18-661 Introduction to Machine Learning

Logistic Regression

Spring 2020

ECE – Carnegie Mellon University

Announcements

- Python tutorial will be held tomorrow (Thursday, 2/6) at 1:30pm ET in WEH 5312. Zoom link will be provided if you cannot attend in person, and we will post the materials on Piazza.
- Recitation on Friday will cover practical considerations for implementing logistic regression, using a digit recognition dataset. Please download the associated Jupyter notebook (to be posted later today) so you can follow along.
- The midterm exam (20% of your grade) will be an in-class exam on 2/26. It will be closed-book and paper-based; more details to come.

Outline

1. Review of Naive Bayes
2. Logistic Regression Model
3. Loss Function and Parameter Estimation
4. Gradient Descent

Review of Naive Bayes

How to identify spam emails?

FROM THE DESK OF MR. AMINU SALEH
DIRECTOR, FOREIGN OPERATIONS DEPARTMENT
AFRI BANK PLC
Afribank Plaza,
14th Floormoney344.jpg
51/55 Broad Street,
P.M.B 12021 Lagos-Nigeria

Attention: Honorable Beneficiary,

IMMEDIATE PAYMENT NOTIFICATION VALUED AT **US$10 MILLION**

Hi Virginia,

Can we meet today at 2pm?

thanks,

Carlee

Bag of words model

Bag-of-word representation of documents (and textual data)

$$\begin{pmatrix} \text{free} & 100 \\ \text{money} & 2 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$

Just wanted to send a quick reminder about the guest lect
noon. We meet in RTH 105. It has a PC and LCD projector
connection for your laptop if you desire. Maybe we can m
to setup the A/V stuff.

Again, if you would be able to make it around 30 minutes
great.

Thanks so much for your willingness to do this,
Mark

$$\begin{pmatrix} \text{free} & 1 \\ \text{money} & 1 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$

Weighted sum of those telltale words

$$\begin{pmatrix} \text{free} & 100 \\ \text{money} & 2 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$

$$\begin{pmatrix} 100 \times 0.2 \\ 2 \times 0.3 \\ \vdots \\ 2 \times 0.3 \\ \vdots \end{pmatrix}$$

$$\begin{pmatrix} 100 \times 0.01 \\ 2 \times 0.02 \\ \vdots \\ 2 \times 0.01 \\ \vdots \end{pmatrix}$$

different weights for spam and ham: representing how compatible the word pattern is to each category

= 3.2

= 1.03

Intuitive approach

- **Class label**: binary
  - $y = \{$ spam, ham $\}$
- **Features**: word counts in the document (bag-of-words)
  - $\mathbf{x} = \{(\text{'free'}, 100), (\text{'lottery'}, 5), (\text{'money'}, 10)\}$
  - Each pair is in the format of $(w_i, \#w_i)$, namely, a unique word in the dictionary, and the number of times it shows up
- **Assign weight to each word**
  - Let $s :=$ spam weights, $h :=$ ham weights
  - Compute compatibility score of **spam**
    - $(\#\ \text{“free”}\ \times s_{\text{free}}) + (\ \#\ \text{“account”}\ \times s_{\text{account}}) + (\#\ \text{“money”}\ \times s_{\text{money}})$
  - Compute compatibility score of **ham**
    - $(\#\ \text{“free”}\ \times h_{\text{free}}) + (\ \#\ \text{“account”}\ \times h_{\text{account}}) + (\#\ \text{“money”}\ \times h_{\text{money}})$
- **Make a decision**
  - if spam score $>$ ham score then spam
  - else ham

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:

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

Naive Bayes Classification Rule

The Naive Bayes assumption: **conditional independence of features**

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

The decision score becomes:

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

Similarly, we have

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

Comparing these log likelihoods. If

$$\sum_i x_i \log p(\text{word}_i|\text{spam}) + \log p(\text{spam}) > \sum_i x_i \log p(\text{word}_i|\text{ham}) + \log p(\text{ham})$$

then declare the email as 'spam'

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

Laplacian smoothing

- If a word has 0 probability within a class, naive Bayes will never put an email with that word into that class.
- Introduce pseudo-counts by pretending you saw each word α times in each class.

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

Effect of Laplacian smoothing diminishes with more training data.

Outline

1. Review of Naive Bayes
2. Logistic Regression Model
3. Loss Function and Parameter Estimation
4. Gradient Descent

Logistic Regression Model

How does naive Bayes work?

