Text Classification with Logistic Regression

COM6513 Natural Language Processing

Nikos Aletras

Computer Science Department

Week 3

Spring 2023

The University Of Sheffield.

In the previous lecture...

Text units

Raw text:

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.

- **word (token/term)**: a sequence of one or more characters excluding whitespaces. Sometimes it includes n-grams.
- **document (text sequence/snippet)**: sentence, paragraph, section, chapter, entire document, search query, social media post, transcribed utterance, pseudo-documents (e.g. all tweets posted by a user), etc.

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.

	bad	good	great	terrible
Doc 1	14	1	0	5
Doc 2	2	5	3	0
Doc 3	0	2	5	0

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

Weighting the Document-Word Matrix: TF.IDF

- 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

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, businness 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, businness 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, businness 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, businness 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, businness 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, businness 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, businness 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**:

Label	**Tweet**
negative	Very sad about Joe.
negative	No Sat off...Have to work 6 days a week.
negative	I'm a sad panda today.
positive	Such a beautiful satisfying day of shopping. Loves it.
positive	Who else is in a happy mood??
positive	Actually quite happy today.

A little test...

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

Label	**Tweet**
negative	Very sad about Joe.
negative	No Sat off...Have to work 6 days a week.
negative	I'm a sad panda today.
positive	Such a beautiful satisfying day of shopping. Loves it.
positive	Who else is in a happy mood??
positive	Actually quite happy today.

- What **features** should be **indicative of** positive/negative **class**?

A little test...

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

Label	**Tweet**
negative	Very sad about Joe.
negative	No Sat off...Have to work 6 days a week.
negative	I'm a sad panda today.
positive	Such a beautiful satisfying day of shopping. Loves it.
positive	Who else is in a happy mood??
positive	Actually quite happy today.

- What **features** should be **indicative of** positive/negative **class**?
- Would they generalize to unseen data?

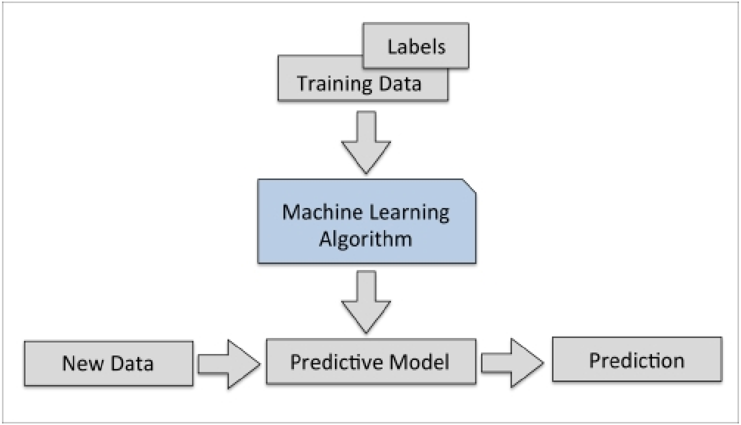
A little test...

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

Label	**Tweet**
negative	Very sad about Joe.
negative	No Sat off...Have to work 6 days a week.
negative	I'm a sad panda today.
positive	Such a beautiful satisfying day of shopping. Loves it.
positive	Who else is in a happy mood??
positive	Actually quite happy today.

- 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



Supervised Learning - Exam Analogy

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 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**).

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.
- 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:

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

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:

$$\hat{y} = f(\mathrm{x}, \mathrm{w})$$

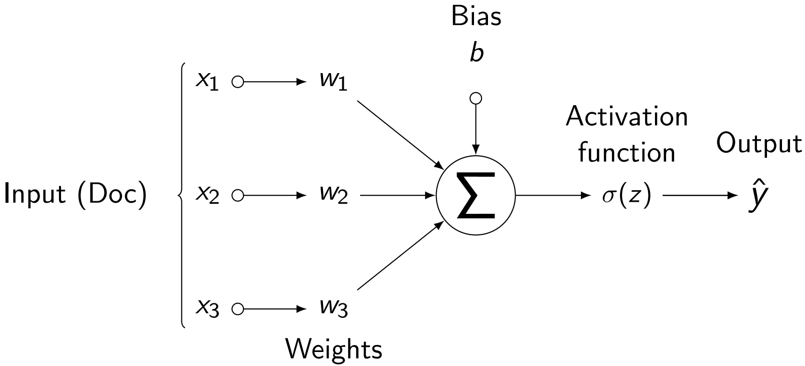
Binary Logistic Regression

- Assume a document vector represented with counts over N words/features, $\mathrm{x} \in \Re^N$.
- 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, $\mathrm{x} \in \Re^N$.
- 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!
- 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



Binary Logistic Regression

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

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

Binary Logistic Regression

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

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

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

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

Binary Logistic Regression

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

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

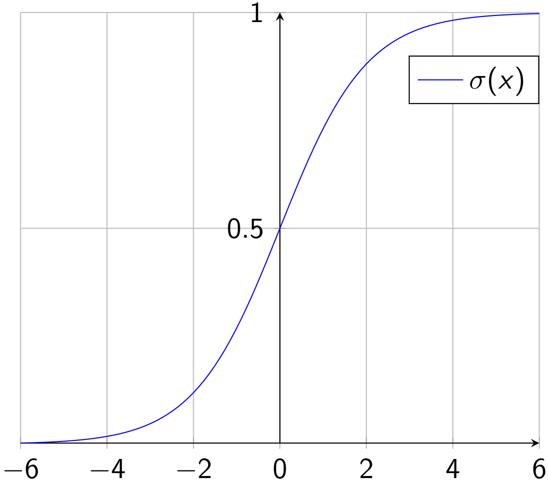
- Compute the probability of the possitive class using the sigmoid function $\sigma(\cdot)$:

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

- Predict the class with the highest probability:

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

Sigmoid function



Squashes a value x between 0 and 1 (or a vector, applied independently to each element).

