# Recitation 4
# Logistic Regression

10-301/601: Introduction to Machine Learning

02/21/2025

## 1 Closed Form Solution for Regression

We would like to fit a regression model to the dataset

$$\mathcal{D} = \left\{ \left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right), \cdots, \left(\mathbf{x}^{(N)}, y^{(N)}\right) \right\}$$

with $\mathbf{x}^{(i)} \in \mathbb{R}^M$ by minimizing the loss function:

$$J(\mathbf{w}) = \sum_{i=1}^{N} \left( y^{(i)} - \sum_{l=1}^{L} w_l \sum_{j=1}^{M} w_j x_j^{(i)} \right).$$

To find a best model match, we want to minimize the loss function with respect to the weights $\mathbf{w}$. This is just a practice in taking the first order derivative of $J(\mathbf{w})$ with respect to each $w_k \in \mathbf{w}$, setting them equal to zero, and solving for $w_k$. In other words, we set $\frac{\partial J(\mathbf{w})}{\partial w_k} = 0$ and solve for $w_k$.

(a) Solve for $\frac{\partial J(\mathbf{w})}{\partial w_k}$ and set it equal to zero.

$$\begin{aligned}
\frac{\partial J(\mathbf{w})}{\partial w_k} &= \frac{\partial}{\partial w_k} \left[ \sum_{i=1}^{N} \left( y^{(i)} - \sum_{l=1}^{L} w_l \sum_{j=1}^{M} w_j x_j^{(i)} \right) \right] \\
&= -\sum_{i=1}^{N} \left( \sum_{j=1}^{M} w_j x_j^{(i)} + x_k \sum_{l=1}^{L} w_l \right) \\
&= 0
\end{aligned}$$

(b) Fill in the blanks to solve for $w_k$.

$$0 = -\sum_{i=1}^{N}\left(\sum_{j=1}^{M} w_j x_j^{(i)} + x_k^{(i)} \sum_{l=1}^{L} w_l\right)$$

$$0 = -\sum_{i=1}^{N}\left(\text{----} + \sum_{j=1,\text{----}}^{M} w_j x_j^{(i)} + \text{----} + x_k^{(i)} \sum_{l=1,\text{----}}^{L} w_l\right)$$

$$0 = \text{-----} - \sum_{i=1}^{N}\left(\sum_{j=1,\text{----}}^{M} w_j x_j^{(i)} + x_k^{(i)} \sum_{l=1,\text{----}}^{L} w_l\right)$$

$$0 = -\sum_{i=1}^{N}\left(\sum_{j=1}^{M} w_j x_j^{(i)} + x_k \sum_{l=1}^{L} w_l\right)$$

$$0 = -\sum_{i=1}^{N}\left(w_k x_k^{(i)} + \sum_{j=1,j\neq k}^{M} w_j x_j^{(i)} + w_k x_k^{(i)} + x_k^{(i)} \sum_{l=1,l\neq k}^{L} w_l\right)$$

$$0 = -2\sum_{i=1}^{N} w_k x_k^{(i)} - \sum_{i=1}^{N}\left(\sum_{j=1,j\neq k}^{M} w_j x_j^{(i)} + x_k^{(i)} \sum_{l=1,l\neq k}^{L} w_l\right)$$

$$0 = -2w_k \sum_{i=1}^{N} x_k^{(i)} - \sum_{i=1}^{N}\left(\sum_{j=1,j\neq k}^{M} w_j x_j^{(i)} + x_k^{(i)} \sum_{l=1,l\neq k}^{L} w_l\right)$$

$$2w_k \sum_{i=1}^{N} x_k^{(i)} = -\sum_{i=1}^{N}\left(\sum_{j=1,j\neq k}^{M} w_j x_j^{(i)} + x_k^{(i)} \sum_{l=1,l\neq k}^{L} w_l\right)$$

$$w_k = \frac{-\sum_{i=1}^{N}\left(\sum_{j=1,j\neq k}^{M} w_j x_j^{(i)} + x_k^{(i)} \sum_{l=1,l\neq k}^{L} w_l\right)}{2\sum_{i=1}^{N} x_k^{(i)}}$$

