Natural Language Models and Interfaces

BSc Artificial Intelligence

Lecturer: Wilker Aziz
Institute for Logic, Language, and Computation

2020, week 5

Data sparsity

We have so far dealt with categorical models using tabular CPDs

- we've encountered problems for maximum likelihood estimation due to data sparsity
- large n-grams lead to large tables $O(v^n)$
- even the emission distributions of HMMs had to be smoothed
- smoothing techniques can be rather brittle

Are there principled ways to tackle data sparsity?

Data sparsity

We have so far dealt with categorical models using tabular CPDs

- we've encountered problems for maximum likelihood estimation due to data sparsity
- large n-grams lead to large tables $O(v^n)$
- even the emission distributions of HMMs had to be smoothed
- smoothing techniques can be rather brittle

Are there principled ways to tackle data sparsity?

Let's check a running example based on *sentiment classification*

Example: sentiment classification

How can we identify whether a sentence is positive or negative towards a subject?

- This movie is slow and repetitive, clearly the direction was careless and the production cheap.
- The movie is quite funny, its spiced humor makes it very interesting.

Example: sentiment classification

How can we identify whether a sentence is positive or negative towards a subject?

- This movie is slow and repetitive, clearly the direction was careless and the production cheap.
- The movie is quite funny, its spiced humor makes it very interesting.

If y is the sentiment of some text x, then we would like to compute $P_{Y|X}(y|x)$, but

- x is very sparse!
- how could we possibly parameterise it?

Basic Probability

One way to model $P_{Y|X}(y|x)$ is to model it directly,
i.e. directly parameterising a distribution over $\mathcal{X} \times \mathcal{Y}$.

Basic Probability

One way to model $P_{Y|X}(y|x)$ is to model it directly,
i.e. directly parameterising a distribution over $\mathcal{X} \times \mathcal{Y}$.

Another way is to infer this conditional from a joint distribution $P_{XY}(x, y)$.

Basic Probability

One way to model $P_{Y|X}(y|x)$ is to model it directly,
i.e. directly parameterising a distribution over $\mathcal{X} \times \mathcal{Y}$.

Another way is to infer this conditional from a joint distribution $P_{XY}(x, y)$. But how is this useful?

Basic Probability

One way to model $P_{Y|X}(y|x)$ is to model it directly,
i.e. directly parameterising a distribution over $\mathcal{X} \times \mathcal{Y}$.

Another way is to infer this conditional from a joint distribution $P_{XY}(x, y)$. But how is this useful?

Because we have an opportunity to make conditional independence assumptions and directly parameterise distributions over smaller sample spaces instead!

Feature functions

Suppose we identify a number of words which typically express sentiment, let's call them *features*

Class	Features (or attributes)
Negative	cheap, slow, repetitive, careless, awful, bad, ...
Positive	funny, spiced, interesting, awesome, good, ...

and let's say we have a vocabulary $\mathcal{F}$ of v such *sentiment words*

Feature functions

Suppose we identify a number of words which typically express sentiment, let's call them *features*

Class	Features (or attributes)
Negative	cheap, slow, repetitive, careless, awful, bad, ...
Positive	funny, spiced, interesting, awesome, good, ...

and let's say we have a vocabulary $\mathcal{F}$ of v such *sentiment words*

Then for an example $x =$ 'This film is fun though shy on the action' with sentiment $y = +$, let us retain only the sentiment words:

- $\langle$'fun', 'shy', 'though', 'action'$\rangle = \mathrm{sentwords}($'This film is fun though shy on the action'$)$

Feature functions

Suppose we identify a number of words which typically express sentiment, let's call them *features*

Class	Features (or attributes)
Negative	cheap, slow, repetitive, careless, awful, bad, ...
Positive	funny, spiced, interesting, awesome, good, ...

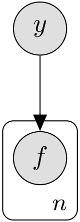
and let's say we have a vocabulary $\mathcal{F}$ of v such *sentiment words*

Then for an example $x =$ 'This film is fun though shy on the action' with sentiment $y = +$, let us retain only the sentiment words:

- $\langle$'fun', 'shy', 'though', 'action'$\rangle =$ $\mathrm{sentwords}($'This film is fun though shy on the action'$)$
- we will denote this by a random pair $(Y, \langle F_1, \ldots, F_n \rangle)$

Naive Bayes Classifiers

In a naive Bayes classifier, we assume the *class* y generates the features $\langle f_1, \ldots, f_n \rangle$

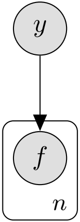


That is, we assume features to be *conditionally independent* given a *class*

$$P_{YF_1^n}(y, \langle f_1, \ldots, f_n \rangle) =$$

Naive Bayes Classifiers

In a naive Bayes classifier, we assume the *class* y generates the features $\langle f_1, \ldots, f_n \rangle$

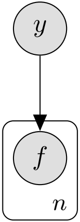


That is, we assume features to be *conditionally independent* given a *class*

$$P_{YF_1^n}(y, \langle f_1, \ldots, f_n \rangle) = P_Y(y) \times$$

Naive Bayes Classifiers

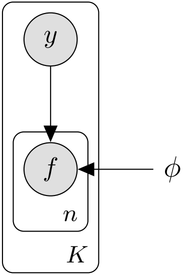
In a naive Bayes classifier, we assume the *class* y generates the features $\langle f_1, \ldots, f_n \rangle$



