Recap: classification

Statistical Methods in NLP 2
ISCL-BA-08

Çağrı Çöltekin
ccoltekin@sfs.uni-tuebingen.de

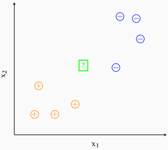
University of Tübingen
Seminar für Sprachwissenschaft

Summer Semester 2025

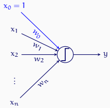
version: 69c3db4 @2025-05-02

Classification

- Given a training set with (categorical) labels/outcome
- Train a model to predict future data points from the same distribution



The perceptron



$$y = f\left(\sum_i^n w_i x_i\right)$$

where

$$f(x) = \begin{cases} +1 & \text{if} \quad \sum_i^n w_i x_i > 0 \\ -1 & \text{otherwise} \end{cases}$$

Note the similarity with linear regression: the separation between the (predicted) classes is a linear line/surface.

Learning with perceptron

- We do not update the parameters if classification is correct
- For misclassified examples, we try to minimize

$$E(\boldsymbol{w}) = -\sum_i \boldsymbol{w}\boldsymbol{x}_i y_i$$

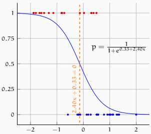
where i ranges over all misclassified examples

- Perceptron algorithm updates the weights for misclassified examples

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla E(\boldsymbol{w})$$
$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \boldsymbol{x}_i y_i$$

- Perceptron algorithm converges if the classes are *linearly separable*

Logistic regression

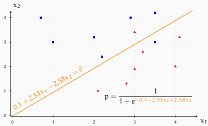


- Logistic regression is probabilistic, we estimate $p = P(y = 1|x)$
- Note that typical regression would penalize correctly classified examples
- Instead we fit a regression model for *logit*$(p) = \boldsymbol{w}^T\boldsymbol{x} + b$
- The probability estimate is the inverse of *logit*, the *logistic* (sigmoid) function

$$p = \frac{1}{1 + e^{-\boldsymbol{w}^T\boldsymbol{x}}}$$

Another example with two predictors

Probability assignments are non-linear, but the discriminant function is linear



Fitting a logistic regression model

$$p = \frac{1}{1 + e^{-\boldsymbol{w}\boldsymbol{x}}}$$

The likelihood of the training set is,

$$\mathcal{L}(\boldsymbol{w}) = \prod_i p^{y_i}(1-p)^{1-y_i}$$

In practice, we maximize log likelihood, or minimize '− log likelihood':

$$-\log \mathcal{L}(\boldsymbol{w}) = -\sum_i y_i \log p + (1 - y_i)\log(1 - p)$$

- The derivative/gradient is easy (a good exercise)
- There is no analytic solution for $\nabla - \log \mathcal{L}(\boldsymbol{w}) = 0$
- But the loss function is convex: we can find the global minimum with gradient descent (most of the time)

Naive Bayes

- *Naive Bayes* is another probabilistic classification method
- In any classification task, our aim is to find

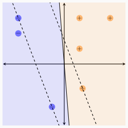
$$\hat{y} = \underset{y}{\arg\max}\, P(y \mid \boldsymbol{x})$$

- Sometimes (with some simplifying assumptions) it is easier to predict $P(\boldsymbol{x} \mid y)$
- Then we use Bayes' formula to invert the conditional proabality

$$\hat{y} = \underset{y}{\arg\max} \frac{P(\boldsymbol{x} \mid y)P(y)}{P(\boldsymbol{x})} = \underset{y}{\arg\max}\, P(\boldsymbol{x} \mid y)P(y)$$

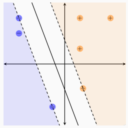
- $P(\boldsymbol{x} \mid y)$ and $P(y)$ are generally estimated using MLE (with *smoothing*)

Maximum-margin methods (e.g., SVMs)



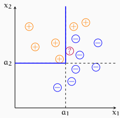
- In perceptron, we stop whenever we found a linear discriminator
- Maximum-margin classifiers seek a discriminator that maximizes the margin
- SVMs have other interesting properties, and they have been one of the best 'out-of-the-box' classifiers for many problems

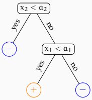
Maximum-margin methods (e.g., SVMs)



- In perceptron, we stop whenever we found a linear discriminator
- Maximum-margin classifiers seek a discriminator that maximizes the margin
- SVMs have other interesting properties, and they have been one of the best 'out-of-the-box' classifiers for many problems