# 2 Binary Logistic Regression

Consider the following dataset,

$$\mathcal{D} = \left\{ \left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(N)}, y^{(N)}\right) \right\} \text{ where } \mathbf{x}^{(i)} \in \mathbb{R}^M, y^{(i)} \in \{0, 1\}.$$

Notice that instead of regressing on continuous variables, as we do in linear regression, we wish to predict on discrete variables, and, more specifically, binary outcomes (this process is called classification). Since we want to take noise into account when making predictions, we will predict the probability of each outcome given some inputs, each denoted $\mathbf{x}^{(i)}$. This means that the result of the linear combination of each input, $\mathbf{x}^{(i)}$ with the parameter vector, $\boldsymbol{\theta}$, or, $\boldsymbol{\theta}^T\mathbf{x}^{(i)}$, must be manipulated to fit between zero and one. We use the sigmoid function, $\sigma(\cdot)$, for this purpose.

Recall that

$$\sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T\mathbf{x}^{(i)})} = \frac{\exp(\boldsymbol{\theta}^T\mathbf{x}^{(i)})}{1 + \exp(\boldsymbol{\theta}^T\mathbf{x}^{(i)})}.$$

The conditional probability of $y^{(i)}$ given $\mathbf{x}^{(i)}$ is,

$$p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}\right) = \begin{cases} \sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)}) & y^{(i)} = 1 \\ 1 - \sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)}) & y^{(i)} = 0 \end{cases}.$$

We can rewrite this as a single statement,

$$p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}\right) = \sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)})^{y^{(i)}}(1 - \sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)}))^{(1-y^{(i)})}.$$

Can you show why this is true?

if $y^{(i)} = 1$:

$$p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}\right) = \sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)})^1(1 - \sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)}))^{(1-1)} = \sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)})$$

if $y^{(i)} = 0$:

$$p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}\right) = \sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)})^0(1 - \sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)}))^{(1-0)} = 1 - \sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)})$$

This is the same result as the piecewise function above.

We assume each observation of $y^{(i)}$ in the data is independent and identically distributed. This means we can write down the likelihood of the data as a product of negative conditional probabilities.

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\log(\prod_{i=1}^{N} p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}\right))$$

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N} \log(p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}\right))$$

Let's plug in the expression for $p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}\right)$ and simplify.

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N} \log\left(\sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)})^{y^{(i)}}(1-\sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)}))^{(1-y^{(i)})}\right)$$
$$= -\frac{1}{N}\sum_{i=1}^{N}(y^{(i)}\log\left(\sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-\sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)})\right))$$

In stochastic gradient descent, we use only a single $\mathbf{x}^{(i)}$. Given $\boldsymbol{\phi}^{(i)} = \sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)})$ and

$$J^{(i)}(\boldsymbol{\theta}) = -y^{(i)}\log(\boldsymbol{\phi}^{(i)}) - (1-y^{(i)})\log(1-\boldsymbol{\phi}^{(i)})$$

Show that the partial derivative of $J^{(i)}(\boldsymbol{\theta})$ with respect to the $j$th parameter $\theta_j$ is as follows:

$$\frac{\partial J^{(i)}(\boldsymbol{\theta})}{\partial \theta_j} = (\sigma(\boldsymbol{\theta}^T\mathbf{x}^{(i)}) - y^i)x_j^{(i)}$$

Remember,

$$\frac{\partial \boldsymbol{\phi}^{(i)}}{\partial \boldsymbol{\theta}_j} = \boldsymbol{\phi}^{(i)} * (1-\boldsymbol{\phi}^{(i)}) * \frac{\partial \boldsymbol{\theta}^T\mathbf{x}^{(i)}}{\partial \theta_j}$$

note

$$\frac{\partial \boldsymbol{\theta}^T \mathbf{x}^{(i)}}{\partial \theta_j} = \mathbf{x}_j^{(i)}$$