That is, we assume features to be *conditionally independent* given a *class*

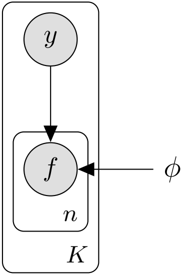
$$P_{YF_1^n}(y, \langle f_1, \ldots, f_n \rangle) = P_Y(y) \times \prod_{i=1}^{n} P_{F|Y}(f_i|y)$$

Naive Bayes Binary Classification



Suppose a dataset of labelled examples $\mathcal{D} = \left\{ y^{(k)}, \langle f_1^{(k)}, \ldots, f_{n_k}^{(k)} \rangle \right\}_{k=1}^{K}$ and a vocabulary of v features

Naive Bayes Binary Classification



Suppose a dataset of labelled examples $\mathcal{D} = \left\{ y^{(k)}, \langle f_1^{(k)}, \ldots, f_{n_k}^{(k)} \rangle \right\}_{k=1}^{K}$ and a vocabulary of v features

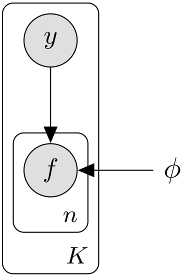
The generative story for each training instance:

$$Y \sim \mathcal{U}(1/2)$$

for $i = 1, \ldots, n$

$$F_i | y \sim \mathrm{Cat}(\phi_1^{(y)}, \ldots, \phi_v^{(y)})$$

Naive Bayes Binary Classification



Suppose a dataset of labelled examples $\mathcal{D} = \left\{ y^{(k)}, \langle f_1^{(k)}, \ldots, f_{n_k}^{(k)} \rangle \right\}_{k=1}^{K}$ and a vocabulary of v features

The generative story for each training instance:

$$Y \sim \mathcal{U}(1/2)$$

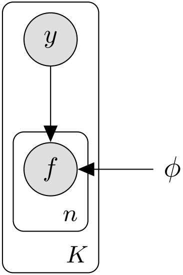
for $i = 1, \ldots, n$

$$F_i | y \sim \mathrm{Cat}(\phi_1^{(y)}, \ldots, \phi_v^{(y)})$$

Thus, for one training instance

$$P_{Y F_1^n}(y, \langle f_1, \ldots, f_n \rangle) = P_Y(y) \times \prod_{i=1}^{n} P_{F|Y}(f_i | y)$$

Naive Bayes Binary Classification



Suppose a dataset of labelled examples $\mathcal{D} = \left\{ y^{(k)}, \langle f_1^{(k)}, \ldots, f_{n_k}^{(k)} \rangle \right\}_{k=1}^{K}$ and a vocabulary of v features

The generative story for each training instance:

$$Y \sim \mathcal{U}(1/2)$$

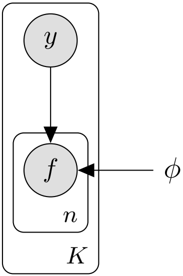
$$\text{for } i = 1, \ldots, n$$

$$F_i | y \sim \mathrm{Cat}(\phi_1^{(y)}, \ldots, \phi_v^{(y)})$$

Thus, for one training instance

$$\begin{aligned} P_{YF_1^n}(y, \langle f_1, \ldots, f_n \rangle) &= P_Y(y) \times \prod_{i=1}^{n} P_{F|Y}(f_i|y) \\ &= \mathcal{U}(1/2) \prod_{i=1}^{n} \mathrm{Cat}(f_i | \phi^{(y)}) \end{aligned}$$

Naive Bayes Binary Classification



Suppose a dataset of labelled examples $\mathcal{D} = \left\{ y^{(k)}, \langle f_1^{(k)}, \ldots, f_{n_k}^{(k)} \rangle \right\}_{k=1}^{K}$ and a vocabulary of v features

The generative story for each training instance:

$$Y \sim \mathcal{U}(1/2)$$

for $i = 1, \ldots, n$

$$F_i | y \sim \mathrm{Cat}(\phi_1^{(y)}, \ldots, \phi_v^{(y)})$$

Thus, for one training instance

$$P_{YF_1^n}(y, \langle f_1, \ldots, f_n \rangle) = P_Y(y) \times \prod_{i=1}^{n} P_{F|Y}(f_i|y)$$

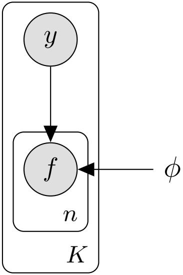
$$= \mathcal{U}(1/2) \prod_{i=1}^{n} \mathrm{Cat}(f_i|\phi^{(y)}) \propto \prod_{i=1}^{n} \phi_{f_i}^{(y)}$$

Naive Bayes Binary Classification: MLE

The log-likelihood of the data is proportional to

$$\log P(\mathcal{D}|\phi) = \prod_{k=1}^{K} P_{Y|F_1^n}(y_k, \langle f_1^{(k)}, \ldots, f_{n_k}^{(k)} \rangle | \phi)$$

$$\propto \sum_{k=1}^{K} \sum_{i=1}^{n} \log P_{F|Y}(f_i^{(k)} | y^{(k)}, \phi)$$

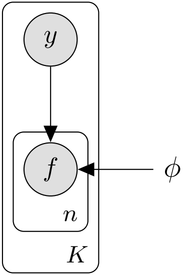


How many parameters?