Examine the classification rule for naive 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) $\{x_k\}$

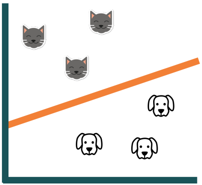
$$w_0 + \sum_k x_k w_k$$

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

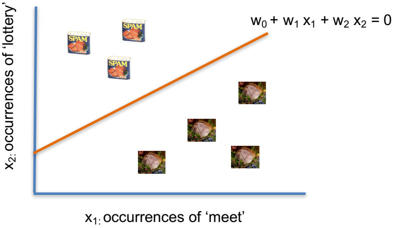
- If $\mathbf{w}^\top \mathbf{x} \geq 0$ declare $y = 1$ (cat)
- If $\mathbf{w}^\top \mathbf{x} < 0$ declare $y = 0$ (dog)



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

Back to spam vs. ham classification...

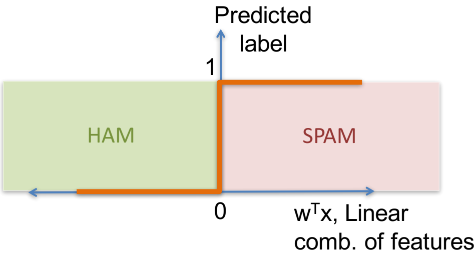
- $x_1 = \#$ of times 'meet' appears in an email
- $x_2 = \#$ of times 'lottery' appears in an email
- Define feature vector $\mathbf{x} = [1, x_1, x_2]$
- Learn the decision boundary $w_0 + w_1 x_1 + w_2 x_2 = 0$ such that
  - If $\mathbf{w}^\top \mathbf{x} \geq 0$ declare $y = 1$ (spam)
  - If $\mathbf{w}^\top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



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?

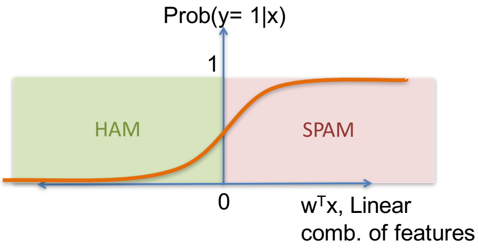
$$\mathbf{x} = [1, 1, 1, 2]^\top$$

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

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

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

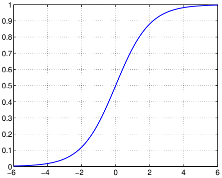
Why the sigmoid function?

What does it look like?

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

where

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

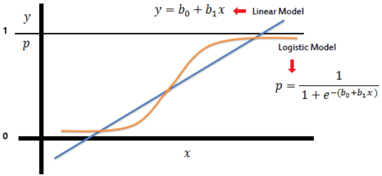


Sigmoid properties

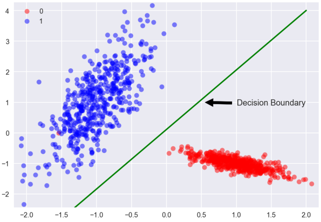
- Bounded between 0 and 1 $\leftarrow$ thus, interpretable as probability
- Monotonically increasing $\leftarrow$ thus, usable to derive classification rules
  - $\sigma(a) \geq 0.5$, positive (classify as '1')
  - $\sigma(a) < 0.5$, negative (classify as '0')
- Nice computational properties $\leftarrow$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

Comparison to Linear Regression

Sigmoid function returns values in [0,1]



Decision boundary is linear



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 $\mathbf{w} = [0.3, 0.3, -0.1, -0.04]^\top$

What is the probability that the email is spam?

$$\mathbf{x} = [1, 1, 1, 2]^\top$$

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

$$\Pr(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-0.42}} = 0.603$$

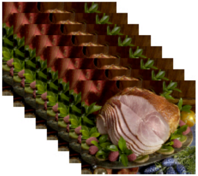
Loss Function and Parameter Estimation

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?



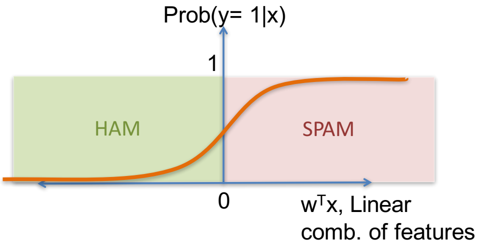
Likelihood function

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

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

How to find the optimal parameters for logistic regression?

