# Classification with the Perceptron

L101: Machine Learning for Language Processing

Andreas Vlachos

# L101 Practicalities

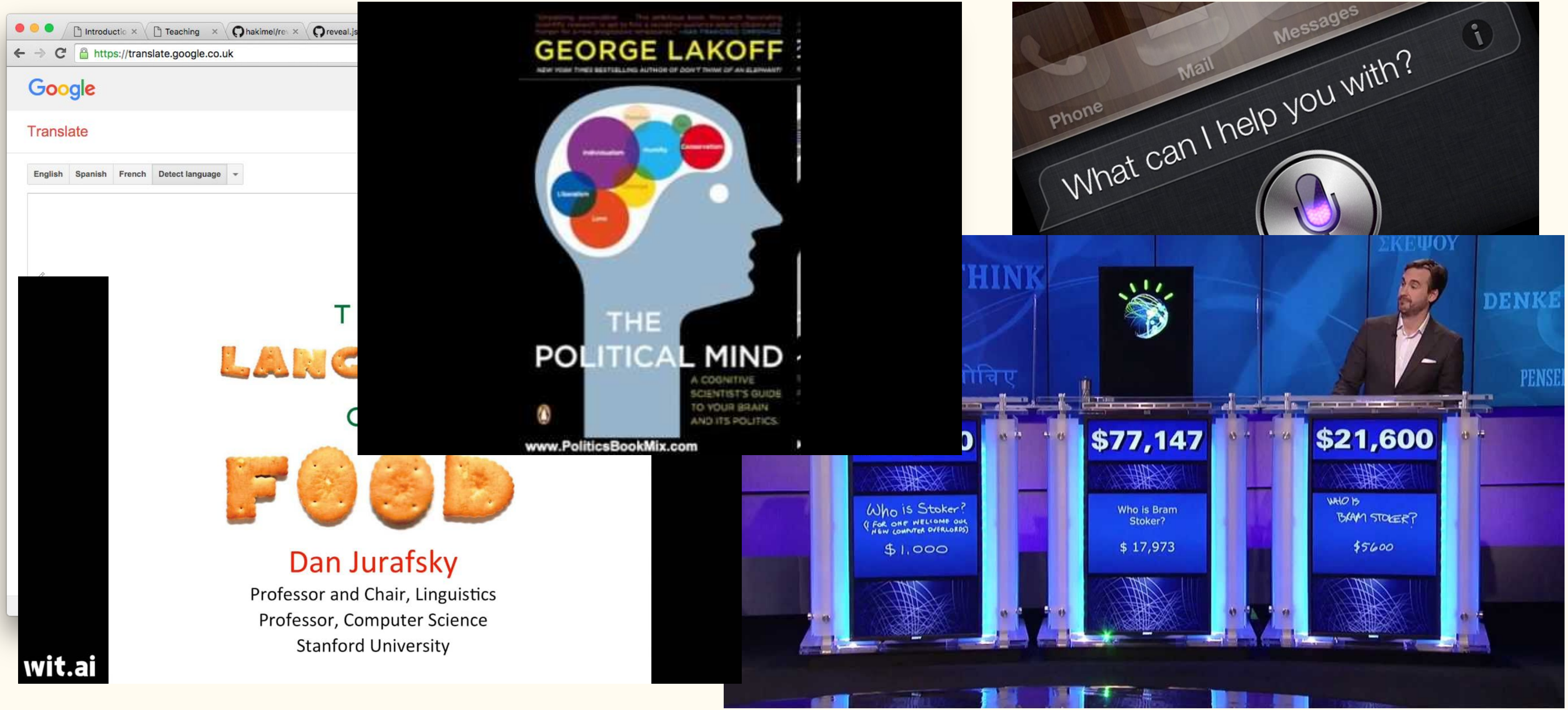
Lecturers:

- Ted Briscoe (ejb@cam.ac.uk)
- Andreas Vlachos (av308@cam.ac.uk)

Materials and information about the assessment:

https://www.cl.cam.ac.uk/teaching/2021/L101/

# Why Natural Language Processing (NLP)?



# Classification

Given a piece of text, assign a label from a predefined set

What could the labels be?