Naive Bayes Binary Classification: MLE

The log-likelihood of the data is proportional to

$$\log P(\mathcal{D}|\phi) = \prod_{k=1}^{K} P_{Y|F_1^n}(y_k, \langle f_1^{(k)}, \ldots, f_{n_k}^{(k)} \rangle | \phi)$$
$$\propto \sum_{k=1}^{K} \sum_{i=1}^{n} \log P_{F|Y}(f_i^{(k)} | y^{(k)}, \phi)$$

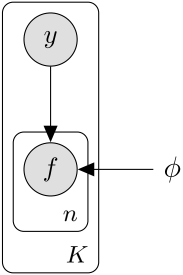


How many parameters? $2 \times v$

Naive Bayes Binary Classification: MLE

The log-likelihood of the data is proportional to

$$\log P(\mathcal{D}|\phi) = \prod_{k=1}^{K} P_{Y|F_1^n}(y_k, \langle f_1^{(k)}, \ldots, f_{n_k}^{(k)} \rangle | \phi)$$
$$\propto \sum_{k=1}^{K} \sum_{i=1}^{n} \log P_{F|Y}(f_i^{(k)} | y^{(k)}, \phi)$$



How many parameters? $2 \times v$

MLE

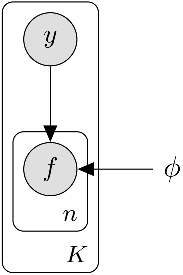
$$\phi_f^{(y)} =$$

Naive Bayes Binary Classification: MLE

The log-likelihood of the data is proportional to

$$\log P(\mathcal{D}|\phi) = \prod_{k=1}^{K} P_{Y|F_1^n}(y_k, \langle f_1^{(k)}, \ldots, f_{n_k}^{(k)} \rangle | \phi)$$

$$\propto \sum_{k=1}^{K} \sum_{i=1}^{n} \log P_{F|Y}(f_i^{(k)} | y^{(k)}, \phi)$$



How many parameters? $2 \times v$

MLE

$$\phi_f^{(y)} = \frac{\text{count}_{YF}(y, f)}{\text{count}_Y(y)}$$

Naive Bayes Binary Classification: Predictions

For some new example with features $\langle f_1, \ldots, f_n \rangle$, we can predict its class easily by solving a maximisation problem

$$y^\star = \operatorname*{argmax}_y \; P_{Y|F_1^n}(y|\langle f_1, \ldots, f_n \rangle)$$

Naive Bayes Binary Classification: Predictions

For some new example with features $\langle f_1, \ldots, f_n \rangle$, we can predict its class easily by solving a maximisation problem

$$\begin{aligned} y^\star &= \operatorname*{argmax}_y \; P_{Y|F_1^n}(y|\langle f_1, \ldots, f_n \rangle) \\ &= \operatorname*{argmax}_y \; \frac{P_Y(y) P_{F_1^n|Y}(\langle f_1, \ldots, f_n \rangle | y)}{P_{F_1^n}(\langle f_1, \ldots, f_n \rangle)} && \text{Bayes rule} \end{aligned}$$

Naive Bayes Binary Classification: Predictions

For some new example with features $\langle f_1, \ldots, f_n \rangle$, we can predict its class easily by solving a maximisation problem

$$
\begin{aligned}
y^\star &= \operatorname*{argmax}_y \; P_{Y|F_1^n}(y|\langle f_1, \ldots, f_n \rangle) \\
&= \operatorname*{argmax}_y \; \frac{P_Y(y) P_{F_1^n|Y}(\langle f_1, \ldots, f_n \rangle | y)}{P_{F_1^n}(\langle f_1, \ldots, f_n \rangle)} && \text{Bayes rule} \\
&= \operatorname*{argmax}_y \; P_Y(y) P_{F_1^n|Y}(\langle f_1, \ldots, f_n \rangle | y) && \text{Proportionality}
\end{aligned}
$$

Naive Bayes Binary Classification: Predictions

For some new example with features $\langle f_1, \ldots, f_n \rangle$, we can predict its class easily by solving a maximisation problem

$$
\begin{aligned}
y^\star &= \operatorname*{argmax}_y \; P_{Y|F_1^n}(y|\langle f_1, \ldots, f_n \rangle) \\
&= \operatorname*{argmax}_y \; \frac{P_Y(y) P_{F_1^n|Y}(\langle f_1, \ldots, f_n \rangle | y)}{P_{F_1^n}(\langle f_1, \ldots, f_n \rangle)} && \text{Bayes rule} \\
&= \operatorname*{argmax}_y \; P_Y(y) P_{F_1^n|Y}(\langle f_1, \ldots, f_n \rangle | y) && \text{Proportionality} \\
&= \operatorname*{argmax}_y \; P_{F_1^n|Y}(\langle f_1, \ldots, f_n \rangle | y) && \text{Proportionality}
\end{aligned}
$$

Naive Bayes Binary Classification: Predictions

For some new example with features $\langle f_1, \ldots, f_n \rangle$, we can predict its class easily by solving a maximisation problem

$$\begin{aligned}
y^\star &= \underset{y}{\operatorname{argmax}} \; P_{Y|F_1^n}(y|\langle f_1, \ldots, f_n \rangle) \\
&= \underset{y}{\operatorname{argmax}} \; \frac{P_Y(y) P_{F_1^n|Y}(\langle f_1, \ldots, f_n \rangle | y)}{P_{F_1^n}(\langle f_1, \ldots, f_n \rangle)} && \text{Bayes rule} \\
&= \underset{y}{\operatorname{argmax}} \; P_Y(y) P_{F_1^n|Y}(\langle f_1, \ldots, f_n \rangle | y) && \text{Proportionality} \\
&= \underset{y}{\operatorname{argmax}} \; P_{F_1^n|Y}(\langle f_1, \ldots, f_n \rangle | y) && \text{Proportionality} \\
&= \underset{y}{\operatorname{argmax}} \; \prod_{i=1}^{n} P_{F|Y}(f_i|y) && \text{Conditional independence}
\end{aligned}$$

