18-661 Introduction to Machine Learning

Logistic Regression

Spring 2019

Prof. Gauri Joshi
1. Review of Naive Bayes

2. Logistic Regression Model

3. Loss Function and Parameter Estimation

4. Second Order Methods: Newton’s method
Review of Naive Bayes
I’m going to be rich!!

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

Attention: Honorable Beneficiary,

IMMEDIATE PAYMENT NOTIFICATION VALUED AT US$10 MILLION

It is my modest obligation to write you this letter in regards to the authorization of your owed payment through our most respected financial institution (AFRI BANK PLC). I am Mr. Aminu Saleh, The Director, Foreign Operations Department, AFRI Bank Plc, NIGERIA. The British Government, in conjunction with the US GOVERNMENT, WORLD BANK, UNITED NATIONS ORGANIZATION on foreign payment matters, has empowered my bank after much consultation and consideration, to handle all foreign payments and release them to their appropriate beneficiaries with the help of a representative from Federal Reserve Bank.

To facilitate the process of this transaction, please kindly re-confirm the following information below:

1) Your full Name and Address:
2) Phones, Fax and Mobile No.:
3) Profession, Age and Marital Status:
4) Copy of any valid form of your Identification:
FROM THE DESK OF MR. AMINU SALEH
DIRECTOR, FOREIGN OPERATIONS DEPARTMENT
AFRI BANK PLC
Afribank Plaza,
14th Floor money344.jpg
51/55 Broad Street,
P.M.B 12021 Lagos-Nigeria

Attention: Honorable Beneficiary,

IMMEDIATE PAYMENT NOTIFICATION VALUED AT USS10 MILLION

Hi Virginia,

Can we meet today at 2pm?

thanks,

Carlee
Simple Strategy: count the words

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

```
<table>
<thead>
<tr>
<th>word</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>free</td>
<td>100</td>
</tr>
<tr>
<td>money</td>
<td>2</td>
</tr>
<tr>
<td>account</td>
<td>2</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>word</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>free</td>
<td>1</td>
</tr>
<tr>
<td>money</td>
<td>1</td>
</tr>
<tr>
<td>account</td>
<td>2</td>
</tr>
</tbody>
</table>
```

Just wanted to send a quick reminder about the guest lecture. We meet at MTH 185. It has a PC and LCD projector connection for your laptop if you desire. Maybe we can set up the AV stuff.

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

Thanks so much for your willingness to do this,
Mark
Weighted sum of those telltale words

different weights for spam and ham: representing how compatible the word pattern is to each category
Weighted sum of those telltale words

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

\[
\begin{pmatrix}
100 \times 0.2 \\
2 \times 0.2 \\
\vdots \\
2 \times 0.3
\end{pmatrix} = 3.2
\]

\[
\begin{pmatrix}
100 \times 0.01 \\
2 \times 0.02 \\
\vdots \\
2 \times 0.01
\end{pmatrix} = 1.03
\]
Naive Bayes model for identifying spam

- **Class label**: binary
  - \( y = \{ \text{spam}, \text{ham} \} \)

- **Features**: word counts in the document (bag-of-words)
  - \( x = \{ (\text{‘free’}, 100), (\text{‘lottery’}, 5), (\text{‘money’}, 10) \} \)
  - 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

- **Model**

\[
p(x|\text{spam}) = p(\text{‘free’}|\text{spam})^{100} p(\text{‘lottery’}|\text{spam})^{5} p(\text{‘money’}|\text{spam})^{10} \ldots
\]

**Key idea**: the parameters that we need to estimate are these conditional probabilities
Naive Bayes classification rule

For any document \( \mathbf{x} \), we want to compare \( p(\text{spam}|\mathbf{x}) \) and \( p(\text{ham}|\mathbf{x}) \).

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 we compare:

\[
\log[p(\mathbf{x}|\text{spam})p(\text{spam})] \text{ versus } \log[p(\mathbf{x}|\text{ham})p(\text{ham})]
\]
Classifier in linear form

\[
\log[p(x|\text{spam})p(\text{spam})] = \log \left[ \prod_i p(\text{word}_i|\text{spam})^{x_i} p(\text{spam}) \right] \\
= \sum_i x_i \log p(\text{word}_i|\text{spam}) + \log p(\text{spam}) \quad (1)
\]

