COMPUTATIONAL OPINION ANALYSIS, COST ACTION CA21129 OPINION

14-06-2024, 12.30-13.30, Salamanca

»Supervised Text Classification«

Damian Trilling

d.c.trilling@uva.nl, @damian0604

14-06-2024, 12.30-13.30

Vrije Universiteit Amsterdam

Today

- Predicting things
  - You have done it before!
  - From regression to classification
  - Classifiers
  - Vectorizers
- Summing up
  - Revisiting the difference between the dictionary approach and the SML

Predicting things

Short recap from this morning

Predicting things

You have done it before!

You have done it before!

Regression

1. Based on your data, you estimate some regression equation
   $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$
3. Example: You estimated a regression equation where y is newspaper reading in days/week
   $y = \alpha + \beta_1 \text{Age} + \beta_2 \text{Wage}$
4. You could now calculate $\hat{y}$ for a man of 20 years and another of 40 years — even if no such person exists in your data set
   $\hat{y}_{\text{man,20}} = \alpha + \beta_1 \cdot 20 + \beta_2 \cdot 0$
   $\hat{y}_{\text{man,40}} = \alpha + \beta_1 \cdot 40 + \beta_2 \cdot 0$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation
   $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
   $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
   $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$
   $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation
   $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
   $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
   $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$
   $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
   $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
   $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$
   $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation
   $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
   $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
   $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$
   $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation
   $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
   $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
   $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$
   $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

This is

Supervised Machine Learning!

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")
- Typically, IVs are word frequencies (often weighted, e.g. tf×idf) (⇒BOW-representation)

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")
- Typically, IVs are word frequencies (often weighted, e.g. tf×idf) (⇒BOW-representation)

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")
- Typically, IVs are word frequencies (often weighted, e.g. tf×idf) (⇒BOW-representation)

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")
- Typically, IVs are word frequencies (often weighted, e.g. tf×idf) (⇒BOW-representation)

Predicting things

From regression to classification

In the machine learning world, predicting some continous value is referred to as a regression task. If we want to predict a binary or categorical variable, we call it a classification task.

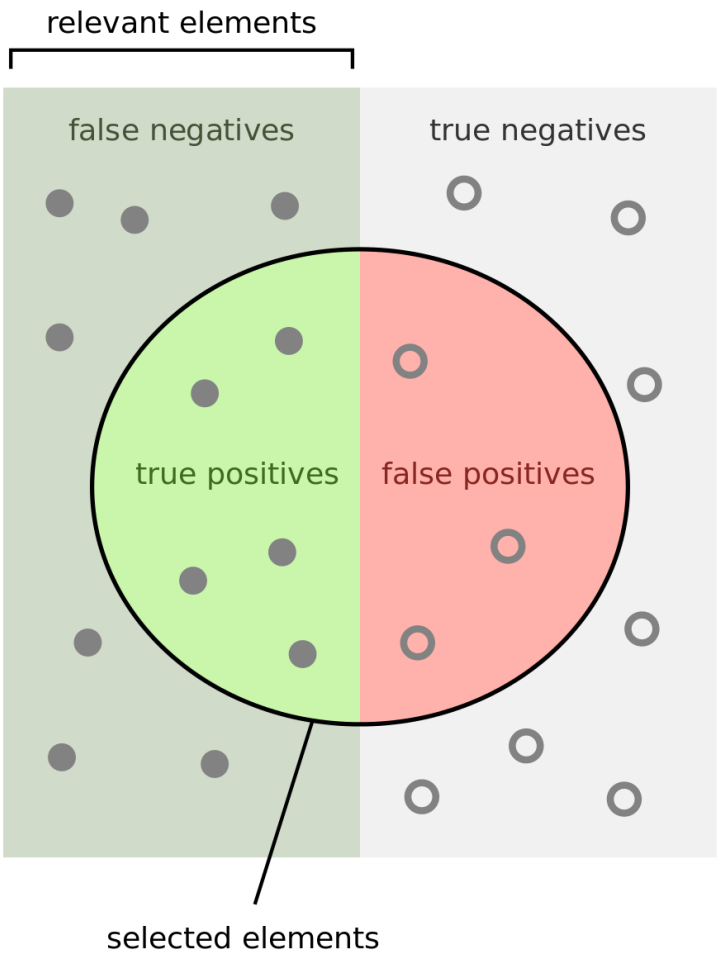
(quite confusingly, even if we use a logistic regression for the latter)

