Text Classification with Logistic Regression

COM4513/6513 Natural Language Processing

Nikos Aletras
n.aletras@sheffield.ac.uk
@nikeletras

Computer Science Department

Week 2
Spring 2020
In previous lecture…
As far as I’m concerned, this is Lynch at his best. ‘Lost Highway’ is a dark, violent, surreal, beautiful, hallucinatory masterpiece. 10 out of 10 stars.
Documents: Document-Word Matrix (Bag-of-Words)

- A matrix $X$, $|D| \times |\mathcal{V}|$ where rows are documents in corpus $D$, and columns are vocabulary words in $\mathcal{V}$.
- For each document, count how many times words $w \in \mathcal{V}$ appear in it.
Documents: Document-Word Matrix (Bag-of-Words)

- A matrix $X$, $|D| \times |\mathcal{V}|$ where rows are documents in corpus $D$, and columns are vocabulary words in $\mathcal{V}$.

- For each document, count how many times words $w \in \mathcal{V}$ appear in it.

<table>
<thead>
<tr>
<th></th>
<th>bad</th>
<th>good</th>
<th>great</th>
<th>terrible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc 1</td>
<td>14</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Doc 2</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Doc 3</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

- $X$ can also be obtained by adding all the one-hot vectors of the words in the documents and then transpose!
Penalise words appearing in many documents.

Multiply word frequencies with their inverted document frequencies:

\[ idf_w = \log_{10} \frac{N}{df_w} \]

where \( N \) is the number of documents in the corpus, \( df_w \) is document frequency of word \( w \)

To obtain:

\[ x_{id} = tf_{id} \log_{10} \frac{N}{df_{id}} \]

We can also squash the raw frequency, by using the \( \log_{10} \).
In this lecture...

- Our first NLP problem: **Text Classification**
- How to train and evaluate a **Logistic Regression** classifier for text classification
In this lecture...

- Our first NLP problem: **Text Classification**
- How to train and evaluate a **Logistic Regression** classifier for text classification
- Get fully prepared for Assignment 1!
Text classification

A very common problem in NLP:
Given a piece of text, assign a label from a predefined set of labels.
Text classification

A very common problem in NLP:
Given a piece of text, assign a label from a predefined set of labels

What could the labels be?
Label Types

- positive vs negative (e.g. sentiment in reviews)
- topics (e.g. sports vs. politics)
- author name (author identification)
- pass or fail in essay grading
- supporting a stance or not
- any task with a finite set of classes!
Label Types in a more abstract way...

- **Binary** (0 or 1), e.g. a movie review is positive or negative
Label Types in a more abstract way...

- **Binary** (0 or 1), e.g. a movie review is positive or negative
- **Multi-class** (1 out of k classes), e.g. what is the topic of a news article (one out of sports, politics, business or technology)
Label Types in a more abstract way...

- **Binary** (0 or 1), e.g. a movie review is positive or negative
- **Multi-class** (1 out of k classes), e.g. what is the topic of a news article (one out of sports, politics, business or technology)
- **Multi-label** (n out of k classes), e.g. what are the topics of a news article (zero or more out of sports, politics, business or technology)

Real number, predict the average review score of a movie between 1 and 5 (formally called regression). In this lecture, we focus only on binary and multi-class problems.
Label Types in a more abstract way...

- **Binary** (0 or 1), e.g. a movie review is positive or negative
- **Multi-class** (1 out of k classes), e.g. what is the topic of a news article (one out of sports, politics, business or technology)
- **Multi-label** (n out of k classes), e.g. what are the topics of a news article (zero or more out of sports, politics, business or technology)
- **Real number**, predict the average review score of a movie between 1 and 5 (formally called regression).
Label Types in a more abstract way...