Similarly, we have

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

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., \( p(\text{funny\_word} | \text{ham}) \)
  \[
  p(\text{funny\_word} | \text{ham}) = \frac{\# \text{of funny\_word in ham emails}}{\# \text{of words in ham emails}} \quad (3)
  \]
  \[
  p(\text{funny\_word} | \text{spam}) = \frac{\# \text{of funny\_word in spam emails}}{\# \text{of words in spam emails}} \quad (4)
  \]
- Use Laplacian smoothing (adding pseudo-counts) to avoid these probabilities begin 0 for any word
Examine the classification rule for naive Bayes

• Comparing the log likelihoods. Declare the email as ‘spam’ if

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

• Equivalent to checking whether

\[ \sum_i x_i [\log p(\text{word}_i | \text{spam}) - \log p(\text{word}_i | \text{ham})] + [\log p(\text{spam}) - \log p(\text{ham})] > 0 \]

• This is just a linear decision boundary

\[ \sum_i w_i x_i + w_0 > 0 \]
1. Review of Naive Bayes

2. Logistic Regression Model

3. Loss Function and Parameter Estimation

4. Second Order Methods: Newton’s method
Logistic Regression Model
In a general linear model,

$$y \approx w_0 + w_1 x_1 + \cdots + w_d x_d$$

the response $y_i$, is modeled by a linear function of features $x_j, j = 1, \cdots, d$ plus an error term.
Restriction of Linear Models

Although a very useful framework, there are some situations where linear models are not appropriate

- more complex $x \rightarrow y$ relationship
  - **Solution**: Non-linear basis functions
- the range of $y$ is restricted (e.g. binary, count)
  - **Solution**: Logistic Regression

$y$ is a quadratic function of $x$

$y = 0$ for dog, $y = 1$ for cat
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)
Intuition: Logistic Regression

- $x_1$ = # of times 'meet' appears in an email
- $x_2$ = # of times 'lottery' appears in an email
- Define feature vector $x = [1, x_1, x_2]$
- Learn the decision boundary $w_0 + w_1 x_1 + w_2 x_2 = 0$ such that
  - If $w^T x \geq 0$ declare $y = 1$ (spam)
  - If $w^T x < 0$ declare $y = 0$ (ham)

Key Idea: If 'meet' appears few times and 'lottery' appears many times than the email is spam
• $x_1 = \#$ 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 $\mathbf{w}^T \mathbf{x} \geq 0$ declare $y = 1$ (spam)
  • If $\mathbf{w}^T \mathbf{x} < 0$ 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?

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

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

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 $\sigma(w^T x)$ that maps $w^T x$ to a value between 0 and 1

Key Problem: Finding optimal weights $w$ that accurately predict this probability for a new email
Formal Setup: Binary Logistic Classification

- **Input:** \( x = [1, x_1, x_2, \ldots, x_D] \in \mathbb{R}^{D+1} \)
- **Output:** \( y \in \{0, 1\} \)
- **Training data:** \( \mathcal{D} = \{(x_n, y_n), n = 1, 2, \ldots, N\} \)
- **Model:**
  \[
p(y = 1|x; w) = \sigma[g(x)]
  \]

where

\[
g(x) = w_0 + \sum_d w_d x_d = w^\top x
\]

and \( \sigma[.] \) 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

\[ a = \mathbf{w}^\top \mathbf{x} \]

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
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?

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

\[
\Pr(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-0.42}} = 0.603
\]
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 \((x_n, y_n)\)

\[
p(y_n|x_n; w) = \begin{cases} 
\sigma(w^T x_n) & \text{if } y_n = 1 \\
1 - \sigma(w^T x_n) & \text{otherwise}
\end{cases}
\]

Compact expression, exploring that \(y_n\) is either 1 or 0

\[
p(y_n|x_n; w) = \sigma(w^T x_n)^{y_n} [1 - \sigma(w^T x_n)]^{1-y_n}
\]

Probability that predicted label is 1 (spam)
Log-likelihood of the whole training data $\mathcal{D}$

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

It is convenient to work with its negation, which is called

cross-entropy error function

$$
\mathcal{E}(b, w) = - \sum_n \{y_n \log \sigma(w^\top x_n) + (1 - y_n) \log[1 - \sigma(w^\top x_n)]\}
$$
We will minimize the error function

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

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: \textit{gradient descent}

Finding the gradient of \(\mathcal{E}(\mathbf{w})\) looks very hard, but it turns out to be simple and intuitive
Simple fact: derivatives of $\sigma(a)$

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

$$= \frac{-1}{(1 + e^{-a})^2} \frac{d}{d a} (1 + e^{-a})$$

$$= \frac{e^{-a}}{(1 + e^{-a})^2}$$

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

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

$$= \sigma(a)[1 - \sigma(a)]$$
Gradients of the cross-entropy error function

Cross-entropy Error Function

\[ E(w) = -\sum_n \{ y_n \log \sigma(w^\top x_n) + (1 - y_n) \log [1 - \sigma(w^\top x_n)] \} \]