Multiclass Logistic Regression

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

Multiclass Logistic Regression

- More than one labels: $y \in \mathcal{Y} = \{0, ..., k\}$. 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 $\mathrm{b} \in 1^k$ of ones (often ignored) to the resulting vector z:

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

Multiclass Logistic Regression

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

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

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

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

Multiclass Logistic Regression

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

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

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

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

- Predict the class with the highest probability:

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

Softmax function

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

$$\begin{bmatrix} 1.2 \\ 0.9 \\ 0.4 \end{bmatrix} \rightarrow \textit{softmax} \rightarrow \begin{bmatrix} 0.46 \\ 0.34 \\ 0.20 \end{bmatrix}$$

Wait...

How can we learn the weights?!

Training

- We start by initialising w with 0s.

Training

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

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

- 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:

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

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

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

- 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**)!
- So we can keep iterating over the training data and adjust the weights towards minimising the loss!

Binary Cross-Entropy Loss

- We assume that the weights w should maximise the log-likelihood of the correct class:

$$log(P(y_i = c|\mathrm{x}_i; \mathrm{w}))$$

- 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)

How to adjust the weights?

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):

$$\mathrm{w}^\star = \arg\min_{\mathrm{w}} L(\mathrm{w}; x_i; y_i)$$

A simpler case..

Binary logistic regression has one parameter per feature $\rightarrow$ many parameters!

Let's look at a simpler case:

$$x^\star = \arg\min_{x \in \Re} f(x), f(x) = x^2$$

Gradient-based optimisation

- 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:

$$\nabla_x f(x)$$

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:
  - $sign(\nabla_x f(x))$ tells us if by increasing x, f(x) will increase (+) or decrease (-)
  - $|\nabla_x f(x)|$ tells us how fast the in/de-crease will be

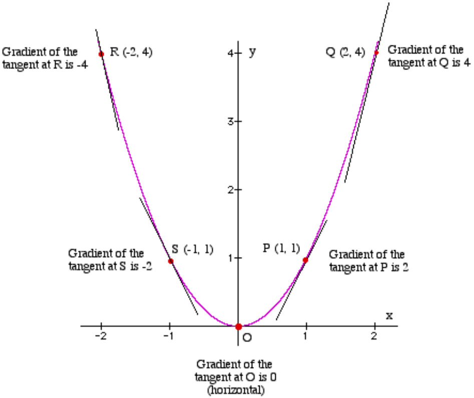
Gradient-based optimisation

- 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:
  - $sign(\nabla_x f(x))$ tells us if by increasing x, f(x) will increase (+) or decrease (-)
  - $|\nabla_x f(x)|$ tells us how fast the in/de-crease 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 $x_k = \arg\min_{x \in \Re} f(x)$

Gradient of the Loss wrt to the weights

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

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

$$\begin{aligned}
\frac{\partial L(\mathrm{w}; \mathrm{x}_i; y_i)}{\partial w_j} &= \frac{\partial \log P(y|\mathrm{x}_i; \mathrm{w})}{\partial w_j} \\
&= \ldots \\
&= (P(y|x_i; \mathrm{w}) - y_i) \cdot \mathrm{x}_i
\end{aligned}$$

You can find more details on deriving the gradients here.

Stochastic Gradient Descent (SGD)

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

initialize w *with zeros*

for each *epoch* e do

randomise order in D_{train}

for each (x_i, y_i) *in* D_{train} do

update $\mathrm{w} = \mathrm{w} - \eta \nabla_{\mathrm{w}} L(\mathrm{w}; x_i; y_i)$

monitor training and validation loss

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

break

return w

η denotes how much you want to update the weights 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 wrt to the parameters for the entire training set
- **Batch Gradient Descent:** computes the gradient of the loss function wrt to the parameters for small parts of the training set
- You can find more details in this blog post.

Regularisation

- Any model with a lot of features is prone to **overfitting** its training data: high training accuracy, low test accuracy.

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:

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

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:

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

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

$$\frac{\partial L_{reg}(\mathrm{w}; D_{train})}{\partial w_k} = \frac{\partial L(\mathrm{w}; D_{train})}{\partial w_k} + 2\alpha w_k$$

- α 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:

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

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

$$\frac{\partial L_{reg}(\mathrm{w}; D_{train})}{\partial w_k} = \frac{\partial L(\mathrm{w}; D_{train})}{\partial w_k} + 2\alpha w_k$$

- α 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 (Weeks 6-10)
- Gaussian Processes

Evaluation

- The standard way to evaluate a classifier is:

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

- What could go wrong?

Evaluation

- The standard way to evaluate a classifier is:

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

- What could go wrong?
- When one class is much more common than the other, predicting it always gives high accuracy.

Evaluation

Predicted/Correct	MinorityClass	MajorityClass
MinorityClass	TruePositive	FalsePositive
MajorityClass	FalseNegative	TrueNegative

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$F1\_Score = 2\frac{Precision \times Recall}{Precision + Recall}$$

Evaluation

Predicted/Correct	**MinorityClass**	**MajorityClass**
MinorityClass	TruePositive	FalsePositive
MajorityClass	FalseNegative	TrueNegative

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$F1\_Score = 2\frac{Precision \times Recall}{Precision + Recall}$$

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.

Coming up next week

- 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?