- Positive/negative sentiment
- Topical
- Author name (authorship identification)
- Biased or not
- etc.

# Formulation

Given input instance $x$, parameters $w$, a classifier is a function $f$ that predicts $\hat{y}$:

Binary (1/-1, sometimes 0/1):

$$\hat{y} = sign f(x; w)$$

Multiclass (1 of N):

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} f(x; w)$$

Multilabel (a set of labels, possibly empty):

$$\hat{y} = \arg\max_{y \in \mathcal{P}(\mathcal{Y})} f(x; w)$$

# Features

Given a document, what features would you use to predict its:

- topic?
- sentiment?
- author name?
- factual correctness?

Some ideas:

- bag of words (n-grams)
- meta-data
- sub-word features
- external evidence, common sense...
- Often these are denoted with the feature function: $\phi(x) \in \Re^k$

# Binary linear classifier

$$\hat{y} = sign(w \cdot \phi(x))$$

The “linear” part refers to the function f, a linear map:

$$w \cdot \phi(x)$$

How do we learn the weights $w$?

# Supervised learning

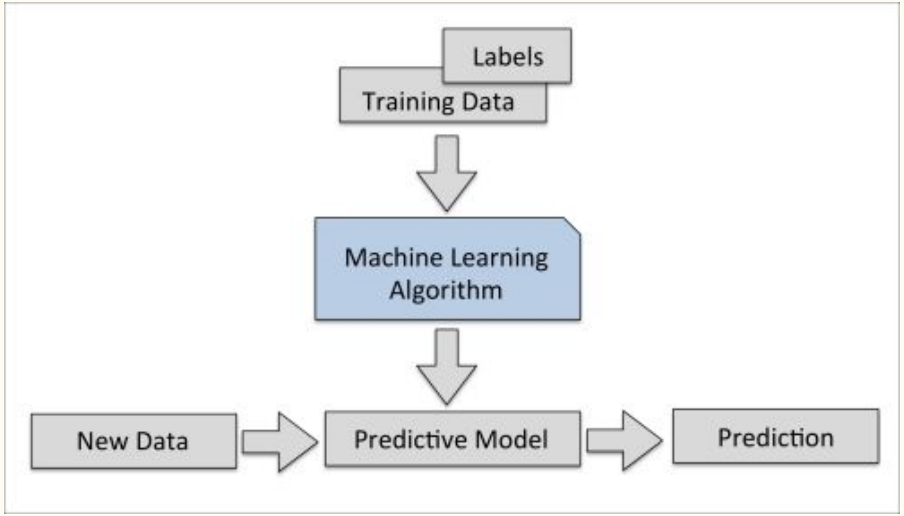


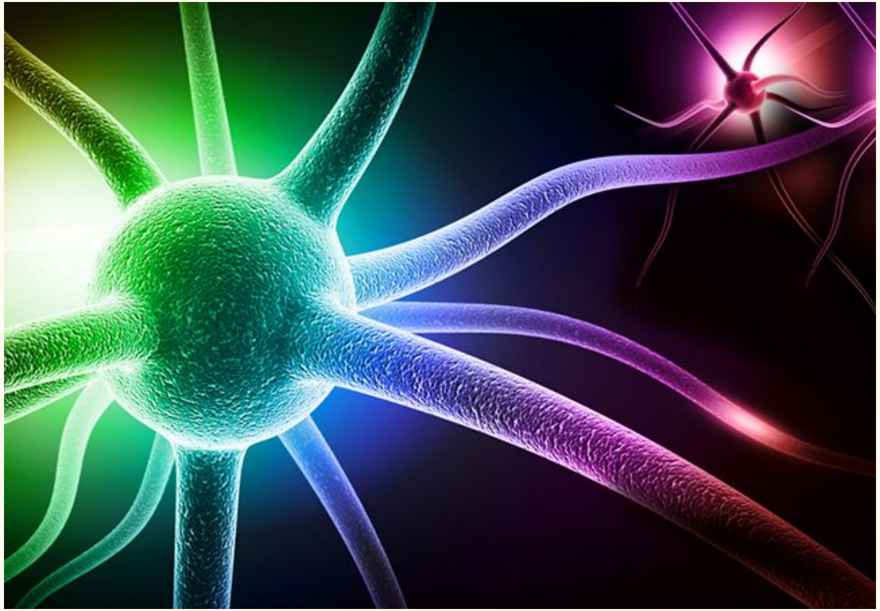

Image Credit:
Sebastian Raschka

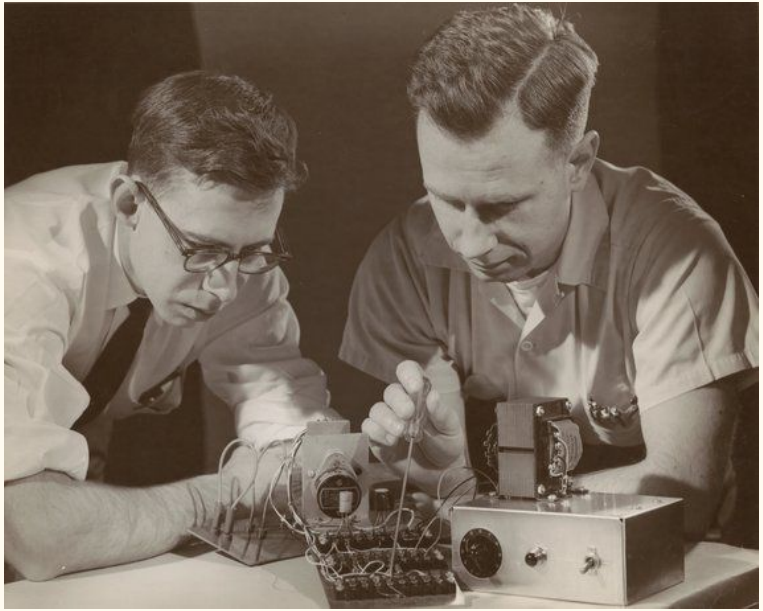
Given labeled training data of the form:

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

Learn weights $w$ that **generalize** well to new instances

# The perceptron





Proposed by Rosenblatt in 1958, still close to state-of-the-art

# The perceptron

1958 New York Times...

## NEW NAVY DEVICE LEARNS BY DOING

### Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

**Without Human Controls**

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