Naive Bayes Binary Classification: Predictions

For some new example with features $\langle f_1, \ldots, f_n \rangle$, we can predict its class easily by solving a maximisation problem

$$
\begin{aligned}
y^\star &= \operatorname*{argmax}_y \; P_{Y|F_1^n}(y|\langle f_1, \ldots, f_n \rangle) \\
&= \operatorname*{argmax}_y \; \frac{P_Y(y) P_{F_1^n|Y}(\langle f_1, \ldots, f_n \rangle | y)}{P_{F_1^n}(\langle f_1, \ldots, f_n \rangle)} && \text{Bayes rule} \\
&= \operatorname*{argmax}_y \; P_Y(y) P_{F_1^n|Y}(\langle f_1, \ldots, f_n \rangle | y) && \text{Proportionality} \\
&= \operatorname*{argmax}_y \; P_{F_1^n|Y}(\langle f_1, \ldots, f_n \rangle | y) && \text{Proportionality} \\
&= \operatorname*{argmax}_y \; \prod_{i=1}^n P_{F|Y}(f_i|y) && \text{Conditional independence} \\
&= \operatorname*{argmax}_y \; \sum_{i=1}^n \log \phi_{f_i}^{(y)} && \text{Monotonicity of logarithm}
\end{aligned}
$$

Problem with Naive Bayes classification

Instead of computing $P_{Y|X}(y|x)$

- we compute $P_{Y|F_1^n}(y|f_1^n)$ using some features f_1^n of x
- instead of modelling $P_{Y|F_1^n}$ directly, we modelled $P_{F_1^n|Y}$ using tabular CPDs
- and got to $P_{Y|F_1^n}(y|f_1^n)$ via Bayes rule
  $P_{Y|F_1^n}(y|f_1^n) \propto P_Y(y)P_{F_1^n|Y}(f_1^n|y)$

Problem with Naive Bayes classification

Instead of computing $P_{Y|X}(y|x)$

- we compute $P_{Y|F_1^n}(y|f_1^n)$ using some features f_1^n of x
- instead of modelling $P_{Y|F_1^n}$ directly, we modelled $P_{F_1^n|Y}$ using tabular CPDs
- and got to $P_{Y|F_1^n}(y|f_1^n)$ via Bayes rule
  $P_{Y|F_1^n}(y|f_1^n) \propto P_Y(y)P_{F_1^n|Y}(f_1^n|y)$
- but isn't conditional independence a bit too strong?

Problem with Naive Bayes classification

Instead of computing $P_{Y|X}(y|x)$

- we compute $P_{Y|F_1^n}(y|f_1^n)$ using some features f_1^n of x
- instead of modelling $P_{Y|F_1^n}$ directly, we modelled $P_{F_1^n|Y}$ using tabular CPDs
- and got to $P_{Y|F_1^n}(y|f_1^n)$ via Bayes rule
  $P_{Y|F_1^n}(y|f_1^n) \propto P_Y(y)P_{F_1^n|Y}(f_1^n|y)$
- but isn't conditional independence a bit too strong?

Consider the following example

- *Some parts are a bit slow and a little repetitive, but overall not too bad.*

Problem with Naive Bayes classification

Instead of computing $P_{Y|X}(y|x)$

- we compute $P_{Y|F_1^n}(y|f_1^n)$ using some features f_1^n of x
- instead of modelling $P_{Y|F_1^n}$ directly, we modelled $P_{F_1^n|Y}$ using tabular CPDs
- and got to $P_{Y|F_1^n}(y|f_1^n)$ via Bayes rule
  $P_{Y|F_1^n}(y|f_1^n) \propto P_Y(y)P_{F_1^n|Y}(f_1^n|y)$
- but isn't conditional independence a bit too strong?

Consider the following example

- *Some parts are a bit slow and a little repetitive, but overall not too bad.*

It's riddled with generally negative words, but in the end the overall opinion is positive

Problem with Naive Bayes classification

Instead of computing $P_{Y|X}(y|x)$

- we compute $P_{Y|F_1^n}(y|f_1^n)$ using some features f_1^n of x
- instead of modelling $P_{Y|F_1^n}$ directly, we modelled $P_{F_1^n|Y}$ using tabular CPDs
- and got to $P_{Y|F_1^n}(y|f_1^n)$ via Bayes rule
  $P_{Y|F_1^n}(y|f_1^n) \propto P_Y(y)P_{F_1^n|Y}(f_1^n|y)$
- but isn't conditional independence a bit too strong?

Consider the following example

- *Some parts are a bit slow and a little repetitive, but overall not too bad.*

It's riddled with generally negative words, but in the end the overall opinion is positive

- we need richer features, such as
  a bit slow, *a little repetitive*, *not too bad*

Problem with Naive Bayes classification

Instead of computing $P_{Y|X}(y|x)$

- we compute $P_{Y|F_1^n}(y|f_1^n)$ using some features f_1^n of x
- instead of modelling $P_{Y|F_1^n}$ directly, we modelled $P_{F_1^n|Y}$ using tabular CPDs
- and got to $P_{Y|F_1^n}(y|f_1^n)$ via Bayes rule
  $P_{Y|F_1^n}(y|f_1^n) \propto P_Y(y)P_{F_1^n|Y}(f_1^n|y)$
- but isn't conditional independence a bit too strong?