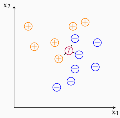
Decision trees





- Note that the decision boundary is non-linear

Instance/memory based methods



- No training: just memorize the instances
- During test time, decide based on the k nearest neighbors
- Like decision trees, kNN is non-linear
- It can also be used for regression

More than two classes

- Some algorithms can naturally be extended to handle multiple class labels
- Any binary classifier can be turned into a k-way classifier by
  - OvR one-vs-rest or one-vs-all
    - train k classifiers: each learns to discriminate one of the classes from the others
    - at prediction time the classifier with the highest confidence wins
    - needs a confidence score from the base classifiers
  - OvO one-vs-one
    - train $\frac{k(k-1)}{2}$ classifiers: each learns to discriminate a pair of classes
    - decision is made by (weighted) majority vote
    - works without need for confidence scores, but needs more classifiers

Measuring success in classification

Accuracy, precision, recall, F-score

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$F_1\text{-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

		predicted	
		positive	negative
true value	pos.	TP	FN
	neg.	FP	TN

Multi-class evaluation

- For multi-class problems, it is common to report average precision/recall/f-score
- For C classes, averaging can be done two ways:

$$\text{precision}_M = \frac{\sum_i^C \frac{TP_i}{TP_i + FP_i}}{C} \qquad \text{recall}_M = \frac{\sum_i^C \frac{TP_i}{TP_i + FN_i}}{C}$$

$$\text{precision}_\mu = \frac{\sum_i^C TP_i}{\sum_i^C TP_i + FP_i} \qquad \text{recall}_\mu = \frac{\sum_i^C TP_i}{\sum_i^C TP_i + FN_i}$$

(M = macro, μ = micro)

- The averaging can also be useful for binary classification, if there is no natural positive class

Confusion matrix

- A confusion matrix is often useful for multi-class classification tasks

		predicted		
		negative	neutral	positive
true value	negative	10.00	2.00	0.00
	neutral	3.00	12.00	7.00
	positive	4.00	8.00	7.00

- Are the classes balanced?
- What is the accuracy?
- What is per-class, and averaged precision/recall?

Overfitting & Underfitting

We want our models to *generalize*, perform well on unseen data.

- *Overfitting* occurs when the model learns the idiosyncrasies of the training data
- *Underfitting* occurs when the model is not flexible enough for solving the problem at hand

We want simpler models, but not too simple for the task at hand.

Bias and variance

Bias of an estimate is the difference between the value being estimated, and the expected value of the estimate

$$B(\hat{\boldsymbol{w}}) = E[\hat{\boldsymbol{w}}] - \boldsymbol{w}$$

- An *unbiased* estimator has 0.00 bias

Variance of an estimate is, simply its variance, the value of the squared deviations from the mean estimate

$$\text{var}(\hat{\boldsymbol{w}}) = E\left[(\hat{\boldsymbol{w}} - E[\hat{\boldsymbol{w}}])^2\right]$$

$\boldsymbol{w}$ is the parameter (vector) that defines the model

Bias–variance relationship is a trade-off:
models with low bias result in high variance.

ML evaluation in general

The first principle is that you must not fool yourself and you are the easiest person to fool. – *Richard P. Feynman*

- We want models with low bias and low variance, but this is a trade-off
- Estimators/models with low/no bias *overfit*
- Estimators/models with high bias (may) *underfit*
- Evaluating ML system requires special care:
  - Tuning your system on a development set
  - Cross-validation allows efficient use of labeled data during tuning
  - A test set is often used when comparing results obtained by different models
- In most cases, the scores are not meaningful by themselves, we need to compare with *baselines*

Final remarks

- Most NLP problems we try to solve are classification problems
- We reviewed some of the 'traditional' classification methods
- Understanding them will help understanding more 'modern' methods
- The models we review can serve as baselines for more complex models, and sometimes their performance may surprise you

Next:

Mon Lab: numpy tutorial

Wed Representing linguistic data

Some sources of information

- Any ML textbook covers most of the methods reviewed (and more), here are a few: James et al. (2024), Bishop (2006), and MacKay (2003)

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Springer. ISBN: 978-0387-31073-2.

James, Gareth, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor (2024). *An introduction to statistical learning*. Springer. ISBN: 9783031391897. URL: https://www.statlearning.com/.

MacKay, David J. C. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press. ISBN: 978-05-2164-298-9. URL: http://www.inference.phy.cam.ac.uk/itprnn/book.html.