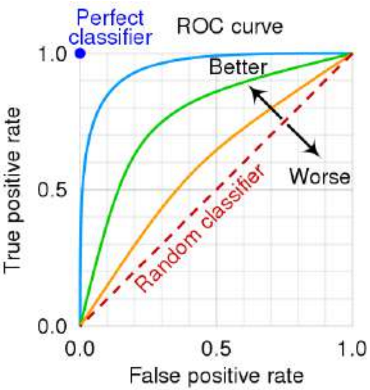
$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

- Precision: positive predictive value

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

These metrics are difficult to optimize directly, but they have the advantage of being easily interpretable.

Combining These Metrics: the ROC Curve



Receiver Operating Characteristic (ROC)

- Define a "threshold" for the positive/negative split
- Increasing the threshold: more samples are predicted to be positive
- Area Under the ROC Curve: want this as large as possible

You Should Know

- What is logistic regression and solving for **w** using gradient descent on the cross entropy loss function
- Differences between Naïve Bayes and Logistic Regression.
- How to generalize logistic regression to handle nonlinear decision boundaries.
- How to measure classification accuracy