Machine Learning

The Maximum Entropy classifier

Maximum Entropy Model

- We are now deriving one of most widely used classifier (still today!)
- The intuition is based on the *„Principle of Maximum Entropy“*

Principle of Maximum Entropy (Jaynes, 1957)
If incomplete information about a probability distribution is available, the only unbiased assumption that can be made is a distribution which is as uniform as possible, given the available information.

Principle of Maximum Entropy

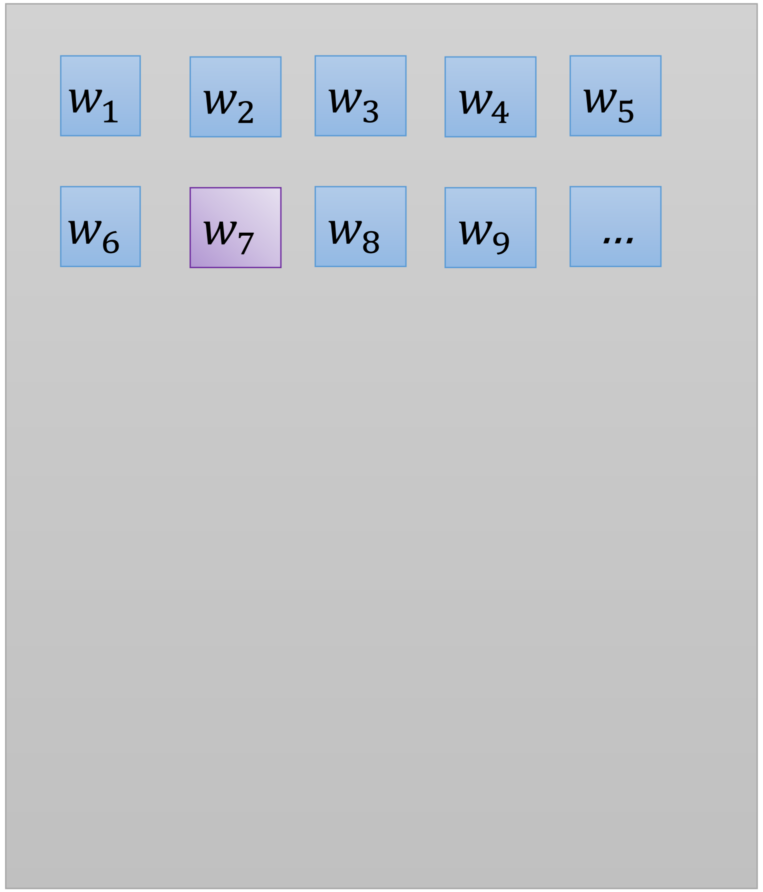
- What does this even mean?
- Assume, that we are given text and want to predict POS-tags for every word
- And we know (from arbitrary sources, e.g. Dbpedia, labelled data), that „Markus“ is 4 times as likely to be a noun than a verb
- Since we have no idea about the rest, we would assume they are uniformly distributed(all are equally as probable)

Principle of Maximum Entropy

- Introduction of features
- We can formulate the piece of knowledge, that „the word = Markus is 4 times more likely to be a noun than a verb“ using **feature functions**

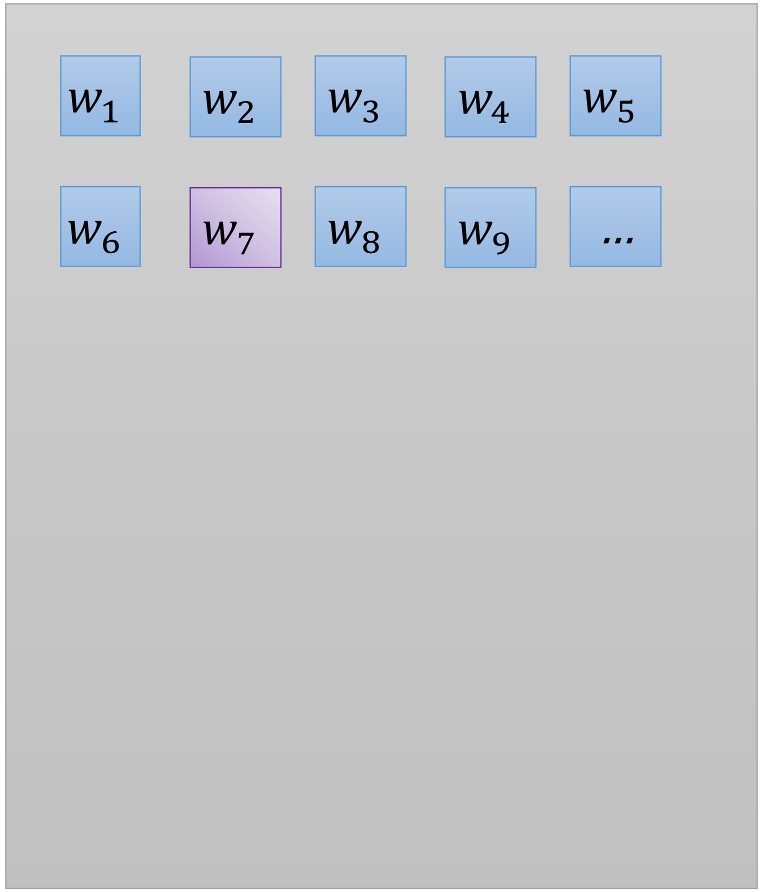
- Let us introduce the feature:
  - Currentword = „Markus“

We will address this soon!