Consider the following example

- *Some parts are a bit slow and a little repetitive, but overall not too bad.*

It's riddled with generally negative words, but in the end the overall opinion is positive

- we need richer features, such as
  a bit slow, *a little repetitive*, *not too bad*
- but an increase in feature space, e.g. $O(v^3)$ for trigram features, leads to problems for parameter estimation

Conditioning on high-dimensional data

The problem is that we only know tabular CPDs

If Y takes on values in $\mathcal{Y}$ and X takes on values in $\mathcal{X}$, tabular CPDs associate a parameter $\theta_y^{(x)}$ with each outcome y in context x

$$P_{Y|X}(y|x) = \mathrm{Cat}(y|\theta^{(x)}) = \theta_y^{(x)}$$

This can only work if $|\mathcal{Y}|$ and $|\mathcal{X}|$ are relatively small

- representation cost $O(|\mathcal{Y}| \times |\mathcal{X}|)$

If x is itself very high-dimensional (e.g. a sentence), this cannot possibly work (as in this case $\mathcal{X} \subseteq \Sigma^*$)

Are there other representations to CPDs?

Let's focus on the case where x is high-dimensional and y is binary
How can we assign a value to $P_{Y|X}(y|x)$?

- can we avoid a table lookup?
- can we let outcomes share statistical evidence?

Are there other representations to CPDs?

Let's focus on the case where x is high-dimensional and y is binary
How can we assign a value to $P_{Y|X}(y|x)$?

- can we avoid a table lookup?
- can we let outcomes share statistical evidence?

Let's model the probability value!

Are there other representations to CPDs?

Let's focus on the case where x is high-dimensional and y is binary
How can we assign a value to $P_{Y|X}(y|x)$?

- can we avoid a table lookup?
- can we let outcomes share statistical evidence?

Let's model the probability value!

- suppose a function $f(x, y)$ makes a finite summary of the aspects of the joint outcome (x, y) relevant to a problem

Are there other representations to CPDs?

Let's focus on the case where x is high-dimensional and y is binary
How can we assign a value to $P_{Y|X}(y|x)$?

- can we avoid a table lookup?
- can we let outcomes share statistical evidence?

Let's model the probability value!

- suppose a function $f(x, y)$ makes a finite summary of the aspects of the joint outcome (x, y) relevant to a problem
- we call $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$ a *feature function*

Are there other representations to CPDs?

Let's focus on the case where x is high-dimensional and y is binary
How can we assign a value to $P_{Y|X}(y|x)$?

- can we avoid a table lookup?
- can we let outcomes share statistical evidence?

Let's model the probability value!

- suppose a function $f(x, y)$ makes a finite summary of the aspects of the joint outcome (x, y) relevant to a problem
- we call $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$ a *feature function*
- we can then make the probability value $P_{Y|X}(y|x)$ depend functionally on $f(x, y)$

Are there other representations to CPDs?

Let's focus on the case where x is high-dimensional and y is binary
How can we assign a value to $P_{Y|X}(y|x)$?

- can we avoid a table lookup?
- can we let outcomes share statistical evidence?

Let's model the probability value!

- suppose a function $f(x, y)$ makes a finite summary of the aspects of the joint outcome (x, y) relevant to a problem
- we call $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$ a *feature function*
- we can then make the probability value $P_{Y|X}(y|x)$ depend functionally on $f(x, y)$
- but we need to make sure that $0 \leq P_{Y|X}(y|x) \leq 1$ and that $\sum_y P_{Y|X}(y|x) = 1$

Feature function

An example of a binary feature function

Y X	1 This film is fun and full of action	0 This film is full of boring action
action_+	1	0
action_-	0	1
boring_+	0	0
boring_-	0	1
full_+	1	0
full_-	0	1
fun_+	1	0
fun_-	0	0

Table: Feature function: $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}^D$

- binary feature functions map the input to a D-dimensional vector of feature indicators

Logistic regression

Suppose we have a D-dimensional real vector $w \in \mathbb{R}^D$

Logistic regression

Suppose we have a D-dimensional real vector $w \in \mathbb{R}^D$

- Can we say $P_{Y|X}(y|x) = w^\top f(x, y) = \sum_{d=1}^{D} w_d f_d(x, y)$?

Logistic regression

Suppose we have a D-dimensional real vector $w \in \mathbb{R}^D$

- Can we say $P_{Y|X}(y|x) = w^\top f(x, y) = \sum_{d=1}^{D} w_d f_d(x, y)$?
  No! Because $w^\top f(x, y)$ can be negative and $w^\top f(x, y = 1) + w^\top f(x, y = 0)$ may not sum to 1

Logistic regression

Suppose we have a D-dimensional real vector $w \in \mathbb{R}^D$

- Can we say $P_{Y|X}(y|x) = w^\top f(x, y) = \sum_{d=1}^{D} w_d f_d(x, y)$?
  No! Because $w^\top f(x, y)$ can be negative and
  $w^\top f(x, y = 1) + w^\top f(x, y = 0)$ may not sum to 1
- What if we make $P_{Y|X}(y|x) = \exp\Big(w^\top f(x, y)\Big)$?