**Learns by Doing**

In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

AI hype is not a new problem

We should always remember when we talk to the public

# The perceptron algorithm

**Input**: training examples $\mathcal{D} = \{(x^1, y^1), \ldots (x^M, y^M)\}$

Initialize weights $w = (0, ..., 0)$

**for** $(x, y) \in \mathcal{D}$ **do**

  Predict label $\hat{y} = sign(w \cdot \phi(x))$

  **if** $\hat{y} \neq y$ **then**

    Update $w = w + y\phi(x)$

  **end if**

**end for**

error-driven, online learning

# Testing our intuitions

Given the following tweets labeled with sentiment and assuming bag of words:

| | |
|---|---|
| negative | Very sad about Iran. |
| negative | No Sat off...Need to work 6 days a week. |
| negative | I’m a sad panda today. |
| positive | such a beautiful satisfying day of bargain shopping. loves it. |
| positive | who else is in a happy mood?? |
| positive | actually quite happy today. |

- What weights do you expect your perceptron to learn?
- Do you think they would generalize well?

# Sparsity and bias

In NLP, no matter how large our training dataset, we will never see (enough of) all the words/features.

- features unseen in training are ignored in testing
- there are ways to ameliorate this issue (e.g. word clusters, word vectors), but it never goes away
- there will be texts containing only unseen words

Bias: a feature that appears in each instance

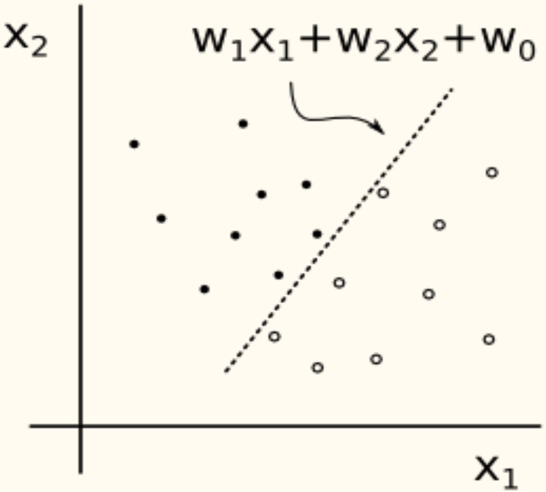
- its value is hardcoded to 1
- often omitted from equations due to its omnipresence
- effectively learns to predict the majority class