We will minimize the error function

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

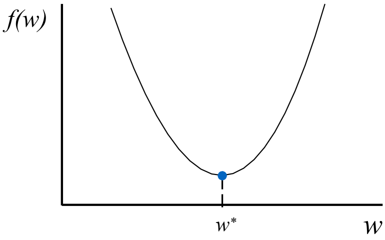
However, this function is complex and we cannot find the simple solution as we did in Naive Bayes. So we need to use numerical methods.

- 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

Gradient descent algorithm

Start at a random point



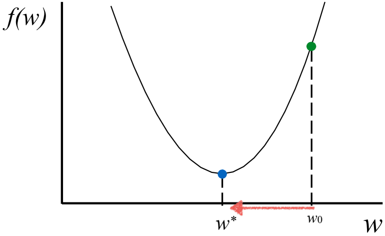
f(w)

Gradient descent algorithm

Start at a random point.

Repeat:

- Determine a descent direction.

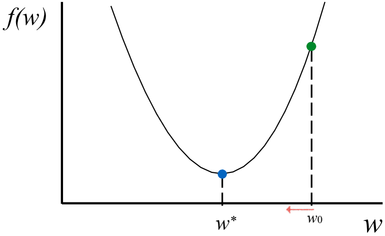


Gradient descent algorithm

Start at a random point.

Repeat:

- Determine a descent direction.
- Choose a step size.

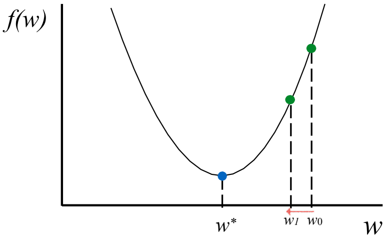


Gradient descent algorithm

Start at a random point.

Repeat:

- Determine a descent direction.
- Choose a step size.
- Update.



f(w

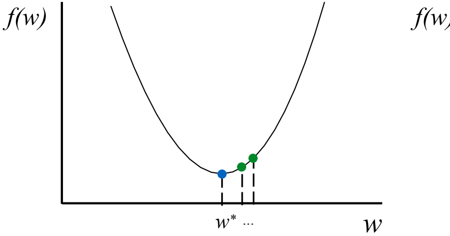
Gradient descent algorithm

Start at a random point.

Repeat:

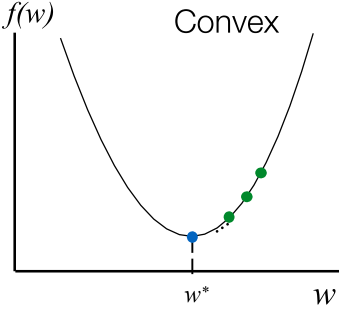
- Determine a descent direction.
- Choose a step size.
- Update.

Until stopping criterion is reached.

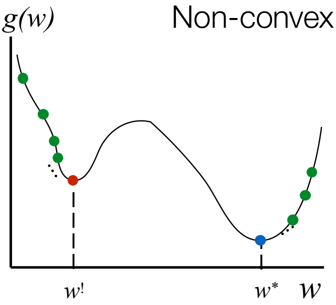


Can we converge to the optimum?

Gradient descent (with proper step size) converges to the global optimum for when minimizing a convex function.



Any local minimum is also a global minimum.



Multiple local minima may exist.

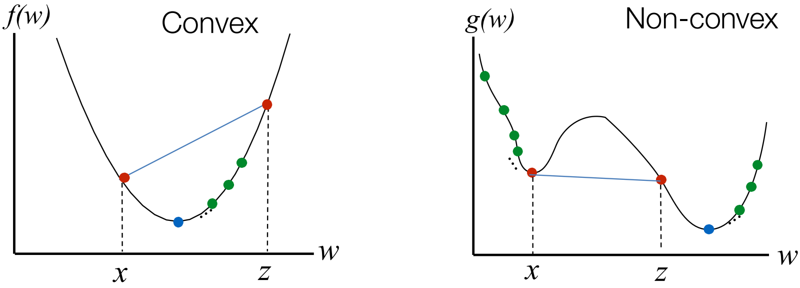
Linear regression, ridge regression, and logistic regression are all convex!