- **Binary** (0 or 1), e.g. a movie review is positive or negative
- **Multi-class** (1 out of k classes), e.g. what is the topic of a news article (one out of sports, politics, business or technology)
- **Multi-label** (n out of k classes), e.g. what are the topics of a news article (zero or more out of sports, politics, business or technology)
- **Real number**, predict the average review score of a movie between 1 and 5 (formally called regression).
- In this lecture, we focus only on binary and multi-class problems.
Text Types

- news articles
- social media posts
- legal, biomedical text
- any type of text!
A little test...

Given the following **tweets** labelled with **sentiment**:

<table>
<thead>
<tr>
<th>Label</th>
<th>Tweet</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>Very sad about Joe.</td>
</tr>
<tr>
<td>negative</td>
<td>No Sat off...Have to work 6 days a week.</td>
</tr>
<tr>
<td>negative</td>
<td>I’m a sad panda today.</td>
</tr>
<tr>
<td>positive</td>
<td>Such a beautiful satisfying day of shopping. Loves it.</td>
</tr>
<tr>
<td>positive</td>
<td>Who else is in a happy mood??</td>
</tr>
<tr>
<td>positive</td>
<td>Actually quite happy today.</td>
</tr>
</tbody>
</table>
A little test...

Given the following tweets labelled with sentiment:

<table>
<thead>
<tr>
<th>Label</th>
<th>Tweet</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>Very sad about Joe.</td>
</tr>
<tr>
<td>negative</td>
<td>No Sat off...Have to work 6 days a week.</td>
</tr>
<tr>
<td>negative</td>
<td>I’m a sad panda today.</td>
</tr>
<tr>
<td>positive</td>
<td>Such a beautiful satisfying day of shopping. Loves it.</td>
</tr>
<tr>
<td>positive</td>
<td>Who else is in a happy mood??</td>
</tr>
<tr>
<td>positive</td>
<td>Actually quite happy today.</td>
</tr>
</tbody>
</table>

- What features should be indicative of positive/negative class?
Given the following **tweets** labelled with **sentiment**:

<table>
<thead>
<tr>
<th>Label</th>
<th>Tweet</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>Very sad about Joe.</td>
</tr>
<tr>
<td>negative</td>
<td>No Sat off...Have to work 6 days a week.</td>
</tr>
<tr>
<td>negative</td>
<td>I’m a sad panda today.</td>
</tr>
<tr>
<td>positive</td>
<td>Such a beautiful satisfying day of shopping. Loves it.</td>
</tr>
<tr>
<td>positive</td>
<td>Who else is in a happy mood??</td>
</tr>
<tr>
<td>positive</td>
<td>Actually quite happy today.</td>
</tr>
</tbody>
</table>

- What **features** should be **indicative of** positive/negative **class**?
- Would they generalize to unseen data?
Given the following **tweets** labelled with **sentiment**:

<table>
<thead>
<tr>
<th>Label</th>
<th>Tweet</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>Very sad about Joe.</td>
</tr>
<tr>
<td>negative</td>
<td>No Sat off...Have to work 6 days a week.</td>
</tr>
<tr>
<td>negative</td>
<td>I’m a sad panda today.</td>
</tr>
<tr>
<td>positive</td>
<td>Such a beautiful satisfying day of shopping. Loves it.</td>
</tr>
<tr>
<td>positive</td>
<td>Who else is in a happy mood??</td>
</tr>
<tr>
<td>positive</td>
<td>Actually quite happy today.</td>
</tr>
</tbody>
</table>

- What **features** should be **indicative of** positive/negative **class**?
- Would they generalize to unseen data?
- Let’s see how we can learn from data to predict class labels and class important features!
Supervised Learning
Imagine you want to prepare for an exam in a module.
Supervised Learning - Exam Analogy

Imagine you want to prepare for an exam in a module.

- Your **training data** consists of only of all the available past exam papers.
- During training (studying), you learn by studying past exam papers.
Supervised Learning - Exam Analogy

Imagine you want to prepare for an exam in a module.

- Your **training data** consists of only all the available past exam papers.
- During training (studying), you learn by studying past exam papers.
- You can test yourself by holding out a number of past exams (**development/validation set**).
Imagine you want to prepare for an exam in a module.