Classification tasks

For many computational approaches, we are actually not that interested in predicting a continous value. Typical questions include:

- Is this article about topix A, B, C, D, or E?
- Is this review positive or negative?
- Does this text contain frame F?
- I this satire?
- Is this misinformation?
- Given past behavior, can I predict the next click?

http://commons.wikimedia.org/wiki/File:Precisionrecall.svg



How many selected items are relevant?

$$\text{Precision} = \frac{\bullet}{\bullet\bullet}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\bullet}{\bullet\bullet}$$

Some measures

- Accuracy
- Recall
- Precision
- $F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$
- AUC (Area under curve) $[0, 1]$, $0.5 =$ random guessing

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid p-hacking-like scenario's (which we call "overfitting"), there are techniques available (cross-validation, later in this course)

(to make it easier, imagine a binary classfication ("positive"/"negative"), but it doesn't really matter whether there are two or more labels)

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid p-hacking-like scenario's (which we call "overfitting"), there are techniques available (cross-validation, later in this course)

(to make it easier, imagine a binary classfication ("positive"/"negative"), but it doesn't really matter whether there are two or more labels)

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid p-hacking-like scenario's (which we call "overfitting"), there are techniques available (cross-validation, later in this course)

(to make it easier, imagine a binary classfication ("positive"/"negative"), but it doesn't really matter whether there are two or more labels)

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid *p*-hacking-like scenario's (which we call "overfitting"), there are techniques available (cross-validation, later in this course)

(to make it easier, imagine a binary classfication ("positive"/"negative"), but it doesn't really matter whether there are two or more labels)

Naïve Bayes

Bayes' theorem

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$

A = Text is about sports

B = Text contains 'very', 'close', 'game'. Furthermore, we simply multiply the propabilities for the features:

$$P(B) = P(very\ close\ game) = P(very) \times P(close) \times P(game)$$

We can fill in all values by counting how many articles are about sports, and how often the words occur in these texts. (Fully elaborated example on https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/)

Naïve Bayes

Bayes' theorem

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$

A = Text is about sports

B = Text contains 'very', 'close', 'game'. Furthermore, we simply multiply the propabilities for the features:

$$P(B) = P(very\ close\ game) = P(very) \times P(close) \times P(game)$$

We can fill in all values by counting how many articles are about sports, and how often the words occur in these texts. (Fully elaborated example on https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/)

Naïve Bayes

Bayes' theorem

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$

A = Text is about sports

B = Text contains 'very', 'close', 'game'. Furthermore, we simply multiply the propabilities for the features:

$$P(B) = P(very\ close\ game) = P(very) \times P(close) \times P(game)$$

We can fill in all values by counting how many articles are about sports, and how often the words occur in these texts. (Fully elaborated example on https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/)

Naïve Bayes

- It's "naïve" because the features are treated as completely independent ($\neq$ "controlling" in regression analysis)
- It's fast and easy
- It's a good *baseline* for binary classification problems

Naïve Bayes

- It's "naïve" because the features are treated as completely independent ($\neq$ "controlling" in regression analysis)
- It's fast and easy
- It's a good *baseline* for binary classification problems

Naïve Bayes

- It's "naïve" because the features are treated as completely independent ($\neq$ "controlling" in regression analysis)
- It's fast and easy
- It's a good *baseline* for binary classification problems

Naïve Bayes

$$P(\text{label} \mid \text{features}) =$$

$$\frac{P(x_1 \mid \textit{label}) \cdot P(x_2 \mid \text{label}) \cdot \text{P}(\text{x}_3 \mid \text{label}) \cdot \text{P}(\text{label})}{P(x_1) \cdot P(x_2) \cdot P(x_3)}$$

.

- Formulas always look intimidating, but we only need to fill in how many documents containing feature x_n have the label, how often the label occurs, and how often each feature occurs
- Also for computers, this is *really easy and fast*
- Weird assumption: features are independent
- Often used as a baseline

Logistic Regression

Probability of a binary outcome in a regression model

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n)}}$$

Just like in OLS regression, we have an intercept and regression coefficients. We use a threshold (default: 0.5) and above, we assign the positive label ('good movie'), below, the negative label ('bad movie').