Principle of Maximum Entropy

- In the same manner, we can introduce arbitrary features:
    - Previous-Word = „Markus“
    - Next-Word = „Markus“
    - Previous-Word = „and“
    - Next-Word = „drives“
    - …
- We could in fact use whatever comes to our mind



Features in Maximum Entropy

- The introduction of features was a genius idea! We can now describe all of our words using a set of features
- Let us do this while making use of three simple features:

Feature-Name	Feature Value
Current Word	
Previous Word	
Next Word	

w_1 w_2 w_3 w_4 w_5

w_6 w_7 w_8 w_9 …

Features in Maximum Entropy

- We can now describe our words as follows

- For w_1

Feature-Name	Feature Value
Current Word	w_1
Previous Word	start
Next Word	w_2

Features in Maximum Entropy

- We can now describe our words as follows

- For w_2

Feature-Name	Feature Value
Current Word	w_2
Previous Word	w_1
Next Word	w_3

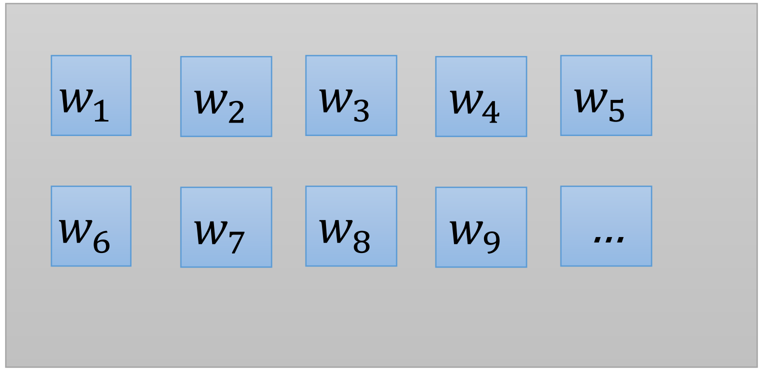
… and so on for the other words

Features in Maximum Entropy

- After this point it is no longer important, that we started with a text classification process, since all we have left are tables of features for every instance (a word)

Feature-Name [ID]	Feature Value
Current Word	w_1
Previous Word	start
Next Word	w_2

Feature-Name [ID]	Feature Value
Current Word	w_2
Previous Word	w_1
Next Word	w_3



… more tables

Features in Maximum Entropy

- We did not deal with one aspect:
- „the word = Markus is 4 times more likely to be a noun than a verb“
- Let us aggregate all our tables into one huge table

Feature-Name [ID]	Feature Value
Current Word	w_1
Previous Word	start
Next Word	w_2

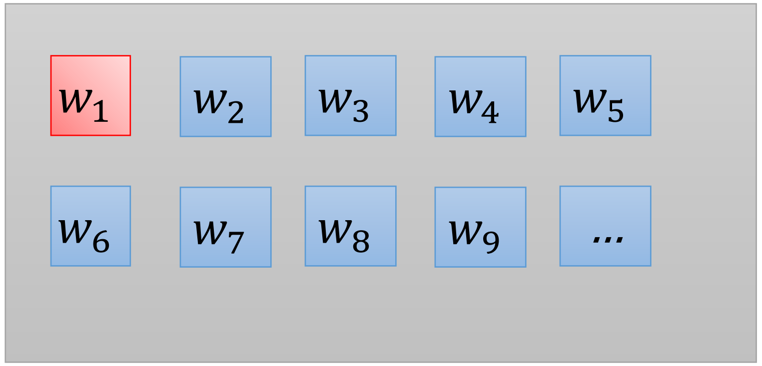
Feature-Name [ID]	Feature Value
Current Word	w_2
Previous Word	w_1
Next Word	w_3

Feature-Name = Value
Current Word = w_1
Current Word = w_2
Previous Word = start
Previous Word = w_1
Next Word = w_2
Next Word = w_3
… many many more

Features in Maximum Entropy

- We did not deal with one aspect:
- „the word = Markus is 4 times more likely to be a noun than a verb“
- Let us aggregate all our tables into one huge table

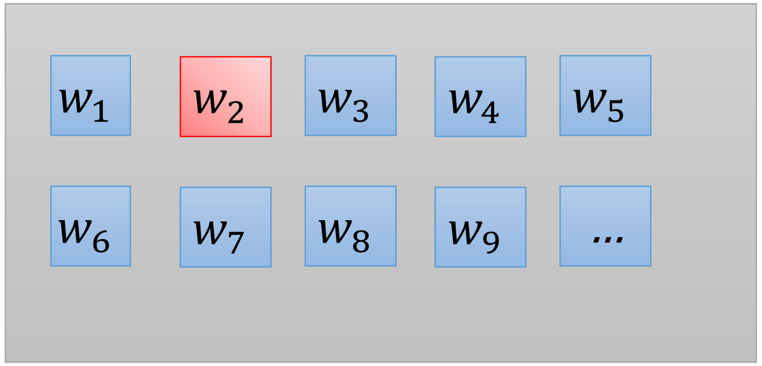
Feature-Name = Value	Active
Current Word = w_1	1
Current Word = w_2	0
Previous Word = start	1
Previous Word = w_1	0
Next Word = w_2	1
Next Word = w_3	0



Features in Maximum Entropy

- We did not deal with one aspect:
- „the word = Markus is 4 times more likely to be a noun than a verb“
- Let us aggregate all our tables into one huge table

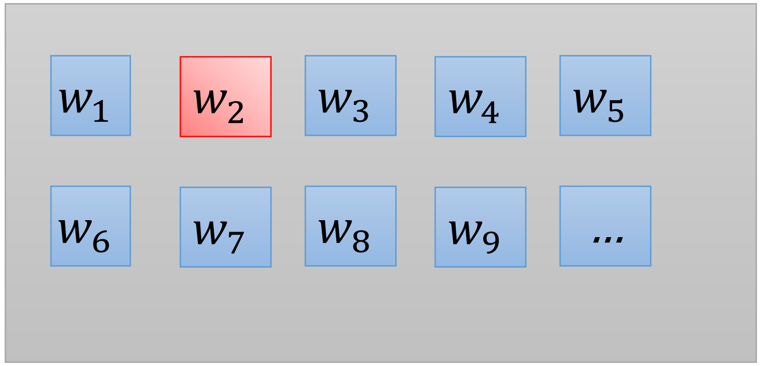
Feature-Name = Value	Active
Current Word = w_1	0
Current Word = w_2	1
Previous Word = start	0
Previous Word = w_1	1
Next Word = w_2	0
Next Word = w_3	1