Convexity

A function $f : \mathbb{R}^k \to \mathbb{R}$ is convex if for any $x, z \in \mathbb{R}^k$ and $t \in [0, 1]$,

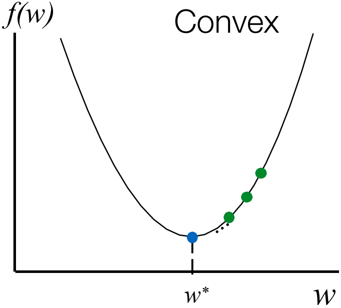
$$\underbrace{f(tx + (1-t)z)}_{\text{Function value at a point between } x \text{ and } z.} \leq \underbrace{tf(x) + (1-t)f(z)}_{\text{Line drawn between } f(x) \text{ and } f(z)} .$$

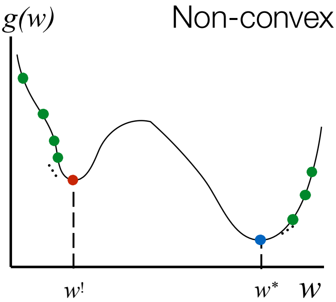
- f always lies below a line drawn between two of its points.
- If it exists, the Hessian $\frac{d^2 f}{dx^2}$ is a positive-(semi)definite matrix.



Can we converge to the optimum?

Gradient descent (with proper step size) converges to the global optimum for when minimizing a convex function.



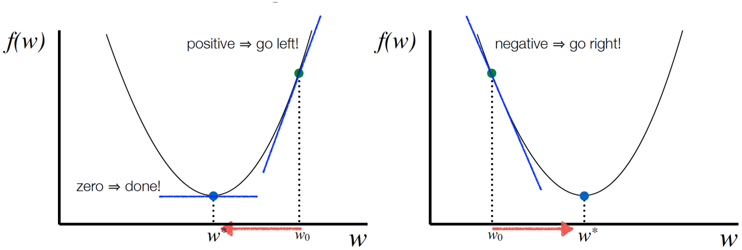


Any local minimum is also a global minimum.

Multiple local minima may exist.

Linear regression, ridge regression, and logistic regression are all convex!

Why do we move in the direction opposite the gradient?



We can only move in two directions

Negative slope is direction of descent!

Step Size

Update Rule: $w_{i+1} = w_i - \alpha_i \frac{df}{dw}(w_i)$

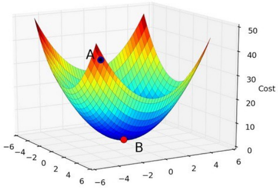
Negative Slope

We have seen this before!

(Batch) gradient descent for linear regression

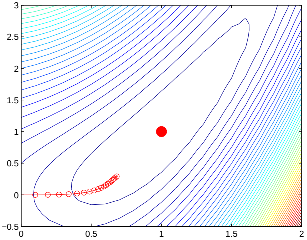
$$RSS(\mathbf{w}) = \sum_n [y_n - \mathbf{w}^\top \mathbf{x}_n]^2 = \left\{\mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\left(\mathbf{X}^\top \mathbf{y}\right)^\top \mathbf{w}\right\} + \text{const}$$

- Initialize $\mathbf{w}$ to $\boldsymbol{w}^{(0)}$ (e.g., randomly); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. Compute the gradient $\nabla RSS(\boldsymbol{w}) = \mathbf{X}^\top(\mathbf{X}\boldsymbol{w}^{(t)} - \boldsymbol{y})$
  2. Update the parameters $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla RSS(\boldsymbol{w})$
  3. $t \leftarrow t + 1$

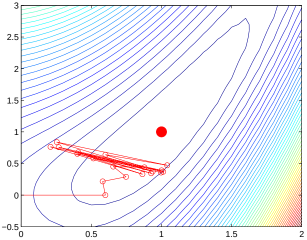


Choosing the right η is important

small η is too slow?

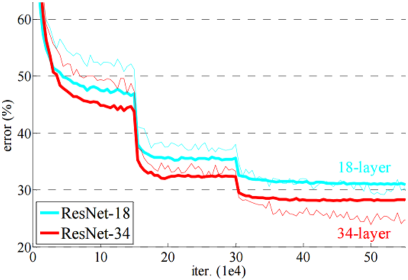


large η is too unstable?



How to choose η in practice?

- Try $0.0001, 0.001, 0.01, 0.1$ etc. on a validation dataset and choose the one that gives fastest, stable convergence
- Reduce η by a constant factor (eg. 10) when learning saturates so that we can reach closer to the true minimum.
- More advanced learning rate schedules such as AdaGrad, Adam, AdaDelta are used in practice.