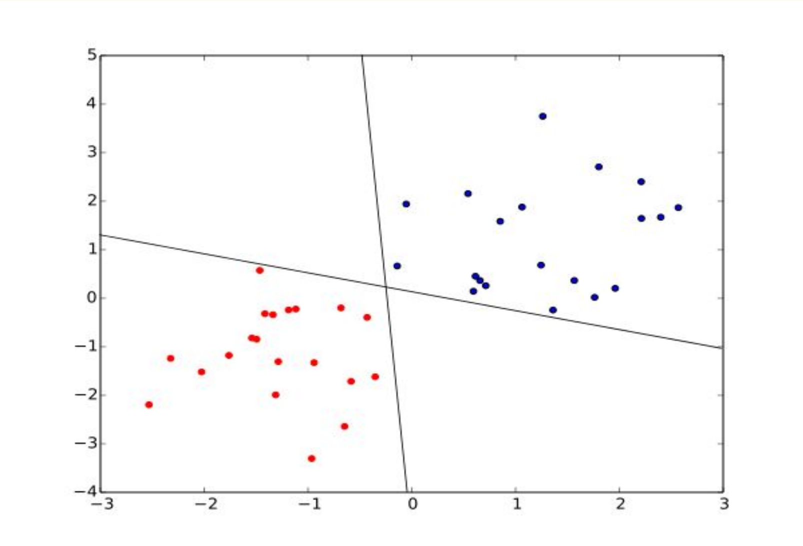
# Improving the perceptron

**Input**: training examples $\mathcal{D} = \{(x^1, y^1), \ldots (x^M, y^M)\}$
Initialize counter $c = 0$, weights $w_c = (0, ..., 0)$
**for** $i = 1 \ldots maxIter$ **do**
  Shuffle $\mathcal{D}$
  **for** $(x, y) \in \mathcal{D}$ **do**
    Predict label $\hat{y} = sign(w_c \cdot \phi(x))$
    **if** $\hat{y} \neq y$ **then**
      Update $w_{c+1} = w_c + y\phi(x)$
    **else**
      $w_{c+1} = w_c$
    **end if**
    $c = c + 1$
  **end for**
**end for**
$w = \frac{1}{c} \sum w_0 + \ldots + w_c$

- Multiple passes
- Shuffling
- Averaging

# On separating hyperplanes




If data not linearly separable, perceptron does not converge (will always update)

Even if linearly separable, no guarantee of finding a good separating hyperplane

Image credits: https://johnpatrickroach.com/2016/09/24/training-a-perceptron-model-in-python/ (left), https://en.wikipedia.org/wiki/Perceptron#/media/File:Perceptron_cant_choose.svg (right)

# Binary to multiclass

Binary: $$\hat{y} = sign(w \cdot \phi(x))$$

Multiclass 1: $$\hat{y} = \arg\max_{y \in \mathcal{Y}} w_y \cdot \phi(x)$$

Input feature map: only describes $x$.

Multiclass 2: $$\hat{y} = \arg\max_{y \in \mathcal{Y}} w \cdot \phi(x, y)$$

Joint feature map: compatibility between $x$ and $y$. More expressive but less intuitive.

# Multiclass perceptron (two versions)

**Input**: training examples $\mathcal{D} = \{(x^1, y^1), \ldots (x^M, y^M)\}$
Initialize weights $w_y = (0, ..., 0)$
Initialize weights $w = (0, ..., 0)$
**for** $(x, y) \in \mathcal{D}$ **do**
Predict label $\hat{y} = \arg\max_{y \in \mathcal{Y}} w_y \cdot \phi(x) \; / \; w \cdot \phi(x, y)$
**if** $\hat{y} \neq y$ **then**
Update $w_y = w_y + \phi(x)$
Update $w_{\hat{y}} = w_{\hat{y}} - \phi(x)$
Update $w = w + \phi(x, y) - \phi(x, \hat{y})$
**end if**
**end for**

# Evaluation

$$accuracy = \frac{correctPredictions}{allPredictions}$$

What can go wrong?