Logistic regression

Suppose we have a D-dimensional real vector $w \in \mathbb{R}^D$

- Can we say $P_{Y|X}(y|x) = w^\top f(x, y) = \sum_{d=1}^{D} w_d f_d(x, y)$?
  No! Because $w^\top f(x, y)$ can be negative and $w^\top f(x, y = 1) + w^\top f(x, y = 0)$ may not sum to 1
- What if we make $P_{Y|X}(y|x) = \exp\Big(w^\top f(x, y)\Big)$?
  Now we managed to ensure positivity, but still $w^\top f(x, y = 1) + w^\top f(x, y = 0)$ may not sum to 1

Logistic regression

Suppose we have a D-dimensional real vector $w \in \mathbb{R}^D$

- Can we say $P_{Y|X}(y|x) = w^\top f(x, y) = \sum_{d=1}^{D} w_d f_d(x, y)$?
  No! Because $w^\top f(x, y)$ can be negative and $w^\top f(x, y = 1) + w^\top f(x, y = 0)$ may not sum to 1
- What if we make $P_{Y|X}(y|x) = \exp\left(w^\top f(x, y)\right)$?
  Now we managed to ensure positivity, but still $w^\top f(x, y = 1) + w^\top f(x, y = 0)$ may not sum to 1
- The functional dependency on $f(x, y)$ needs to be such that the result is a valid distribution, i.e. normalised across outcomes of $Y|X = x$

Logistic regression

Suppose we have a D-dimensional real vector $w \in \mathbb{R}^D$

- Can we say $P_{Y|X}(y|x) = w^\top f(x, y) = \sum_{d=1}^{D} w_d f_d(x, y)$?
  No! Because $w^\top f(x, y)$ can be negative and $w^\top f(x, y = 1) + w^\top f(x, y = 0)$ may not sum to 1
- What if we make $P_{Y|X}(y|x) = \exp\Big(w^\top f(x, y)\Big)$?
  Now we managed to ensure positivity, but still $w^\top f(x, y = 1) + w^\top f(x, y = 0)$ may not sum to 1
- The functional dependency on $f(x, y)$ needs to be such that the result is a valid distribution, i.e. normalised across outcomes of $Y|X = x$

$$P_{Y|X}(y|x) = \frac{\exp\Big(w^\top f(x, y)\Big)}{\sum_{y' \in \mathcal{Y}} \exp\Big(w^\top f(x, y')\Big)}$$

Logistic regression - MLE

We model the conditional using logistic regression

$$P_{Y|X}(y|x) = \frac{\exp\Big(w^\top f(x, y)\Big)}{\sum_{y' \in \mathcal{Y}} \exp\Big(w^\top f(x, y')\Big)}$$

Then with a dataset $\mathcal{D} = \{(x^{(k)}, y^{(k)})\}_{k=1}^{N}$ of i.i.d. observations, what's the maximum likelihood estimate for $w \in \mathbb{R}^D$?

Logistic regression - MLE

We model the conditional using logistic regression

$$P_{Y|X}(y|x) = \frac{\exp\Big(w^\top f(x,y)\Big)}{\sum_{y' \in \mathcal{Y}} \exp\Big(w^\top f(x,y')\Big)}$$

Then with a dataset $\mathcal{D} = \{(x^{(k)}, y^{(k)})\}_{k=1}^N$ of i.i.d. observations, what's the maximum likelihood estimate for $w \in \mathbb{R}^D$?

Count and divide won't do ;)

Logistic regression - MLE

We model the conditional using logistic regression

$$P_{Y|X}(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$$

Then with a dataset $\mathcal{D} = \{(x^{(k)}, y^{(k)})\}_{k=1}^N$ of i.i.d. observations, what's the maximum likelihood estimate for $w \in \mathbb{R}^D$?

Count and divide won't do ;)
recall where "count and divide" comes from

- we looked for the solution to $\nabla_w \mathcal{L}(w|\mathcal{D}) = 0$

Logistic regression - MLE

We model the conditional using logistic regression

$$P_{Y|X}(y|x) = \frac{\exp\left(w^\top f(x, y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x, y')\right)}$$

Then with a dataset $\mathcal{D} = \{(x^{(k)}, y^{(k)})\}_{k=1}^{N}$ of i.i.d. observations, what's the maximum likelihood estimate for $w \in \mathbb{R}^D$?

Logistic regression - MLE

We model the conditional using logistic regression

$$P_{Y|X}(y|x) = \frac{\exp\Big(w^\top f(x,y)\Big)}{\sum\limits_{y' \in \mathcal{Y}} \exp\Big(w^\top f(x,y')\Big)}$$

Then with a dataset $\mathcal{D} = \{(x^{(k)}, y^{(k)})\}_{k=1}^N$ of i.i.d. observations, what's the maximum likelihood estimate for $w \in \mathbb{R}^D$?