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- . . . or to change the threshold to change our precision/recall-tradeoff

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- . . . or to change the threshold to change our precision/recall-tradeoff

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- . . . or to change the threshold to change our precision/recall-tradeoff

Logistic Regression

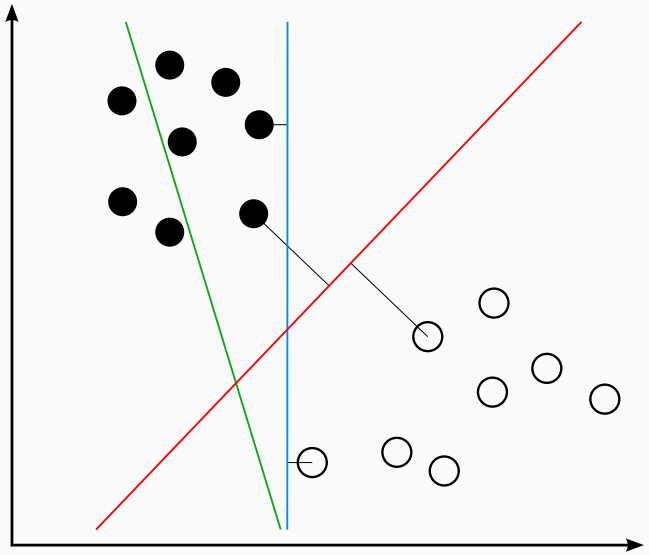
- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- . . . or to change the threshold to change our precision/recall-tradeoff

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- . . . or to change the threshold to change our precision/recall-tradeoff

Support Vector Machines

- Idea: Find a hyperplane that best seperates your cases
- Can be linear, but does not have to be (depends on the so-called kernel you choose)
- Very popular



https://upload.wikimedia.org/wikipedia/commons/b/b5/Svm_separating_hyperplanes_%28SVG%29.svg

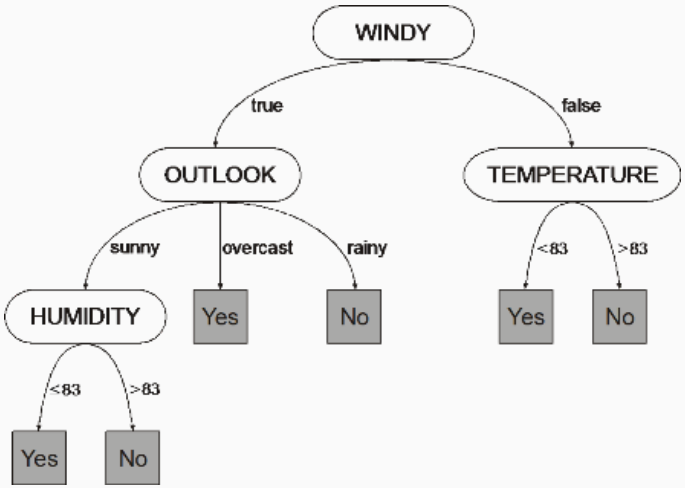
(Further reading: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/)

SVM vs logistic regression

- for *linearly separable* classes not much difference
- with the right hyperparameters, SVM is less sensitive to outliers
- biggest advantage: with the *kernel trick*, data can be transformed that they *become* linearily separable

Decision Trees and Random Forests

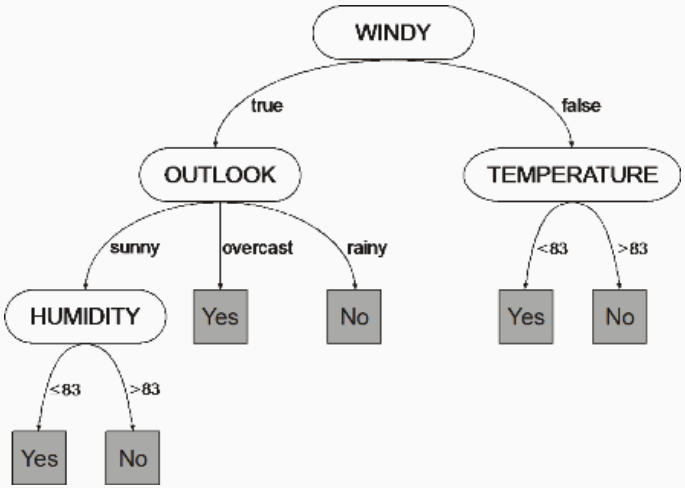
- Model problem as a series of decisions (e.g., if cloudy then . . . if temperature > 30 degrees then . . .)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) (“white box”)