Features in Maximum Entropy

- We can now describe our entire data set using a common vector of these features, but how does this help to express „more likely“

Feature-Name = Value	Active
Current Word = w_1	0
Current Word = w_2	1
Previous Word = start	0
Previous Word = w_1	1
Next Word = w_2	0
Next Word = w_3	1

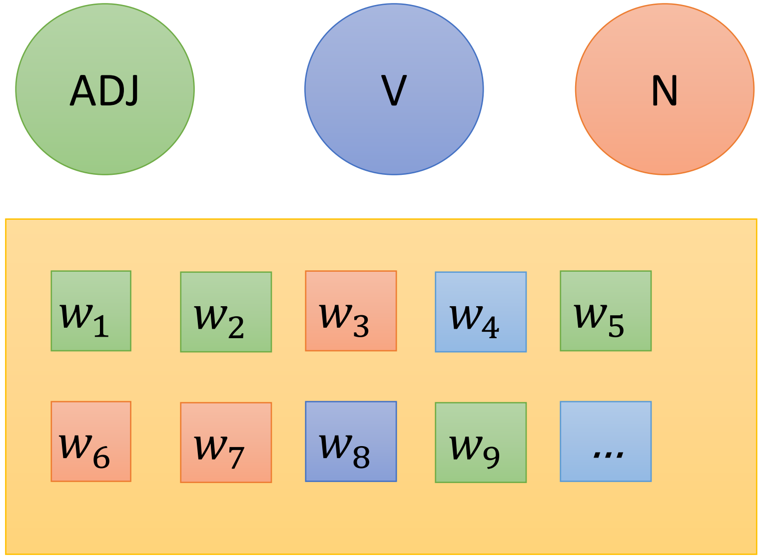


Features in Maximum Entropy

➔ Introduce labelled data

➔ We can now also encode the labels into our features

Legend: Colors of labels



Feature-Name = Value	Active
Current Word = w_1	0
Current Word = w_2	1
Previous Word = start	0
Previous Word = w_1	1
Next Word = w_2	0
Next Word = w_3	1

Features in Maximum Entropy

➔Introduce labelled data

➔We can now also encode the labels into our features

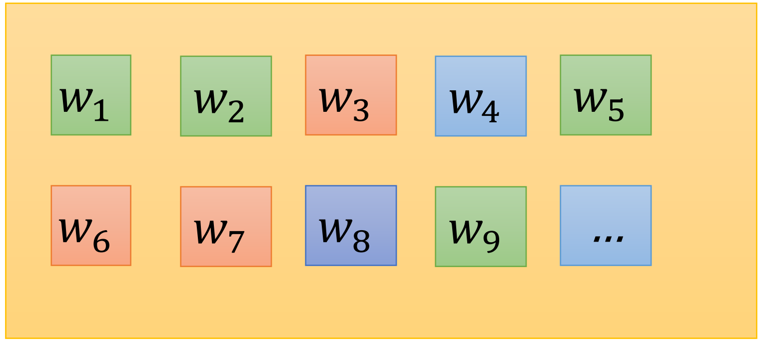
Feature-Name = Value	Active
Current Word = w_1	0
Current Word = w_2	1
Previous Word = start	0
Previous Word = w_1	1
Next Word = w_2	0
Next Word = w_3	1

Feature-Name = Value	Current-Label	Active
Current Word = w_1	ADJ	
Current Word = w_1	N	
Current Word = w_1	V	
Current Word = w_2	ADJ	
Current Word = w_2	N	
Current Word = w_2	V	
Previous Word = w_1	ADJ	
Previous Word = w_1	N	
Previous Word = w_1	V	
…		

Features in Maximum Entropy

- Let us redescribe our words
- For w_1

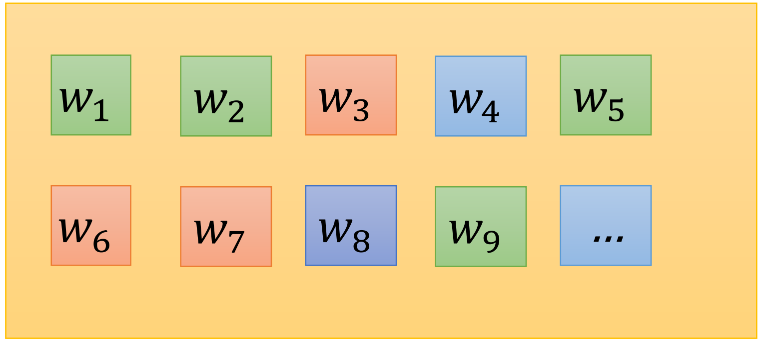
Feature-Name = Value	Current-Label	Active
Current Word = w_1	ADJ	1
Current Word = w_1	N	0
Current Word = w_1	V	0
Current Word = w_2	ADJ	0
Current Word = w_2	N	0
Current Word = w_2	V	0
Previous Word = w_1	ADJ	0
Previous Word = w_1	N	0
Previous Word = w_1	V	0
…		



Features in Maximum Entropy

- Let us redescribe our words
- For w_2

Feature-Name = Value	Current-Label	Active
Current Word = w_1	ADJ	0
Current Word = w_1	N	0
Current Word = w_1	V	0
Current Word = w_2	ADJ	1
Current Word = w_2	N	0
Current Word = w_2	V	0
Previous Word = w_1	ADJ	1
Previous Word = w_1	N	0
Previous Word = w_1	V	0
…		



Features in Maximum Entropy

- Let us now sum all these tables of all words

Feature-Name = Value	Current-Label	Active
Current Word = w_1	ADJ	1
Current Word = w_1	N	80
Current Word = w_1	V	20
Current Word = w_2	ADJ	43
Current Word = w_2	N	12
Current Word = w_2	V	1337
Previous Word = w_1	ADJ	42
Previous Word = w_1	N	11
Previous Word = w_1	V	0
…		