We look for w that is solution to $\boldsymbol{\nabla}_w \mathcal{L}(w|\mathcal{D}) = 0$ where

$$\mathcal{L}(w|\mathcal{D}) = \sum_{k=1}^N \underbrace{\log P_{Y|X}(y^{(k)}|x^{(k)}, w)}_{\ell(w|x^{(k)}, y^{(k)})}$$

is the log-likelihood function

Let's start with a single training instance

The log-likelihood function gets a contribution $\ell(w|x, y) = \log P_{Y|X}(y|x, w)$ from each training instance

Let's expand ℓ slightly

$$\log P_{Y|X}(y|x, w) = \log \frac{\exp\Big(w^\top f(x, y)\Big)}{\sum_{y' \in \mathcal{Y}} \exp(w^\top f(x, y'))}$$

$$=$$

Let's start with a single training instance

The log-likelihood function gets a contribution $\ell(w|x,y) = \log P_{Y|X}(y|x,w)$ from each training instance

Let's expand ℓ slightly

$$\begin{aligned}\log P_{Y|X}(y|x,w) &= \log \frac{\exp\Big(w^\top f(x,y)\Big)}{\sum_{y' \in \mathcal{Y}} \exp(w^\top f(x,y'))} \\ &= w^\top f(x,y) - \log \underbrace{\sum_{y' \in \mathcal{Y}} \exp\Big(w^\top f(x,y')\Big)}_{\mathcal{Z}(x|w)}\end{aligned}$$

We need $\boldsymbol{\nabla}_w \log P_{Y|X}(y|x,w)$ but let's first take the gradient of the partition function $\mathcal{Z}(x|w)$

Gradient of partition function

Let $\mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x, y)\Big)$,

Gradient of partition function

Let $\mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x, y)\Big)$, its gradient is

$$\boldsymbol{\nabla}_w \mathcal{Z}(x|w) = \boldsymbol{\nabla}_w \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x, y)\Big)$$

Gradient of partition function

Let $\mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x, y)\Big)$, its gradient is

$$\begin{aligned}
\boldsymbol{\nabla}_w \mathcal{Z}(x|w) &= \boldsymbol{\nabla}_w \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x, y)\Big) \\
&= \sum_{y \in \mathcal{Y}} \boldsymbol{\nabla}_w \exp\Big(w^\top f(x, y)\Big)
\end{aligned}$$

Gradient of partition function

Let $\mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\left(w^\top f(x, y)\right)$, its gradient is

$$\begin{aligned}
\boldsymbol{\nabla}_w \mathcal{Z}(x|w) &= \boldsymbol{\nabla}_w \sum_{y \in \mathcal{Y}} \exp\left(w^\top f(x, y)\right) \\
&= \sum_{y \in \mathcal{Y}} \boldsymbol{\nabla}_w \exp\left(w^\top f(x, y)\right) \\
&= \sum_{y \in \mathcal{Y}} \exp\left(w^\top f(x, y)\right) \boldsymbol{\nabla}_w (w^\top f(x, y))
\end{aligned}$$

Gradient of partition function

Let $\mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\left(w^\top f(x,y)\right)$, its gradient is

$$\begin{aligned}
\nabla_w \mathcal{Z}(x|w) &= \nabla_w \sum_{y \in \mathcal{Y}} \exp\left(w^\top f(x,y)\right) \\
&= \sum_{y \in \mathcal{Y}} \nabla_w \exp\left(w^\top f(x,y)\right) \\
&= \sum_{y \in \mathcal{Y}} \exp\left(w^\top f(x,y)\right) \nabla_w (w^\top f(x,y)) \\
&= \sum_{y \in \mathcal{Y}} \exp\left(w^\top f(x,y)\right) f(x,y)
\end{aligned}$$

Gradient of the model

Now we know that $\boldsymbol{\nabla}_w \mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\left(w^\top f(x, y)\right) f(x, y)$

Gradient of the model

Now we know that $\nabla_w \mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\left(w^\top f(x,y)\right) f(x,y)$

And $\ell(w|x,y) = w^\top f(x,y) - \log \mathcal{Z}(x|w)$, thus its gradient is

$$\nabla_w \ell(w|x,y) =$$

Gradient of the model

Now we know that $\boldsymbol{\nabla}_w \mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y)$

And $\ell(w|x,y) = w^\top f(x,y) - \log \mathcal{Z}(x|w)$, thus its gradient is

$$\boldsymbol{\nabla}_w \ell(w|x,y) = \boldsymbol{\nabla}_w (w^\top f(x,y) - \log \mathcal{Z}(x|w))$$

Gradient of the model

Now we know that $\boldsymbol{\nabla}_w \mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y)$

And $\ell(w|x,y) = w^\top f(x,y) - \log \mathcal{Z}(x|w)$, thus its gradient is

$$\begin{aligned}\boldsymbol{\nabla}_w \ell(w|x,y) &= \boldsymbol{\nabla}_w (w^\top f(x,y) - \log \mathcal{Z}(x|w)) \\ &= f(x,y) - \boldsymbol{\nabla}_w \log \mathcal{Z}(x|w)\end{aligned}$$

Gradient of the model

Now we know that $\nabla_w \mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\left(w^\top f(x,y)\right) f(x,y)$

And $\ell(w|x,y) = w^\top f(x,y) - \log \mathcal{Z}(x|w)$, thus its gradient is

$$\begin{aligned}
\nabla_w \ell(w|x,y) &= \nabla_w (w^\top f(x,y) - \log \mathcal{Z}(x|w)) \\
&= f(x,y) - \nabla_w \log \mathcal{Z}(x|w)
\end{aligned}$$

Let's check the second term

$\nabla_w \log \mathcal{Z}(x|w) =$

Gradient of the model

Now we know that $\boldsymbol{\nabla}_w \mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y)$

And $\ell(w|x,y) = w^\top f(x,y) - \log \mathcal{Z}(x|w)$, thus its gradient is

$$\begin{aligned}\boldsymbol{\nabla}_w \ell(w|x,y) &= \boldsymbol{\nabla}_w (w^\top f(x,y) - \log \mathcal{Z}(x|w)) \\ &= f(x,y) - \boldsymbol{\nabla}_w \log \mathcal{Z}(x|w)\end{aligned}$$