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

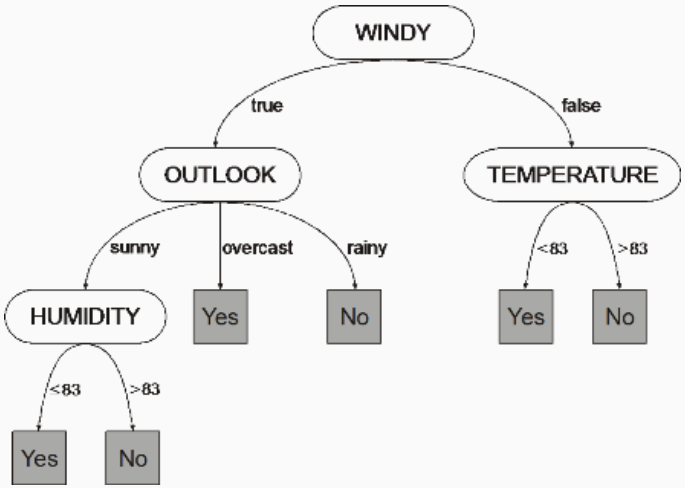
- Model problem as a series of decisions (e.g., if cloudy then . . . if temperature > 30 degrees then . . .)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) (“white box”)



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

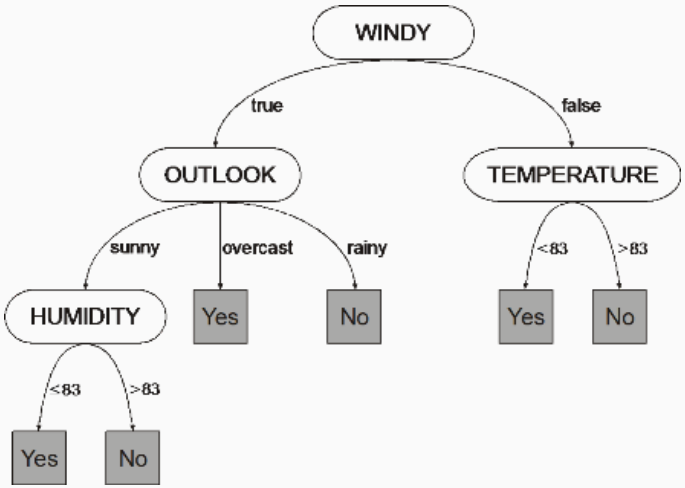
- Model problem as a series of decisions (e.g., if cloudy then . . . if temperature $>$ 30 degrees then . . .)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) (“white box”)



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

- Model problem as a series of decisions (e.g., if cloudy then . . . if temperature $>$ 30 degrees then . . .)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) (“white box”)



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

Disadvantages of decision trees

- comparatively inaccurate
- once you are in the wrong branch, you cannot go 'back up'
- prone to overfitting (e.g., outlier in training data may lead to completely different outcome)

Therfore, nowadays people use *random forests*: Random forests *combine* the predictions of *multiple* trees $\Rightarrow$ might be a good choice for your non-linear classification problem

Decision Trees and Random Forests

Disadvantages of decision trees

- comparatively inaccurate
- once you are in the wrong branch, you cannot go 'back up'
- prone to overfitting (e.g., outlier in training data may lead to completely different outcome)

Therfore, nowadays people use *random forests*: Random forests *combine* the predictions of *multiple* trees $\Rightarrow$ might be a good choice for your non-linear classification problem

After we train a first (baseline) model, we can often do better

We can use different vectorizers and different classifiers.