$$\begin{aligned}
\frac{\partial J^{(i)}(\boldsymbol{\theta})}{\partial \theta_j} &= -\frac{y^{(i)}}{\boldsymbol{\phi}^{(i)}}\frac{\partial \boldsymbol{\phi}^{(i)}}{\partial \boldsymbol{\theta}_j} - \frac{(1-y^{(i)})}{1-\boldsymbol{\phi}^{(i)}}\frac{\partial (1-\boldsymbol{\phi}^{(i)})}{\partial \boldsymbol{\theta}_j} \\
&= -\frac{y^{(i)}}{\boldsymbol{\phi}^{(i)}}\frac{\partial \boldsymbol{\phi}^{(i)}}{\partial \boldsymbol{\theta}_j} + \frac{(1-y^{(i)})}{1-\boldsymbol{\phi}^{(i)}}\frac{\partial \boldsymbol{\phi}^{(i)}}{\partial \boldsymbol{\theta}_j} \\
&= -\frac{y^{(i)}}{\boldsymbol{\phi}^{(i)}}\boldsymbol{\phi}^{(i)} * (1-\boldsymbol{\phi}^{(i)}) * \frac{\partial \boldsymbol{\theta}^T \mathbf{x}^{(i)}}{\partial \theta_j} + \frac{(1-y^{(i)})}{1-\boldsymbol{\phi}^{(i)}}\boldsymbol{\phi}^{(i)} * (1-\boldsymbol{\phi}^{(i)}) * \frac{\partial \boldsymbol{\theta}^T \mathbf{x}^{(i)}}{\partial \theta_j} \\
&= (-y^{(i)}(1-\boldsymbol{\phi}^{(i)}) + (1-y^{(i)})\boldsymbol{\phi}^{(i)}))\mathbf{x}_j^{(i)} \\
&= (-y^{(i)} + y^{(i)}\boldsymbol{\phi}^{(i)} + \boldsymbol{\phi}^{(i)} - y^{(i)}\boldsymbol{\phi}^{(i)})\mathbf{x}_j^{(i)} \\
&= (\boldsymbol{\phi}^{(i)} - y^{(i)})\mathbf{x}_j^{(i)} \\
&= (\sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)}) - y^i)\mathbf{x}_j^{(i)}
\end{aligned}$$

.

# 3 Feature Representation for Sentiment Classification

In many machine learning problems, we will want to find appropriate representations for the inputs of the algorithm we are developing. In Homework 4, we will work on using logistic regression for a sentiment classification task, where our algorithm takes a paragraph of movie review as the input and outputs a binary value denoting whether the review is positive or not. To build an appropriate representation for the input (aka. the review text), we consider a representation built using GloVe[1] word embeddings.

In this section, consider a scenario where we are interested in representing the following text:

```
a hot dog is not a sandwich because it is not square
```
(1)

We consider the following dictionary (denoted below as **Vocab**) as the set of vocabulary that we will consider. Note that the vocabulary dictionary might not contain all words in the text shown above.

```
dictionary = {
    "the": 0,
    "square": 1,
    "hot": 2,
    "is": 3,
    "not": 4,
    "a": 5,
    "happy": 6,
    "sandwich": 7
}
```

[1] You can read more about GloVe in the original research paper. You can also check out this explainer on Word2Vec, which is a similar technique for obtaining word embeddings.

1. **Word Embedding Based Representation**

   (a) Word embeddings are reduced dimension vector representations (features) of words. Given a single word in the dictionary, word embeddings can convert it to a vector of fixed dimension. In Homework 4, we will provide a dictionary file specifying pre-computed mappings between every word in **Vocab** and their corresponding word embeddings. To facilitate better understanding towards word embeddings, we produce a plot showing the spatial relationship between several sample words from the vocabulary used in Homework 4, with their corresponding word embeddings (reduced to 2D vectors from 300D vectors using a technique called PCA we will learn about later in this course!):