Features in Maximum Entropy

- Assume $w_1 = \text{Markus}$
- We would expect to have it labelled as a noun N
- Our belief is roughly

$$p(N|w = \text{Markus}) \sim \frac{80}{1 + 80 + 20} \sim 80\%$$

$$p(V|w = \text{Markus}) \sim \frac{20}{1 + 80 + 20} \sim 20\%$$

We did it! We now have an intuition how we can express something to be more likely than something else, just using features

Feature-Name = Value	Current-Label	Active
Current Word = w_1	ADJ	1
Current Word = w_1	N	80
Current Word = w_1	V	20
Current Word = w_2	ADJ	43
Current Word = w_2	N	12
Current Word = w_2	V	1337
Previous Word = w_1	ADJ	42
Previous Word = w_1	N	11
Previous Word = w_1	V	0
…		

Features in Maximum Entropy

Feature-Name = Value	Current-Label	Active
Current Word = w_1	ADJ	1
Current Word = w_1	N	80
Current Word = w_1	V	20
Current Word = w_2	ADJ	43
Current Word = w_2	N	12
Current Word = w_2	V	1337
Previous Word = w_1	ADJ	42
Previous Word = w_1	N	11
Previous Word = w_1	V	0
…		

Principle of Maximum Entropy (Jaynes, 1957)

If incomplete information about a probability distribution is available, the only unbiased assumption that can be made is a distribution which is as uniform as possible, given the available information.

Even though we counted 0, this might be due to our limited amount of training data, so we assume that we do not know anything about this!

Features in Maximum Entropy

- Now we want to learn a classifier that is exactly able to reproduce this table
  - We will call such a classifier „consistent“ to our observations!

Feature-Name = Value	**Current-Label**	**Active**
Current Word = w_1	ADJ	1
Current Word = w_1	N	80
Current Word = w_1	V	20
Current Word = w_2	ADJ	43
Current Word = w_2	N	12
Current Word = w_2	V	1337
Previous Word = w_1	ADJ	42
Previous Word = w_1	N	11
Previous Word = w_1	V	0
…		

Features in Maximum Entropy

- Let us merge some columns again
- Each row now corresponds to a distinct feature $f(x, y)$ which is calculated form the input x and the label y

Feature-Name = Value ∧ CurrentLabel	Active
Current Word = $w_1 \wedge \mathrm{ADJ}$	1
Current Word = $w_1 \wedge \mathrm{N}$	80
Current Word = $w_1 \wedge \mathrm{V}$	20
Current Word = $w_2 \wedge \mathrm{ADJ}$	43
Current Word = $w_2 \wedge \mathrm{N}$	12
Current Word = $w_2 \wedge \mathrm{V}$	1337
Previous Word = $w_1 \wedge \mathrm{ADJ}$	42
Previous Word = $w_1 \wedge \mathrm{N}$	11
Previous Word = $w_1 \wedge \mathrm{V}$	0
…	

Features in Maximum Entropy

- We can now represent all data in a single vector! And the semantics of each index can be stored somewhere else

Feature-Name = Value ∧ CurrentLabel
Current Word = $w_1 \wedge \mathrm{ADJ}$
Current Word = $w_1 \wedge \mathrm{N}$
Current Word = $w_1 \wedge \mathrm{V}$
Current Word = $w_2 \wedge \mathrm{ADJ}$
Current Word = $w_2 \wedge \mathrm{N}$
Current Word = $w_2 \wedge \mathrm{V}$
Previous Word = $w_1 \wedge \mathrm{ADJ}$
Previous Word = $w_1 \wedge \mathrm{N}$
Previous Word = $w_1 \wedge \mathrm{V}$
…

Active
1
80
20
43
12
1337
42
11
0

Features in Maximum Entropy

- A training example is also just such a vector

Feature-Name = Value	Active
Current Word = $w_1 \wedge \mathrm{ADJ}$	1
Current Word = $w_1 \wedge \mathrm{N}$	0
Current Word = $w_1 \wedge \mathrm{V}$	0
Current Word = $w_2 \wedge \mathrm{ADJ}$	0
Current Word = $w_2 \wedge \mathrm{N}$	0
Current Word = $w_2 \wedge \mathrm{V}$	0
Previous Word = $w_1 \wedge \mathrm{ADJ}$	0
Previous Word = $w_1 \wedge \mathrm{N}$	0
Previous Word = $w_1 \wedge \mathrm{V}$	0
…	

Features in Maximum Entropy

- Let us inspect the table a little more

Feature-Name = Value	Active
Current Word = $w_1 \wedge \mathrm{ADJ}$	1
Current Word = $w_1 \wedge \mathrm{N}$	0
Current Word = $w_1 \wedge \mathrm{V}$	0
Current Word = $w_2 \wedge \mathrm{ADJ}$	0
Current Word = $w_2 \wedge \mathrm{N}$	0
Current Word = $w_2 \wedge \mathrm{V}$	0
Previous Word = $w_1 \wedge \mathrm{ADJ}$	0
Previous Word = $w_1 \wedge \mathrm{N}$	0
Previous Word = $w_1 \wedge \mathrm{V}$	0
…	

Only one of each group can be active

Features in Maximum Entropy

- We can also write a probability distribution to each group

Feature-Name = Value	Active
Current Word = $w_1 \wedge \mathrm{ADJ}$	1
Current Word = $w_1 \wedge \mathrm{N}$	0
Current Word = $w_1 \wedge \mathrm{V}$	0
Current Word = $w_2 \wedge \mathrm{ADJ}$	0
Current Word = $w_2 \wedge \mathrm{N}$	0
Current Word = $w_2 \wedge \mathrm{V}$	0
Previous Word = $w_1 \wedge \mathrm{ADJ}$	0
Previous Word = $w_1 \wedge \mathrm{N}$	0
Previous Word = $w_1 \wedge \mathrm{V}$	0
…	

100%
0%
0%

We observe this feature with ADJ, since it is labelled that way!