Predicting things

Classifiers

Different classifiers

Typical options in a nutshell:

- Naïve Bayes
- Logistic Regression
- Support Vector Machine (SVM/SVC)
- Random forests

Predicting things

Vectorizers

Different vectorizers

1. CountVectorizer (=simple word counts)
2. TfidfVectorizer (word counts ("term frequency") weighted by number of documents in which the word occurs at all ("inverse document frequency"))

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t$$

There are different ways to weigh the idf score. A common one is taking the logarithm:

$$idf_t = \log \frac{N}{n_t}$$

where N is the total number of documents and n_t is the number of documents containing term t

Different vectorizers

1. CountVectorizer (=simple word counts)
2. TfidfVectorizer (word counts ("term frequency") weighted by number of documents in which the word occurs at all ("inverse document frequency"))

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t$$

There are different ways to weigh the idf score. A common one is taking the logarithm:

$$idf_t = \log \frac{N}{n_t}$$

where N is the total number of documents and n_t is the number of documents containing term t

Different vectorizers

1. CountVectorizer (=simple word counts)
2. TfidfVectorizer (word counts ("term frequency") weighted by number of documents in which the word occurs at all ("inverse document frequency"))

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t$$

There are different ways to weigh the idf score. A common one is taking the logarithm:

$$idf_t = \log \frac{N}{n_t}$$

where N is the total number of documents and n_t is the number of documents containing term t

Different vectorizers

1. CountVectorizer (=simple word counts)
2. TfidfVectorizer (word counts ("term frequency") weighted by number of documents in which the word occurs at all ("inverse document frequency"))

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t$$

There are different ways to weigh the idf score. A common one is taking the logarithm:

$$idf_t = \log \frac{N}{n_t}$$

where N is the total number of documents and n_t is the number of documents containing term t

Different vectorizer options

- Preprocessing (e.g., stopword removal)
- Remove words below a specific threshold ("occurring in less than $n = 5$ documents") $\Rightarrow$ spelling mistakes etc.
- Remove words above a specific threshold ("occuring in more than 50% of all documents) $\Rightarrow$ de-facto stopwords
- Not only to improve prediction, but also performance (can reduce number of features by a huge amount)

Which one would you (not) use for which purpose?

NB with Count

	precision	recall
positive reviews:	0.87	0.77
negative reviews:	0.79	0.88

NB with TfIdf

	precision	recall
positive reviews:	0.87	0.78
negative reviews:	0.80	0.88

LogReg with Count

	precision	recall
positive reviews:	0.87	0.85
negative reviews:	0.85	0.87

LogReg with TfIdf

	precision	recall
positive reviews:	0.89	0.88
negative reviews:	0.88	0.89

Summing up

Summing up

Revisiting the difference between the dictionary approach and the SML

What *is* our fitted classifier again?

Essentially, just a formula

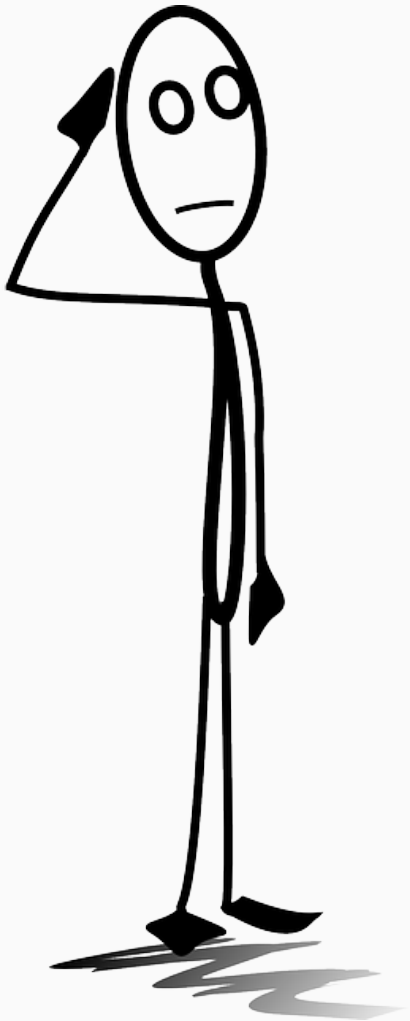
$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n)}}$$

where β_0 is an intercept[1], β_1 a coefficient for the frequency (or tf·idf score) of some word, β_2 a coefficient some other word.

If our fitted *vectorizer* contains 5,000 words, we thus have 5,001 coefficients.