- Your **training data** consists of only of all the available past exam papers.
- During training (studying), you learn by studying past exam papers.
- You can test yourself by holding out a number of past exams (**development/validation set**).
- Evaluation is performed on the exam day (**test data**)! Your score is computed by your examiner.
Supervised Learning

Given a set of $M$ training pairs of documents $x$ (vectors!) and correct class labels $y$:

$$D_{train} = \{(x^1, y^1) \ldots (x^M, y^M)\}$$

Learn a function (or model or classifier) $f$ with parameters $w$ to predict the labels $\hat{y}$ of any new/unseen document $x$ such that:

$$\hat{y} = f(x, w)$$
Binary Logistic Regression

- Assume a document vector represented with counts over $N$ words/features, $\mathbf{x} \in \mathbb{R}^N$.
- Our first classifier is a **linear model** where each element in $\mathbf{x}$ is associated with a weight $w_i$, called **Logistic Regression**. It’s actually a classification algorithm!
Binary Logistic Regression

- Assume a document vector represented with counts over $N$ words/features, $x \in \mathbb{R}^N$.
- Our first classifier is a linear model where each element in $x$ is associated with a weight $w_i$, called Logistic Regression. It’s actually a classification algorithm!
Binary Logistic Regression

- Assume a document vector represented with counts over $N$ words/features, $\mathbf{x} \in \mathbb{R}^N$.

- Our first classifier is a **linear model** where each element in $\mathbf{x}$ is associated with a weight $w_i$, called **Logistic Regression**. It’s actually a classification algorithm!

- How to predict the **class** $\hat{y}$, e.g. positive 1 or negative 0 sentiment, together with the **probability** for each class?
Logistic Regression Overview

\[ \hat{y} = \sigma(z) \]

where \[z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b\]

Input (Doc) \[\xrightarrow{x_1} w_1, \xrightarrow{x_2} w_2, \xrightarrow{x_3} w_3\]

Bias \[b\]

Activation function \[\sigma(z)\]

Output \[\hat{y}\]
Binary Logistic Regression

- Compute the dot product $z$ between the input vector $\mathbf{x}$ and the weight vector $\mathbf{w}$, and add a bias term $b$ (often ignored):

\[ z = \mathbf{w} \cdot \mathbf{x} + b \]
Compute the dot product $z$ between the input vector $\mathbf{x}$ and the weight vector $\mathbf{w}$, and add a bias term $b$ (often ignored):

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

Compute the probability of the positive class using the sigmoid function $\sigma(\cdot)$:

$$P(y = 1|\mathbf{x}; \mathbf{w}) = \sigma(z) = \frac{1}{1 + \exp(-z)}$$
Binary Logistic Regression

- Compute the dot product $z$ between the input vector $\mathbf{x}$ and the weight vector $\mathbf{w}$, and add a bias term $b$ (often ignored):

$$ z = \mathbf{w} \cdot \mathbf{x} + b $$

- Compute the probability of the positive class using the sigmoid function $\sigma(\cdot)$:

$$ P(y = 1|\mathbf{x}; \mathbf{w}) = \sigma(z) = \frac{1}{1 + \exp(-z)} $$

- Predict the class with the highest probability:

$$ \hat{y} := \begin{cases} 0 & \text{if } P(y = 1|\mathbf{x}; \mathbf{w}) < 0.5 \\ 1 & \text{otherwise} \end{cases} $$
Sigmoid function

Squashes a value $x$ between 0 and 1 (or a vector, applied independently to each element).
More than one labels: $y \in \mathcal{Y} = \{0, \ldots, k\}$. E.g. is a news article about sports ($y = 0$), politics ($y = 1$) or technology ($y = 2$).
Multiclass Logistic Regression

- More than one labels: \( y \in \mathcal{Y} = \{0, \ldots, k\} \). E.g. is a news article about sports \((y = 0)\), politics \((y = 1)\) or technology \((y = 2)\)

- A matrix of weights \( W, k \times n \) where \( k \) is the number of classes and \( n \) is the number of features in the input vector.