➔ If a classifier is not certain about this labeling, its distribution would be somewhat different

Features in Maximum Entropy

- We can also write a probability distribution to each group

Feature-Name = Value	Active	Classifier
Current Word = $w_1 \wedge \mathrm{ADJ}$	1	0.8
Current Word = $w_1 \wedge \mathrm{N}$	0	0.1
Current Word = $w_1 \wedge \mathrm{V}$	0	0.1
Current Word = $w_2 \wedge \mathrm{ADJ}$	0	
Current Word = $w_2 \wedge \mathrm{N}$	0	
Current Word = $w_2 \wedge \mathrm{V}$	0	
Previous Word = $w_1 \wedge \mathrm{ADJ}$	0	
Previous Word = $w_1 \wedge \mathrm{N}$	0	
Previous Word = $w_1 \wedge \mathrm{V}$	0	
…		

Classifier believes that this example is labelled ADJ with 80% certainty; 10% Noun and 10% Verb

Features in Maximum Entropy

- So the classifier provides us with his „soft counts" (in fact these are expectations)

Feature-Name = Value	Active	Classifier
Current Word = $w_1 \wedge \mathrm{ADJ}$	1	0.8
Current Word = $w_1 \wedge \mathrm{N}$	0	0.1
Current Word = $w_1 \wedge \mathrm{V}$	0	0.1
Current Word = $w_2 \wedge \mathrm{ADJ}$	0	
Current Word = $w_2 \wedge \mathrm{N}$	0	
Current Word = $w_2 \wedge \mathrm{V}$	0	
Previous Word = $w_1 \wedge \mathrm{ADJ}$	0	
Previous Word = $w_1 \wedge \mathrm{N}$	0	
Previous Word = $w_1 \wedge \mathrm{V}$	0	
…		

Features in Maximum Entropy

- So we can now produce two vectors, one with observed counts and one with „predicted counts“

Feature-Name = Value ∧ CurrentLabel	Observed	Predicted
Current Word = $w_1 \wedge \mathrm{ADJ}$	1	0.8
Current Word = $w_1 \wedge \mathrm{N}$	80	56.4
Current Word = $w_1 \wedge \mathrm{V}$	20	13.4
Current Word = $w_2 \wedge \mathrm{ADJ}$	43	22.9
Current Word = $w_2 \wedge \mathrm{N}$	12	9.4
Current Word = $w_2 \wedge \mathrm{V}$	1337	1228.8
Previous Word = $w_1 \wedge \mathrm{ADJ}$	42	56.3
Previous Word = $w_1 \wedge \mathrm{N}$	11	14.5
Previous Word = $w_1 \wedge \mathrm{V}$	0	0.5
…	…	…

Features in Maximum Entropy

- Goal now is to train a classifier, which is capable of reproducing the observed counts
- In fact we will derive that the difference between these 2 vectors corresponds to the gradient of the „MaximumEntropy“ classifier
  - And the gradient is all we need to use Gradient Descent!

Observed
1
80
20
43
12
1337
42
11
0

Predicted
0.8
56.4
13.4
22.9
9.4
1228.8
56.3
14.5
0.5

Maximum Entropy Model

- A MaxEnt classifier predicts a probability for a label y, given some feature vector x: $\mathrm{p}(y|x)$
- Starting with the (Conditional Entropy)

$$H(Y|X) = -\sum_{(x,y)\in D} p(y,x)\log p(y|x)$$

- Our goal is to find a model $p^*(y|x)$, which:

1. Maximizes the Conditional Entropy $H(Y|X)$ (Maximum Entropy)
2. Is consistent to the training data (we will come to that in a moment)

Maximum Entropy Model: Goal

$$p^*(y|x) = \max_{p(y|x) \in P} H(Y|X)$$

→With P being the set of consistent models (still to be modelled)

- We call a model consistent to our data, if it is consistent with respect to its "features"
  - Introduce "features" or "feature functions" :

$$f_i(x, y) = \begin{cases} 1, if\ a\ boolean\ expression\ is\ true \\ 0, otherwise \end{cases}$$

  - Example:

$$f_i(x, y) = \begin{cases} 1, if\ currentWord = running\ and\ y = verb \\ 0, otherwise \end{cases}$$

Maximum Entropy Model: Consistency

- But what is consistency with respect to a feature ?

- For every feature function f_i, we can count how often it appears in our training data, and we can receive an expected value $\hat{E}(f_i)$:

$$\hat{E}(f_i) = \frac{1}{N} \sum_{(x,y) \in D} f_i(x, y)$$

- A valid model is a model, that can reproduce this expected value

Maximum Entropy Model: Consistency

- We assumed, that all training instances x are independent (and therefore equally probable), in general we would write:

$$\hat{E}(f_i) = \sum_{(x,y)\in D} p(x,y) \cdot f_i(x,y)$$

- We will now derive the same expected value for the model distribution, we will denote it as $E(f_i)$

Maximum Entropy Model: Consistency

- Our model predicts $p(y|x)$ so we have to do some work!

$$E(f_i) = \sum_{(x,y)\in D} p(x,y) \cdot f_i\,(x,y)$$

$$E(f_i) = \sum_{(x,y)\in D} p(x) \cdot p(y|x) \cdot f_i(x,y)$$

Bayes

- But we will yet again assume our instances are independant, so $p(x)$ is the same for every instance, and we can average again

$$E(f_i) = \frac{1}{N} \sum_{(x,y)\in D} p(y|x) \cdot f_i(x,y)$$

Maximum Entropy Model

- We can now force (for every feature), that

$$\hat{E}(f_i) = E(f_i)$$

- Every model that satisfies this is valid according to our observed data!
- We furthermore want to have real probability distribution:

$$p(y|x) \geq 0, for\ all\ x, y$$

And

$$\sum_{y \in Y} p(y|x) = 1, for\ all\ x$$

Maximum Entropy Model

- The full story:
  - Find $p(y|x)$ so that:

$$p^*(y|x) = \max_{p(y|x)\in P} H(Y|X)$$

Subject to:

$$\hat{E}(f_i) - E(f_i) = 0, \qquad \forall f_i$$

$$1\text{-}\sum_{y\in Y} p(y|x) = 0, \ \forall\ x$$

And

$$p(y|x) \geq 0, \qquad \forall x, y$$

Maximum Entropy Model

$$p^*(y|x) = \max_{p(y|x) \in P} H(Y|X)$$

Subject to:

$\hat{E}(f_i) - E(f_i) = 0, \quad \forall f_i$

$1\text{-}\sum_{y \in Y} p(y|x) = 0, \ \forall \ x$

And

$p(y|x) \geq 0, \quad \forall x, y$

- Solving the optimization problem
  - Optimization without conditions is easy
  - But there is a mathematical trick to get rid of them
  - ➔Introduce Lagrangian parameter

Lagrange function:

$$L(p, \vec{\lambda}) = \underbrace{H(y|x)}_{\text{Entropie}} + \underbrace{\sum_i \lambda_i \left(E(f_i) - \hat{E}(f_i)\right)}_{\text{Valid models}} + \underbrace{\lambda_{m+1}(1 - \sum_{y \in Y} p(y_i|x))}_{\text{Prob. Distribution}}$$

➔We can now maximize this function instead!

Getting to the solution

Maximum Entropy Model

- Solution: The general form of a maximum entropy model (a "family" of models)

$$p(y|x) = \frac{\exp \sum_{i=1} \lambda_i f_i(x, y)}{\sum_{y' \in Y} \exp(\sum_{i=1} \lambda_i f_i(x, y'))}$$

- With model parameters λ_i, that are optimized for every task!

Maximum Entropy Model

- What you should understand until now:
  - Beginning with the principle of Maximum Entropy, we found a family of functions ("Exponential Family") that can be used to classify!

$$p(y|x) = \frac{\exp \sum_{i=1} \lambda_i f_i(x, y)}{\sum_{y' \in Y} \exp(\sum_{i=1} \lambda_i f_i(x, y'))}$$

- You do not know anything about how to get those λ_i!

Maximum Entropy Model

- Let us assume all λ_i are 0

$$p(y|x) = \frac{\exp \sum_{i=1} \lambda_i f_i(x,y)}{\sum_{y' \in Y} \exp(\sum_{i=1} \lambda_i f_i(x,y'))} = \frac{1}{\sum_{y' \in Y} 1} = \frac{1}{\#\, labels}$$

➔If we have no information, we will just predict a uniform distribution! (Principle of Maximum Entropy)

➔The same would hold of no feature function would be active!

Maximum Entropy Model: Example

- Problem: Determine the POS-Tag for the word “**Grace**” in the sentence:

“at **Grace** Road”

- We are given some weights λ (we assume a trained classifier for now)

Feature	Weights for Class NN	Weights for Class V
PreviousWord=at	0.35	-0.2
CurrentWord=Grace	0.7	0.05
NextWord=Road	0.4	0.3

Maximum Entropy Model: Example

Feature	Weights for Class NN	Weights for Class V
PreviousWord=at	0.35	-0.2
CurrentWord=Grace	0.7	0.05
NextWord=Road	0.4	0.3

- Y= NN and x =“at Grace Road”:

$$p(NN|x) = \frac{\exp(0.35 + 0.7 + 0.4)}{\sum_{y' \in Y} \exp(\sum_{i=1} \lambda_i f_i(x, y'))}$$

Maximum Entropy Model: Example

Feature	Weights for Class NN	Weights for Class V
PreviousWord=at	0.35	-0.2
CurrentWord=Grace	0.7	0.05
NextWord=Road	0.4	0.3

- Y= V and x =“at Grace Road”:

$$p(V|x) = \frac{\exp(-0.2 + 0.05 + 0.3)}{\sum_{y' \in Y} \exp(\sum_{i=1} \lambda_i f_i(x, y'))}$$

Maximum Entropy Model:Example

- Denominator:

$$\sum_{y' \in \{NN, V\}} \exp\Big(\sum_{i=1} \lambda_i f_i(x, y')\Big)$$

$$= \exp(0.35 + 0.7 + 0.4) + \exp(-0.2 + 0.05 + 0.3)$$

$$= \exp(1.45) + \exp(0.15)$$

Maximum Entropy Model:Example

$$p(NN|x) = \frac{\exp(0.35+0.7+0.4)}{\exp(1.45)+\exp(0.15)}=0.78$$

$$p(V|x) = \frac{\exp(-0.2+0.05+0.3)}{\exp(1.45)+\exp(0.15)}= 0.22$$

➔We would classify it as a noun

Maximum Entropy Model: Example

- A different view:

$$p(V|x) = \frac{\text{score}(x, V)}{\text{score}(x, V) + \text{score}(x, NN)}$$

➔In essence the weights show the importance of a feature and an instance gets the score as the sum of all active features

➔ The denominator is just there to normalize our distribution

Maximum Entropy Models

Parameter Learning

Roadmap

- We have found a family of functions „Exponential Family“, which has some parameters λ_i
- We can now choose an appropriate Loss function
  - Then form the gradient of the loss with respect to every λ_i
  - Use Gradient Descent to optimize the parameters

Maximum Entropy: Parameter Learning

- So far we derived the classifier, but we left how we determine the $\lambda`s$

- In fact in order to „train“ the classifier, we need:
  - some data D consisting of tuples (x, y)
  - A loss function (which tells us which lambda values are better than others)

- This classifier is usually trained using „Maximum Likelihood“

Maximum Entropy: Parameter Learning

- We can do this, using “Maximum Likelihood”:

$$p(y|D,\lambda) = \prod_{(x,y)\in D} p(y|x)$$

- We take the logarithm (numerical stability)

$$p(y|D,\lambda) = \sum_{(x,y)\in D} \log p(y|x)$$

➔ Maximum, when the model always predicts the correct class y with p(y|x)=1

Maximum Entropy Model: Parameter Learning

- Taking the log, for easier computations ("Log Likelihood"):

$$p(y|D,\lambda) = \sum_{(x,y)\in D} \log p(y|x)$$

$$= \sum_{(x,y)\in D} \log \frac{\exp(\sum_{i=1} \lambda_i f_i(x,y))}{\sum_{y'\in Y} \exp(\sum_{i=1} \lambda_i f_i(x,y'))}$$

$$= \sum_{(x,y)\in D} \log \exp\left(\sum_{i=1} \lambda_i f_i(x,y)\right) - log \sum_{y\prime\in Y} \exp(\sum_{i=1} \lambda_i f_i(x,y'))$$

➔find parameter so that this is maximized!