Gradients

\[ \frac{\partial E(w)}{\partial w} = -\sum_n \left\{ y_n \frac{\sigma(w^\top x)}{\sigma(w^\top x)} [1 - \sigma(w^\top x_n)] x_n - (1 - y_n) \frac{1 - \sigma(w^\top x)}{1 - \sigma(w^\top x)} \sigma(w^\top x_n) x_n \right\} \]

\[ = -\sum_n \left\{ y_n [1 - \sigma(w^\top x_n)] x_n - (1 - y_n) \sigma(w^\top x_n) x_n \right\} \]

\[ = \sum_n \left\{ \sigma(w^\top x_n) - y_n \right\} x_n \]

Remark

- \( e_n = \{ \sigma(w^\top x_n) - y_n \} \) is called error for the \( n \)th training sample.
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

\[
\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \sum_n \left\{ \sigma(\mathbf{w}^\top \mathbf{x}_n) - y_n \right\} \mathbf{x}_n
\]
Example: Spam Classification

<table>
<thead>
<tr>
<th></th>
<th>free</th>
<th>bank</th>
<th>meet</th>
<th>time</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email 1</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Spam</td>
</tr>
<tr>
<td>Email 2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>Spam</td>
</tr>
<tr>
<td>Email 3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>Ham</td>
</tr>
<tr>
<td>Email 4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>Ham</td>
</tr>
</tbody>
</table>

Perform gradient descent to learn weights \( w \)

- Feature vector for email 1 \( x_1 = [1, 5, 3, 1, 1]^\top \)
- Initial weights \( w = [0.5, 0.5, 0.5, 0.5, 0.5]^\top \)
- Prediction \( \sigma(w^\top x_1) = [0.996, 0.989, 0.989, 0.989]^\top \)
Example: Spam Classification, Batch Gradient Descent

<table>
<thead>
<tr>
<th></th>
<th>free</th>
<th>bank</th>
<th>meet</th>
<th>time</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email 1</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Spam</td>
</tr>
<tr>
<td>Email 2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>Spam</td>
</tr>
<tr>
<td>Email 3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>Ham</td>
</tr>
<tr>
<td>Email 4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>Ham</td>
</tr>
</tbody>
</table>

Perform gradient descent to learn weights $\mathbf{w}$

- Prediction $\sigma(\mathbf{w}^T \mathbf{x}_1) = [0.996, 0.989, 0.989, 0.989]^T$
- Difference from labels $y = [1, 1, 0, 0]^T$ is $[-0.004, -0.011, 0.989, 0.989]^T$
- Gradient is $g_1 = (\sigma(\mathbf{w}^T \mathbf{x}_n) - y)\mathbf{x}_1 = [1.96, 2.9, 2.93, 4.93, 4.93]$
- $\mathbf{w} \leftarrow \mathbf{w} - 0.01 \sum_n g_n$
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

<table>
<thead>
<tr>
<th></th>
<th>free</th>
<th>bank</th>
<th>meet</th>
<th>time</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email 1</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Spam</td>
</tr>
<tr>
<td>Email 2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>Spam</td>
</tr>
<tr>
<td>Email 3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>Ham</td>
</tr>
<tr>
<td>Email 4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>Ham</td>
</tr>
</tbody>
</table>

• Prediction $\sigma(w^\top x_r) = 0.996$ for a randomly chosen email $r$

• Difference from label $y = 1$ is $-0.004$

• Gradient is $g_r = (\sigma(w^\top x_n) - y)x_r = [1.96, 2.9, 2.93, 4.93, 4.93]$

• $w \leftarrow w - 0.01g_r$
### Example: Spam Classification, Stochastic Gradient Descent

<table>
<thead>
<tr>
<th></th>
<th>free</th>
<th>bank</th>
<th>meet</th>
<th>time</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email 1</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Spam</td>
</tr>
<tr>
<td>Email 2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>Spam</td>
</tr>
<tr>
<td>Email 3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>Ham</td>
</tr>
<tr>
<td>Email 4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>Ham</td>
</tr>
</tbody>
</table>

Learning Rate $\eta = 0.01$

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

<table>
<thead>
<tr>
<th></th>
<th>free</th>
<th>bank</th>
<th>meet</th>
<th>time</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email 1</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Spam</td>
</tr>
<tr>
<td>Email 2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>Spam</td>
</tr>
<tr>
<td>Email 3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>Ham</td>
</tr>
<tr>
<td>Email 4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>Ham</td>
</tr>
</tbody>
</table>