- Resulting into a vector of weights per class \( y \)
Multiclass Logistic Regression

- Compute the product between the input vector $x$ and the weight matrix $W$, and add a bias vector $b \in 1^k$ of ones (often ignored) to the resulting vector $z$:

$$z = W^T x + b$$
Multiclass Logistic Regression

- Compute the product between the input vector $\mathbf{x}$ and the weight matrix $W$, and add a bias vector $\mathbf{b} \in 1^k$ of ones (often ignored) to the resulting vector $\mathbf{z}$:

$$\mathbf{z} = W^T \mathbf{x} + \mathbf{b}$$

- Compute the probability for each class $c$ using the softmax function:

$$P(y = c|\mathbf{x}; W) = \text{softmax}(\mathbf{z}) = \frac{\exp(z_c)}{\sum_{i \in \mathcal{Y}} \exp(z_i)}$$
Multiclass Logistic Regression

- Compute the product between the input vector $\mathbf{x}$ and the weight matrix $W$, and add a bias vector $\mathbf{b} \in 1^k$ of ones (often ignored) to the resulting vector $\mathbf{z}$:

$$\mathbf{z} = W^T \mathbf{x} + \mathbf{b}$$

- Compute the probability for each class $c$ using the softmax function:

$$P(y = c|\mathbf{x}; W) = \text{softmax}(\mathbf{z}) = \frac{\exp(z_c)}{\sum_{i \in \mathcal{Y}} \exp(z_i)}$$

- Predict the class with the highest probability:

$$\hat{y} := \arg \max_c P(y = c|\mathbf{x}; W)$$
Softmax function

Squashes the values of a vector between 0 and 1 and the elements add up to 1 resulting into a probability distribution:

\[
\begin{bmatrix}
1.2 \\
0.9 \\
0.4
\end{bmatrix} \rightarrow \text{softmax} \rightarrow \begin{bmatrix}
0.46 \\
0.34 \\
0.20
\end{bmatrix}
\]
Wait...

How can we learn the weights?!
We start by initialising $w$ with 0s.
Training

- We start by initialising $\mathbf{w}$ with 0s.
- We adjust $\mathbf{w}$ by iterating over training data:

$$D_{\text{train}} = \{(x^1, y^1) \ldots (x^M, y^M)\}$$

- But how can we decide how much we adjust the weights?
Training

- We start by initialising $w$ with 0s.
- We adjust $w$ by iterating over training data:

$$D_{train} = \{(x_1, y_1), \ldots, (x_M, y_M)\}$$

- But how can we decide how much we adjust the weights?
- We need a function that tells us the difference between predicted and true labels (loss or cost or objective function)!
Training

- We start by initialising \( \mathbf{w} \) with 0s.
- We adjust \( \mathbf{w} \) by iterating over training data:

\[
D_{train} = \{(x^1, y^1) \ldots (x^M, y^M)\}
\]

- But how can we decide how much we adjust the weights?
- We need a function that tells us the difference between predicted and true labels (\textbf{loss or cost or objective function})!
- So we can keep iterating over the training data and adjust the weights towards minimising the loss!
We assume that the weights $w$ should maximise the log-likelihood of the correct class:

$$\log(P(y_i = c| x_i; w))$$

Since we want to minimise a loss function, we take the negative of the log-likelihood:

$$L_{BCE} = -y \cdot \log(P(y = 1)) - (1 - y)\log(1 - P(y = 1))$$

Cross-entropy loss increases as the predicted probability diverges from the actual label.

Note that $\log$ is a natural logarithm.
Categorical Cross-Entropy Loss

- To extend into multi-class, we just need to compute the negative log-likelihood of the true class $y_c$:

$$L_{CE} = -y_c \cdot \log(P(y_c = 1|x_i; W_c))$$

- The loss of all other classes is 0

- $y_c$ is either 0 or 1, $c$ takes values from 1, ..., $k$ (number of classes)
Numerical Optimisation is the research field that studies how to max/minimise the value of a function $f(w)$ by changing $w$.