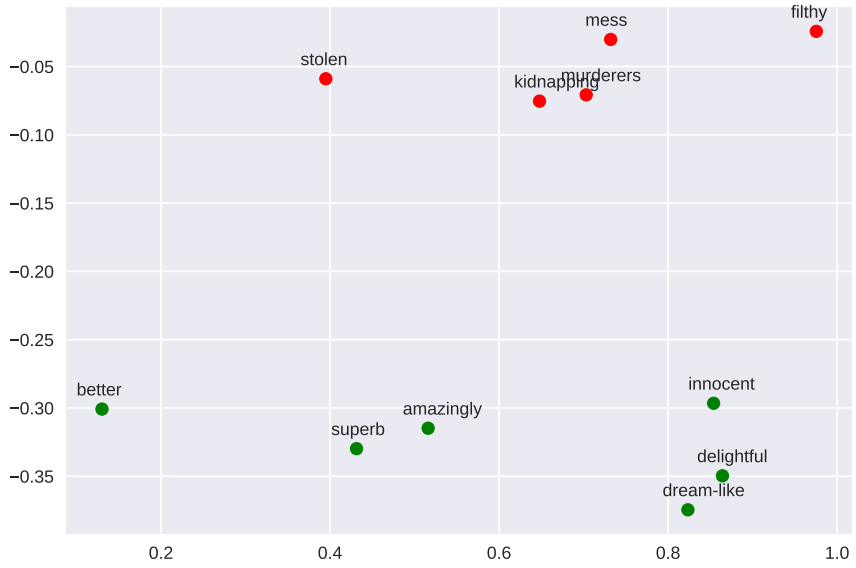


Figure 1: Visualization of word embeddings. We select a few positive words (shown in green) and a few negative words (shown in red). To make the plot, we map the high-dimensional word representations of these words to 2D space using PCA and then visualize them in the scatter plot above.

Please comment on your observations and findings based on this plot.

Closer-related words are located closer in the representation space, while farther-related words are located farther from each other.

(b) Now, we must translate these word embeddings to sentence embeddings (a vector representing the sentence as a whole). One approach to building a sentence embedding is to average out the vector representation of every word in the sentence that is in the dictionary. For example, given text "`a hot dog flies like a sandwich`", we can find the sentence embedding for this text by taking the average of the vector representation of the words "`a`", "`hot`", "`a`", and "`sandwich`".

Now suppose we have the following word embedding dictionary for building sentence embeddings (this is a toy example used for illustrative purposes; actual word embeddings will have higher dimensions than this example):

```
dictionary = {
    "the": [0.2, 0.3],
    "square": [0.8, 0.9],
    "hot": [0.1, -0.2],
    "is": [0.1, 0.1],
    "not": [-0.2, -0.3],
    "a": [0.0, 0.0],
    "happy": [0.4, 0.4],
    "sandwich": [0.2, -0.3]
}
```

Write the word embedding based representation of the sample text defined above, repeated here for convenience:

`a hot dog is not a sandwich because it is not square` (2)

$$\begin{aligned}\boldsymbol{\phi}_2(\mathbf{x}) &= \frac{1}{9}\big(f(\text{square}) + f(\text{hot}) + 2\cdot f(\text{is}) + 2\cdot f(\text{not}) + 2\cdot f(\text{a}) + f(\text{sandwich})\big) \\ &= \begin{bmatrix}0.1 & 0.0\end{bmatrix}^T.\end{aligned}$$

# 4 Gradient Descent and Stochastic Gradient Descent

Now we will compare two different optimization methods using pseudocode. Consider a model with parameter $\boldsymbol{\theta} \in \mathbb{R}^M$ being trained with a design matrix $\mathbf{X} \in \mathbb{R}^{N \times M}$ and labels $\mathbf{y} \in \mathbb{R}^N$. Say we update $\boldsymbol{\theta}$ using the objective function $J(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta}|\mathbf{x}^{(i)}, y^{(i)}) \in \mathbb{R}$. Recall that an epoch refers to one complete cycle through the dataset.

(a) Complete the pseudocode for gradient descent.

```
def dJ(theta, X, y, i):
    (omitted) # Returns ∂J^(i)(θ|x^(i), y^(i))/∂θ
    # You may call this function in your pseudocode.

def GD(theta, X, y, learning_rate):
    for epoch in range(num_epoch):
          Complete this section with the update rule
    return theta # return the updated theta
```

```
grad = zeros(M)
for i in range(N):
    grad += dJ(theta, X, y, i)
theta -= learning_rate * grad / N
```

(b) Complete the pseudocode for stochastic gradient descent that samples *without* replacement.

```
def dJ(theta, X, y, i):
    (omitted) # Returns ∂J^(i)(θ|x^(i), y^(i))/∂θ
    # You may call this function in your pseudocode.

def SGD(theta, X, y, learning_rate):
    for epoch in range(num_epoch):
        indices = shuffle(range(len(X)))
        for i in indices:
              Complete this section with the update rule
    return theta # return the updated theta
```

```
theta -= learning_rate * dJ(theta, X, y, i)
```