➔Equivalently you can minimize the negative log likelihood

Maximum Entropy Model: Parameter Learning

- In order to be able to optimize the parameters using Gradient Descent, we still need the gradient…
- Objective function:

$$\sum_{(x,y)\in D} \underbrace{\log \exp\left(\sum_{i=1} \lambda_i f_i(x,y)\right)}_{\text{A}} - \underbrace{log \sum_{y\prime\in Y} \exp(\sum_{i=1} \lambda_i f_i(x,y'))}_{\text{B}}$$

The Derivative I: Numerator (A)

$$\frac{\partial A}{\partial \lambda_i} = \frac{\partial \sum_{x,y} \log e^{\sum_i \lambda_i f_i(x,y)}}{\partial \lambda_i}$$

$$= \frac{\partial \sum_{x,y} \sum_i \lambda_i f_i(x,y)}{\partial \lambda_i}$$

$$= \sum_{x,y} f_i(x,y)$$

➔Derivative of the numerator is: the empirical count(f_i, y)

The Derivative II: Denominator

$$\frac{\partial B}{\partial \lambda_i} = \frac{\partial \sum_{x,y} \log \sum_{y\prime} e^{\sum_i \lambda_i f_i(y\prime,x)}}{\partial \lambda_i}$$

$$= \sum_{x,y} \frac{1}{\sum_{y\prime\prime} e^{\sum_i \lambda_i f_i(y\prime\prime,x)}} \cdot \frac{\partial \sum_{y'} e^{\sum_i \lambda_i f_i(y',x)}}{\partial \lambda_i}$$

$$= \sum_{y,x} \frac{1}{\sum_{y\prime\prime} e^{\sum_i \lambda_i f_i(y\prime\prime,x)}} \cdot \sum_{y\prime} e^{\sum_i \lambda_i f_i(y',x)} \frac{\partial \sum_i \lambda_i f_i(y',x)}{\partial \lambda_i}$$

$$= \sum_{y,x} \sum_{y\prime} \frac{e^{\sum_i \lambda_i f_i(y\prime,x)}}{\sum_{y\prime\prime} e^{\sum_i \lambda_i f_i(y\prime\prime,x)}} \cdot \frac{\partial \sum_i \lambda_i f_i(y',x)}{\partial \lambda_i}$$

$$= \sum_{x,y} \sum_{y\prime} p(y'|x,\lambda) \cdot f_i(y',x) = \text{predicted count}(f_i,\lambda)$$

The Derivative III

$$\frac{\partial \log P(C \mid D, \lambda)}{\partial \lambda_i} = \text{actual count}(f_i, C) - \text{predicted count}(f_i, \lambda)$$

- The optimum parameters are the ones for which each feature's predicted expectation equals its empirical expectation. The optimum distribution is:
  - Always unique (but parameters may not be unique)
  - Always exists (if feature counts are from actual data).

Recap: Maximum Entropy

- We started with the definition of Entropy
- Went through some ugly math to find our family $p(y|x)$

- Learned how to optimize the parameters λ_i of our model
- Learned how to apply the model to new data

➔ We could now use a Maximum Entropy classifier to derive our node scores! And we can integrate arbitrary features!

Example: Maximum Entropy

- Running example:
  - Determine the POS-Tags of the sentence

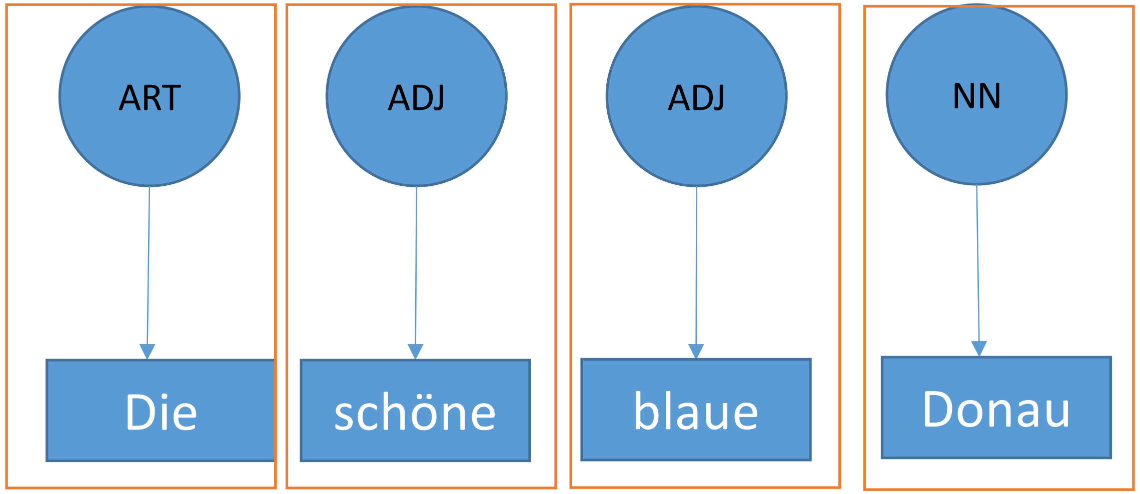
Die schöne blaue Donau

The beautiful blue Danube

  - Available POS-Tags: {ART,ADJ,NN}

Example: Maximum Entropy

- We use a local MaxEnt at every token

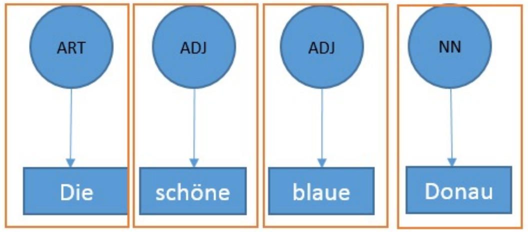


The beautiful blue Danube

➔4 local classifier (we assume it is always the same classifier, reused)