In our case (and a lot of supervised machine learning):

$$ \mathbf{w}^* = \arg\min_w L(\mathbf{w}; x_i; y_i) $$
A simpler case..

Binary logistic regression has one parameter per feature → many parameters!

Let’s look at a simpler case:

$$x^* = \arg \min_{x \in \mathbb{R}} f(x), \quad f(x) = x^2$$
How can we use the knowledge that $f(x) = x^2$ in selecting points?
Gradient-based optimisation

- How can we use the knowledge that $f(x) = x^2$ in selecting points?
- Gradient of the function $f$ w.r.t to parameter $x$:

  $$\nabla_x f(x)$$

- What does it mean if the gradient at $x_k$ is 0?
  Reached a minimum, no single direction to go
Gradient-based optimisation

- How can we use the knowledge that $f(x) = x^2$ in selecting points?
- Gradient of the function $f$ w.r.t to parameter $x$:

\[ \nabla_x f(x) \]

- When evaluated at $x_k$:
  - $\text{sign}(\nabla_x f(x))$ tells us if by increasing $x$, $f(x)$ will increase (+) or decrease (-)
  - $|\nabla_x f(x)|$ tells us how fast the in/decrease will be
Gradient-based optimisation

- How can we use the knowledge that $f(x) = x^2$ in selecting points?
- Gradient of the function $f$ w.r.t to parameter $x$:
  \[ \nabla_x f(x) \]
- When evaluated at $x_k$:
  - $\text{sign}(\nabla_x f(x))$ tells us if by increasing $x$, $f(x)$ will increase (+) or decrease (-)
  - $|\nabla_x f(x)|$ tells us how fast the in/decrease will be
- What does it mean if the gradient at $x_k$ is 0?
  - Reached a minimum, no single direction to go
Gradient-based optimisation

\[ f(x) = x^2 \]
\[ \nabla_x f(x) = 2x \]

\( f(x) \) is convex, thus if \( \nabla_x f(x_k) = 0 \) then \( x_k = \arg \min_{x \in \mathbb{R}} f(x) \)
Gradient of the Loss wrt to the weights

The gradient of the loss wrt to the parameter vector $\mathbf{w} \in \mathbb{R}^n$ is decomposed into the partial derivatives wrt to each parameter $w_k$:

$$
\nabla_{\mathbf{w}} L(\mathbf{w}; \mathbf{x}_i; y_i) = \left( \frac{\partial \log P(y|x_i; \mathbf{w})}{\partial w_1}, \ldots, \frac{\partial \log P(y|x_i; \mathbf{w})}{\partial w_n} \right)
$$

$$
\frac{\partial L(\mathbf{w}; \mathbf{x}_i; y_i)}{\partial w_j} = \frac{\partial \log P(y|x_i; \mathbf{w})}{\partial w_j}
$$

$$
= \ldots
$$

$$
= (P(y|x_i; \mathbf{w}) - y_i) \cdot \mathbf{x}_i
$$

You can find more details on deriving the gradients here.
Stochastic Gradient Descent (SGD)

**Input:** $D_{\text{train}} = \{(x_1, y_1)\ldots(x_M, y_M)\}$, $D_{\text{val}} = \{(x_1, y_1)\ldots(x_D, y_D)\}$, learning rate $\eta$, epochs $e$, tolerance $t$

initialize $\mathbf{w}$ with zeros

for each epoch $e$ do

    randomise order in $D_{\text{train}}$

    for each $(x_i, y_i)$ in $D_{\text{train}}$ do

        update $\mathbf{w} = \mathbf{w} - \eta \nabla_w L(\mathbf{w}; x_i; y_i)$

    monitor training and validation loss

    if previous validation loss $-$ current validation loss; smaller than $t$

        break

return $\mathbf{w}$
Stochastic Gradient Descent (SGD) for Multi-class

- You compute the gradient for the weights of the correct class only.
- Gradient is computed as in the binary case.
Other Gradient Descent Optimisers

- **Gradient Descent**: computes the gradient of the loss function with respect to the parameters for the entire training set.
- **Batch Gradient Descent**: computes the gradient of the loss function with respect to the parameters for small parts of the training set.
- You can find more details in this blog post.
Any model with a lot of features is prone to **overfitting** its training data: high training accuracy, low test accuracy.
Any model with a lot of features is prone to **overfitting** its training data: high training accuracy, low test accuracy.