# 5 Logistic Regression: Toy Example

Let's go through a toy problem.

| Y | $X_1$ | $X_2$ | $X_3$ |
|---|---|---|---|
| 1 | 1 | 2 | 1 |
| 1 | 1 | 1 | -1 |
| 0 | 1 | -2 | 1 |

(a) What is $J(\boldsymbol{\theta})$ of above data given initial $\boldsymbol{\theta} = \begin{bmatrix} -2 \\ 2 \\ 1 \end{bmatrix}$?

$$J(\boldsymbol{\theta}) = \frac{-1}{N} \sum y^i \log(\sigma(\theta^T x^{(i)})) + (1 - y^i) \log(1 - \sigma(\theta^T x^{(i)}))$$

$$J(\boldsymbol{\theta}) = -\frac{1}{3}[\log(\sigma(3)) + \log(\sigma(-1)) + \log(1 - \sigma(-5))] \approx 0.46$$

(b) Calculate $\frac{\partial J^{(1)}(\boldsymbol{\theta})}{\partial \theta_1}$, $\frac{\partial J^{(1)}(\boldsymbol{\theta})}{\partial \theta_2}$ and $\frac{\partial J^{(1)}(\boldsymbol{\theta})}{\partial \theta_3}$ for first training example. Note that $\sigma(3) \approx 0.95$.

$$\frac{\partial J^{(i)}(\boldsymbol{\theta})}{\partial \theta_j} = x_j^{(i)}(\sigma(\theta^T x^{(i)})) - y^{(i)})$$

$$\frac{\partial J^{(1)}(\boldsymbol{\theta})}{\partial \theta_1} = (\sigma(3) - 1)1 = -0.05$$

$$\frac{\partial J^{(1)}(\boldsymbol{\theta})}{\partial \theta_2} = (\sigma(3) - 1)2 = -0.10$$

$$\frac{\partial J^{(1)}(\boldsymbol{\theta})}{\partial \theta_3} = (\sigma(3) - 1)1 = -0.05$$

(c) Calculate $\frac{\partial J^{(2)}(\boldsymbol{\theta})}{\partial \theta_1}$, $\frac{\partial J^{(2)}(\boldsymbol{\theta})}{\partial \theta_2}$ and $\frac{\partial J^{(2)}(\boldsymbol{\theta})}{\partial \theta_3}$ for second training example. Note that $\sigma(-1) \approx 0.25$.

$$\frac{\partial J^{(2)}(\boldsymbol{\theta})}{\partial \theta_1} = (\sigma(-1) - 1)1 = -0.75$$
$$\frac{\partial J^{(2)}(\boldsymbol{\theta})}{\partial \theta_2} = (\sigma(-1) - 1)1 = -0.75$$
$$\frac{\partial J^{(2)}(\boldsymbol{\theta})}{\partial \theta_3} = (\sigma(-1) - 1) - 1 = 0.75$$

(d) Assuming we are doing stochastic gradient descent with a learning rate of 1.0, what are the updated parameters $\boldsymbol{\theta}$ if we update $\boldsymbol{\theta}$ using the second training example?

$$\begin{bmatrix} -2 \\ 2 \\ 1 \end{bmatrix} - 1 \begin{bmatrix} -0.75 \\ -0.75 \\ 0.75 \end{bmatrix} = \begin{bmatrix} -1.25 \\ 2.75 \\ 0.25 \end{bmatrix}$$

(e) What is the new $J(\boldsymbol{\theta})$ after doing the above update? Would you expect it to decrease or increase?

$J(\boldsymbol{\theta}) = 0.09$

It should decrease for logistic classifier to learn.

(f) Given a test example where $(X_1 = 1, X_2 = 3, X_3 = 4)$, what will the classifier output following this update?

$\boldsymbol{\theta}^T X = -1.25 * 1 + 2.75 * 3 + 0.25 * 4 = 8$

$\sigma(\boldsymbol{\theta}^T X) = \sigma(8) \approx 0.999 > 0.5 \implies Y = 1$