Imbalanced datasets: predicting one class gives 90% accuracy

Very common: most topics are not relevant to most documents, etc.

# Evaluation

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

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

Often combined into their harmonic mean, aka F1-score

# Evaluation (cont.)

So far we assumed a fixed decision threshold but often it makes sense to adjust it

Don't pick a single threshold, check them all!

Can summarize in the area under the precision-recall curve (AUC)

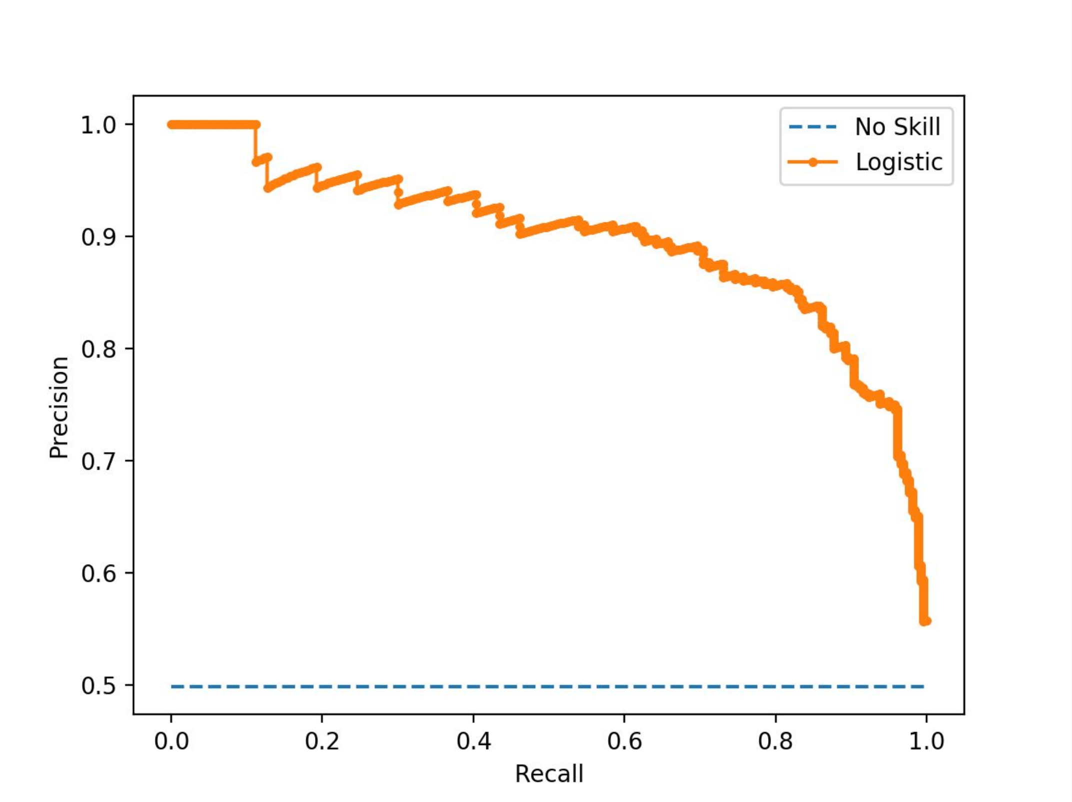



Image from https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-imbalanced-classification/

# Multilabel classification

$$\hat{y} = \arg\max_{y \in \mathcal{P}(\mathcal{Y})} f(x; w)$$

Can be done in two ways:

- Binary relevance: build a binary classifier for each label in *Y*
  - Ignores label dependencies
- Multiclass: build a multiclass classifier for all members of the powerset *P(Y)*
  - Can be computationally expensive, training data can be sparse

Both are instances of **reduction:** transform complex problems to simpler ones

More advanced methods exist, but always try these two first

# What sentiment classifier would you build?


Type: Binary, multiclass, something else?

What features? What weights do you expect for them?

# Bibliography

Hal Daumé III's chapter on the perceptron from his book on machine learning

For more background reading on classification, Kevin Murphy's introduction touches upon most important concepts in ML