Gradient descent for a general function

General form for minimizing $f(\boldsymbol{\theta})$

$$\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^{t} - \eta \frac{\partial f}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{t}}$$

- η is step size, also called the learning rate – how far we go in the direction of the negative gradient
  - Step size needs to be chosen carefully to ensure convergence.
  - Step size can be adaptive, e.g., we can use line search
- We are minimizing a function, hence the subtraction $(-\eta)$
- With a suitable choice of η, we converge to a stationary point

$$\frac{\partial f}{\partial \boldsymbol{\theta}} = 0$$

- Stationary point not always global minimum (but happy when convex)
- Popular variant called stochastic gradient descent

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 error 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

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

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

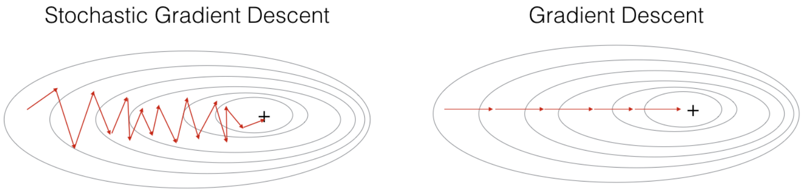
$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \sum_{n} \left\{ \sigma(\boldsymbol{w}^{(t)\top} \boldsymbol{x}_n) - y_n \right\} \boldsymbol{x}_n$$

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



- 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 $\mathbf{w}$

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

$$[\sigma(\mathbf{w}^\top \mathbf{x}_1), \sigma(\mathbf{w}^\top \mathbf{x}_2), \sigma(\mathbf{w}^\top \mathbf{x}_3), \sigma(\mathbf{w}^\top \mathbf{x}_4)]^\top = [0.996, 0.989, 0.989, 0.989]^\top$$

which can be obtained by computing $\mathbf{w}^\top \mathbf{X}$ and then apply $\sigma(\cdot)$ entrywise, which we abuse the notation and write $\sigma(\boldsymbol{X}^\top \mathbf{w})$.

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 $\mathbf{w}$

- Prediction $\sigma(\boldsymbol{X}^\top \mathbf{w}) = [0.996, 0.989, 0.989, 0.989]^\top$
- Difference from labels $\boldsymbol{y} = [1, 1, 0, 0]^\top$ is

$$\sigma(\boldsymbol{X}^\top \mathbf{w}) - \boldsymbol{y} = [-0.004, -0.011, 0.989, 0.989]^\top$$

- Gradient of the first email,

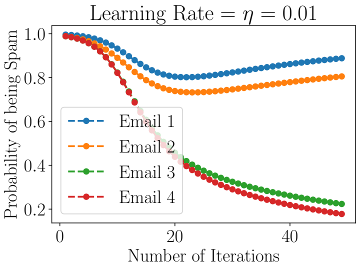
$$\boldsymbol{g}_1 = (\sigma(\mathbf{w}^\top \mathbf{x}_1) - y_1)\mathbf{x}_1 = -0.004[1, 5, 3, 1, 1]^\top$$

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

  notice the similarity with linear regression

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)

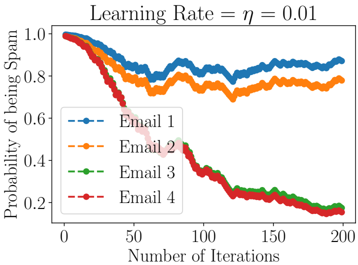
Example: Spam Classification, Stochastic Gradient Descent

	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 Naive 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 v.s. 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.

Summary

Setup for binary classification

- Logistic Regression models conditional distribution as:
  $p(y = 1|\boldsymbol{x}; \boldsymbol{w}) = \sigma[g(\boldsymbol{x})]$ where $g(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$
- Linear decision boundary: $g(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} = 0$

Minimizing cross-entropy error (negative log-likelihood)

- $\mathcal{E}(b, \boldsymbol{w}) = -\sum_n \{y_n \log \sigma(b + \boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(b + \boldsymbol{w}^\top \boldsymbol{x}_n)]\}$
- No closed form solution; must rely on iterative solvers

Numerical optimization

- Gradient descent: simple, scalable to large-scale problems
  - Move in direction opposite of gradient!
  - Gradient of the cross-entropy error takes nice form

What's next?

What about when we want to predict multiple classes?

- Dog vs. cat. vs crocodile
- Movie genres (action, horror, comedy, ...)
- Yelp ratings (1, 2, 3, 4, 5)
- Part of speech tagging (verb, noun, adjective, ...)
- ...