- Final $\mathbf{w} = [0.187, 0.482, 0.179, -0.512, -0.524]^T$ 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
Both classification models are linear functions of features

**Joint vs. conditional dist**
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 occurrences
Logistic Regression implicitly captures correlations when training weights

**Generative vs. Discriminative**
NB is a *generative* model, LR is a *discriminative* model
Second Order Methods: Newton’s method
Approximate the true function with an easy-to-solve optimization problem

In particular, we can approximate the cross-entropy error function around $w^{(t)}$ by a quadratic function, and then minimize this quadratic function.
Approximation

Second Order Taylor expansion around $x_t$

$$f(x) \approx f(x_t) + f'(x_t)(x - x_t) + \frac{1}{2}f''(x_t)(x - x_t)^2$$

Taylor expansion of cross-entropy error function around $w^{(t)}$

$$\mathcal{E}(w) \approx \mathcal{E}(w^{(t)}) + (w - w^{(t)})^\top \nabla \mathcal{E}(w^{(t)}) + \frac{1}{2}(w - w^{(t)})^\top H^{(t)}(w - w^{(t)})$$

where

- $\nabla \mathcal{E}(w^{(t)})$ is the gradient
- $H^{(t)}$ is the Hessian matrix evaluated at $w^{(t)}$
The matrix of second-order derivatives

\[ H = \frac{\partial^2 \mathcal{E}(\mathbf{w})}{\partial \mathbf{w}\mathbf{w}^\top} \]

In other words,

\[ H_{ij} = \frac{\partial}{\partial w_j} \left( \frac{\partial \mathcal{E}(\mathbf{w})}{\partial w_i} \right) \]

So the Hessian matrix is \( \mathbb{R}^{D \times D} \), where \( \mathbf{w} \in \mathbb{R}^D \).
Optimizing the approximation

Minimize the approximation

\[ E(w) \approx E(w^{(t)}) + (w - w^{(t)})^T \nabla E(w^{(t)}) + \frac{1}{2}(w - w^{(t)})^T H^{(t)}(w - w^{(t)}) \]

and use the solution as the new estimate of the parameters

\[ w^{(t+1)} \leftarrow \min_w (w - w^{(t)})^T \nabla E(w^{(t)}) + \frac{1}{2}(w - w^{(t)})^T H^{(t)}(w - w^{(t)}) \]

The quadratic function minimization has a \textit{closed} form, thus, we have

\[ w^{(t+1)} \leftarrow w^{(t)} - \left( H^{(t)} \right)^{-1} \nabla E(w^{(t)}) \]

i.e., the Newton’s method.
Contrast gradient descent and Newton’s method

**Similar**

- Both are iterative procedures.

**Different**

- Newton’s method requires second-order derivatives.
- Newton’s method does not have the magic $\eta$ to be set.
Our cross-entropy error function is convex

\[
\frac{\partial \mathcal{E}(\mathbf{w})}{\partial \mathbf{w}} = \sum_n \{\sigma(\mathbf{w}^\top \mathbf{x}_n) - y_n\} \mathbf{x}_n
\]  \hspace{1cm} (5)

\[
\Rightarrow \mathbf{H} = \frac{\partial^2 \mathcal{E}(\mathbf{w})}{\partial \mathbf{w} \mathbf{w}^\top}
\] \hspace{1cm} (6)

For any vector \( \mathbf{v} \),

\[
\mathbf{v}^\top \mathbf{H} \mathbf{v} \geq 0
\]

Thus, positive semi-definite. Thus, the cross-entropy error function is convex, with only one global optimum.
The Good: Fast (in terms of convergence)!

Newton’s method finds the optimal point in a single iteration when the function we’re optimizing is quadratic.

In general, the better our Taylor approximation, the more quickly Newton’s method will converge.

The Bad: Not scalable!

Computing and inverting Hessian matrix can be very expensive for large-scale problems where the dimensionally $D$ is very large. There are fixes and alternatives, such as Quasi-Newton/Quasi-second order method.
Summary

Setup for 2 classes

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

Minimizing cross-entropy error (negative log-likelihood)

- \( \mathcal{E}(b, w) = -\sum_n \{y_n \log \sigma(b + w^\top x_n) + (1 - y_n) \log[1 - \sigma(b + w^\top x_n)]\} \)
- 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 logistic function takes nice form
- Newton method: fast to converge but not scalable
  - At each iteration, find optimal point in 2nd-order Taylor expansion
  - Closed form solution exists for each iteration
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, ...)
- ...