(for logistic regression in this case, but same argument applies to other classifiers as well)

[1] Machine Learning people sometimes call the intercept "bias" (yes, I know, that's confusing)



But isn't that then essentially very much like a dictionary, except that the words have different weights?

In some sense, yes.

- But we don't pretend that we can construct the dictionary *a priori*.
- It's specifically tailored to our use-case.
- The weights are *really* essential here.

We *could* print all coefficients-word pairs, but probably it's enough to just look at those with the largest absolute value:

In some sense, yes.

- But we don't pretend that we can construct the dictionary *a priori*.
- It's specifically tailored to our use-case.
- The weights are *really* essential here.

We *could* print all coefficients-word pairs, but probably it's enough to just look at those with the largest absolute value:

Do we care about the features?

In [98]:
```
import eli5
eli5.show_weights(pipe, top=10)
```

Out[98]:

y=1 top features

Weight[?]	Feature
+9.043	great
+8.487	excellent
+6.908	perfect
... 37662 more positive ...	
... 37178 more negative ...	
-6.507	worse
-7.347	poor
-8.341	boring
-8.944	waste
-8.976	bad
-9.152	awful
-12.749	worst

In [111]:
```
eli5.show_prediction(clf, test[0][0],vec=vec)
```

Out[111]:

y=1 (probability **0.844**, score **1.689**) top features

Contribution[?]	Feature
+1.920	Highlighted in text (sum)
-0.232	<BIAS>

it is a rare and fine spectacle, an allegory of death and transfiguration that is neither preachy nor mawkish. a work of mature and courageous insight, northfork avoids arthouse distinction by refusing to belong to a kind. unlike the most memorable and accomplished film to impose an obvious comparison, wim wenders' 1987 wings of desire (der himmel über berlin), it sustains an ambivalence in a narrative spectrum spanning from the mundane to the supernatural. this story of earthly and celestial eminent domains in the american west withholds the fairytale literalness that marked its german predecessor in the ad hoc genre of angels shedding their wings with obsequious sentimentalism. its celestial transcendence, be it inspired by doleful faith or impelled by a fever dream, never parts ways with crud and rot. this firm grounding redounds to great credit for writers and directors mark and michael polish.

ELI5

- Inspecting *all* coefficients of a ML model usually doesn't make much sense
- But that does not mean that we cannot understand how the model makes its predictions
- We can look at the most important coefficients
- We can look which words in a given text contributed most to its classfication

But have we solved all problems of dictionaries?

No.

For instance, the negation and/or intensifier problem.

Possible approaches

- *n*-grams as features
- preprocessing (?)
- deep learning
- ...

⇒ But ultimately, it's just an empirical question how big the problem is!

But have we solved all problems of dictionaries?

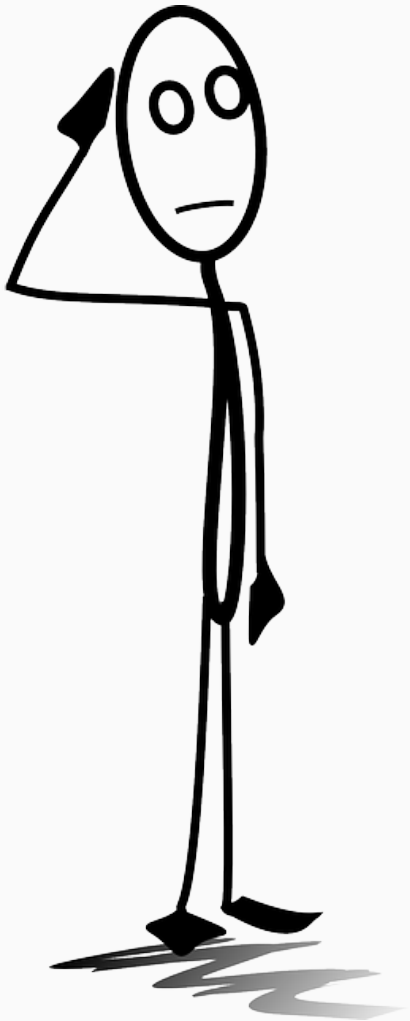
No.

For instance, the negation and/or intensifier problem.

Possible approaches

- *n*-grams as features
- preprocessing (?)
- deep learning
- ...

⇒ **But ultimately, it's just an empirical question how big the problem is!**



Ready for exercisting?