Avoid it by adding a regulariser in the objective:

$$L_{\text{reg}} = L + \alpha R(w)$$
Regularisation

- Any model with a lot of features is prone to **overfitting** its training data: high training accuracy, low test accuracy.
- Avoid it by adding a regulariser in the objective:

\[ L_{\text{reg}} = L + \alpha R(w) \]

- If \( R(w) = \sum_{k=1}^{K} w_k^2 \) (L2-regularisation) then:

\[ \frac{\partial L_{\text{reg}}(w; D_{\text{train}})}{\partial w_k} = \frac{\partial L(w; D_{\text{train}})}{\partial w_k} + 2\alpha w_k \]

- \( \alpha \) is the regularisation strength
Regularisation

- Any model with a lot of features is prone to **overfitting** its training data: high training accuracy, low test accuracy.
- Avoid it by adding a regulariser in the objective:

  \[
  L_{\text{reg}} = L + \alpha R(w)
  \]

- If \( R(w) = \sum_{k=1}^{K} w_k^2 \) (\(L_2\)-regularisation) then:

  \[
  \frac{\partial L_{\text{reg}}(w; D_{\text{train}})}{\partial w_k} = \frac{\partial L(w; D_{\text{train}})}{\partial w_k} + 2\alpha w_k
  \]

- \( \alpha \) is the regularisation strength
- Intuitively: prefer small parameter values, by not updating as much.
Other popular supervised ML algorithms

- Perceptron
- Support Vector Machines
- Naive Bayes
- Neural Networks (Week 8)
- Gaussian Processes
Evaluation

- The standard way to evaluate a classifier is:

\[ \text{Accuracy} = \frac{\# \text{correctly classified}}{\# \text{all documents}} \]

- What could go wrong?
The standard way to evaluate a classifier is:

$$\text{Accuracy} = \frac{\#\text{correctly classified}}{\#\text{all documents}}$$

What could go wrong?

When one class is much more common than the other, predicting it always gives high accuracy.
# Evaluation

<table>
<thead>
<tr>
<th>Predicted/Correct</th>
<th>MinorityClass</th>
<th>MajorityClass</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinorityClass</td>
<td>TruePositive</td>
<td>FalsePositive</td>
</tr>
<tr>
<td>MajorityClass</td>
<td>FalseNegative</td>
<td>TrueNegative</td>
</tr>
</tbody>
</table>

\[
\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}
\]

\[
\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}
\]

\[
\text{F1-Score} = 2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

Use macro scores (averaging across classes) in multiclass classification (with imbalanced data).
### Evaluation

<table>
<thead>
<tr>
<th>Predicted/Correct</th>
<th>MinorityClass</th>
<th>MajorityClass</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinorityClass</td>
<td>TruePositive</td>
<td>FalsePositive</td>
</tr>
<tr>
<td>MajorityClass</td>
<td>FalseNegative</td>
<td>TrueNegative</td>
</tr>
</tbody>
</table>

\[
\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}
\]

\[
\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}
\]

\[
F1\_\text{Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

Use macro scores (averaging across classes) in multiclass classification (with imbalanced data).
Bibliography

- Chapter 5 from Jurafsky & Martin.
- Sections 2.5 and 2.6 from Eisenstein.
- For more background reading on classification, Kevin Murphy’s *introduction* touches upon most important concepts in ML.
So far we saw how to do text classification using a bag of words representation and the logistic regression classifier.

But we have ignored word order. Language is structured! How we can develop sequential language models?