➔Only available feature template $f(\boldsymbol{x}, y_i)$
  - That is, features can be calculated from the entire word sequence and the label at the current position

Example: MaxEnt

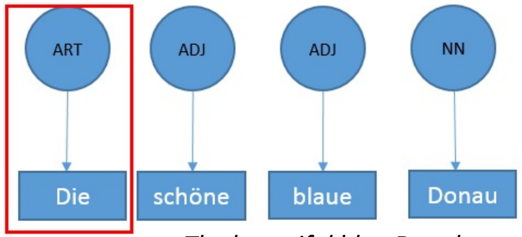


The beautiful blue Danube

- Given example features and their weights, trained on an arbitrary data set

Feature	λ for ART	λ for ADJ	λ for NN
CurrentWord=Die	0.6	0.1	0.25
CurrentWord=schöne	-0.1	0.8	0.3
CurrentWord=blaue	0.1	0.6	0.2
CurrentWord=Donau	0.1	0.1	1.4

Example: MaxEnt



The beautiful blue Danube

Feature	λ for ART	λ for ADJ	λ for NN
CurrentWord=Die	0.6	0.1	0.25
CurrentWord=schöne	-0.1	0.8	0.3
CurrentWord=blaue	0.1	0.6	0.2
CurrentWord=Donau	0.1	0.1	1.4

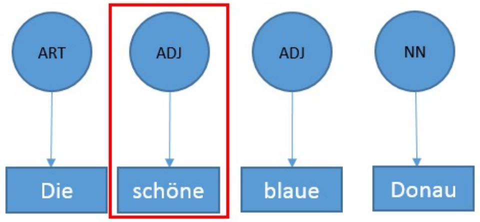
$$p(ART|x) = \frac{\exp(0.6)}{\exp(0.6)+\exp(0.1)+\exp(0.25)} = 0.44$$

$$p(ADJ|x) = \frac{\exp(0.1)}{\exp(0.6)+\exp(0.1)+\exp(0.25)} = 0.26$$

$$p(NN|x) = \frac{\exp(0.25)}{\exp(0.6)+\exp(0.1)+\exp(0.25)} = 0.30$$

➔”Die” is an ART

Example: MaxEnt



The beautiful blue Danube

Feature	λ for ART	λ for ADJ	λ for NN
CurrentWord=Die	0.6	0.1	0.25
CurrentWord=schöne	-0.1	0.8	0.3
CurrentWord=blaue	0.1	0.6	0.2
CurrentWord=Donau	0.1	0.1	1.4

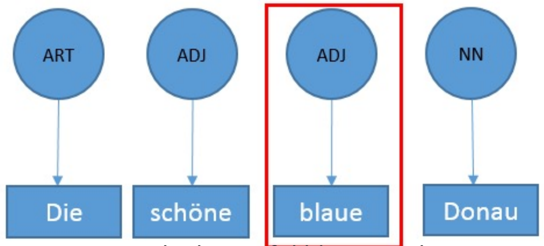
$$p(ART|x) = \frac{\exp(-0.1)}{\exp(-0.1)+\exp(0.8)+\exp(0.3)} = 0.2$$

$$p(ADJ|x) = \frac{\exp(0.8)}{\exp(-0.1)+\exp(0.8)+\exp(0.3)} = 0.5$$

$$p(NN|x) = \frac{\exp(0.3)}{\exp(-0.1)+\exp(0.8)+\exp(0.3)} = 0.3$$

➔”schöne” is an ADJ

Example: MaxEnt



The beautiful blue Danube

Feature	λ for ART	λ for ADJ	λ for NN
CurrentWord=Die	0.6	0.1	0.25
CurrentWord=schöne	-0.1	0.8	0.3
CurrentWord=blaue	0.1	0.6	0.2
CurrentWord=Donau	0.1	0.1	1.4

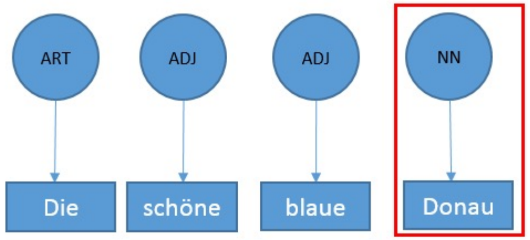
$$p(ART|x) = \frac{\exp(0.1)}{\exp(0.1)+\exp(0.6)+\exp(0.2)} = 0.26$$

$$p(ADJ|x) = \frac{\exp(0.6)}{\exp(0.1)+\exp(0.6)+\exp(0.2)} = 0.44$$

$$p(NN|x) = \frac{\exp(0.2)}{\exp(0.1)+\exp(0.6)+\exp(0.2)} = 0.3$$

➔”blaue” is an ADJ

Example: MaxEnt



The beautiful blue Danube

Feature	λ for ART	λ for ADJ	λ for NN
CurrentWord=Die	0.6	0.1	0.25
CurrentWord=schöne	-0.1	0.8	0.3
CurrentWord=blaue	0.1	0.6	0.2
CurrentWord=Donau	0.1	0.1	1.4

$$p(ART|x) = \frac{\exp(0.1)}{\exp(0.1)+\exp(0.1)+\exp(1.4)} = 0.176$$

$$p(ADJ|x) = \frac{\exp(0.1)}{\exp(0.1)+\exp(0.1)+\exp(1.4)} = 0.176$$

$$p(NN|x) = \frac{\exp(1.4)}{\exp(0.1)+\exp(0.1)+\exp(1.4)} = 0.647$$

➔"Donau" is a NN

Maximum Entropy: Recap

- Only local information is used

  ➔ Decoding is always greedy
- Only one feature template $f(\boldsymbol{x}, y_i)$
- Still manages to classify the sequence correct, because the features are expressive enough
- In general the used features do not generalize to unseen words