Let's check the second term

$$\boldsymbol{\nabla}_w \log \mathcal{Z}(x|w) = \frac{1}{\mathcal{Z}(x|w)} \boldsymbol{\nabla}_w \mathcal{Z}(x|w)$$

Gradient of the model

Now we know that $\nabla_w \mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y)$

And $\ell(w|x,y) = w^\top f(x,y) - \log \mathcal{Z}(x|w)$, thus its gradient is

$$\begin{aligned}\nabla_w \ell(w|x,y) &= \nabla_w (w^\top f(x,y) - \log \mathcal{Z}(x|w)) \\ &= f(x,y) - \nabla_w \log \mathcal{Z}(x|w)\end{aligned}$$

Let's check the second term

$$\begin{aligned}\nabla_w \log \mathcal{Z}(x|w) &= \frac{1}{\mathcal{Z}(x|w)} \nabla_w \mathcal{Z}(x|w) \\ &= \frac{1}{\mathcal{Z}(x|w)} \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y)\end{aligned}$$

Gradient of the model

Now we know that $\nabla_w \mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y)$

And $\ell(w|x,y) = w^\top f(x,y) - \log \mathcal{Z}(x|w)$, thus its gradient is

$$\begin{aligned}\nabla_w \ell(w|x,y) &= \nabla_w (w^\top f(x,y) - \log \mathcal{Z}(x|w)) \\ &= f(x,y) - \nabla_w \log \mathcal{Z}(x|w)\end{aligned}$$

Let's check the second term

$$\begin{aligned}\nabla_w \log \mathcal{Z}(x|w) &= \frac{1}{\mathcal{Z}(x|w)} \nabla_w \mathcal{Z}(x|w) \\ &= \frac{1}{\mathcal{Z}(x|w)} \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y) \\ &= \sum_{y \in \mathcal{Y}} \frac{\exp\Big(w^\top f(x,y)\Big)}{\mathcal{Z}(x|w)} f(x,y)\end{aligned}$$

Gradient of the model

Now we know that $\nabla_w \mathcal{Z}(x|w) = \sum_{y\in\mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y)$

And $\ell(w|x,y) = w^\top f(x,y) - \log \mathcal{Z}(x|w)$, thus its gradient is

$$\begin{aligned}\nabla_w \ell(w|x,y) &= \nabla_w (w^\top f(x,y) - \log \mathcal{Z}(x|w)) \\ &= f(x,y) - \nabla_w \log \mathcal{Z}(x|w)\end{aligned}$$

Let's check the second term

$$\begin{aligned}\nabla_w \log \mathcal{Z}(x|w) &= \frac{1}{\mathcal{Z}(x|w)} \nabla_w \mathcal{Z}(x|w) \\ &= \frac{1}{\mathcal{Z}(x|w)} \sum_{y\in\mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y) \\ &= \sum_{y\in\mathcal{Y}} \frac{\exp\Big(w^\top f(x,y)\Big)}{\mathcal{Z}(x|w)} f(x,y) \\ &= \sum_{y\in\mathcal{Y}} P_{Y|X}(y|x,w) f(x,y)\end{aligned}$$

Gradient of the model

Now we know that $\nabla_w \mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y)$

And $\ell(w|x,y) = w^\top f(x,y) - \log \mathcal{Z}(x|w)$, thus its gradient is

$$\begin{aligned} \nabla_w \ell(w|x,y) &= \nabla_w (w^\top f(x,y) - \log \mathcal{Z}(x|w)) \\ &= f(x,y) - \nabla_w \log \mathcal{Z}(x|w) \end{aligned}$$

Let's check the second term

$$\begin{aligned} \nabla_w \log \mathcal{Z}(x|w) &= \frac{1}{\mathcal{Z}(x|w)} \nabla_w \mathcal{Z}(x|w) \\ &= \frac{1}{\mathcal{Z}(x|w)} \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y) \\ &= \sum_{y \in \mathcal{Y}} \frac{\exp\Big(w^\top f(x,y)\Big)}{\mathcal{Z}(x|w)} f(x,y) \\ &= \sum_{y \in \mathcal{Y}} P_{Y|X}(y|x,w) f(x,y) = \mathbb{E}[f(X,Y)|X=x; w] \end{aligned}$$

Putting everything together

We know

- $\nabla_w \mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\left(w^\top f(x,y)\right) f(x,y)$
- $\ell(w|x,y) = w^\top f(x,y) - \log \mathcal{Z}(x|w)$
- $\nabla_w \ell(w|x,y) = f(x,y) - \nabla_w \log \mathcal{Z}(x|w)$
- and $\nabla_w \log \mathcal{Z}(x|w) = \mathbb{E}[f(X,Y)|X=x;w]$

Gradient descent

Putting everything together

We know

- $\nabla_w \mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y)$
- $\ell(w|x,y) = w^\top f(x,y) - \log \mathcal{Z}(x|w)$
- $\nabla_w \ell(w|x,y) = f(x,y) - \nabla_w \log \mathcal{Z}(x|w)$
- and $\nabla_w \log \mathcal{Z}(x|w) = \mathbb{E}[f(X,Y)|X=x;w]$

Then

$$\nabla_w \ell(w|x,y) = f(x,y) - \mathbb{E}[f(X,Y)|X=x;w]$$

Gradient descent

Putting everything together

We know

- $\nabla_w \mathcal{Z}(x|w) = \sum_{y \in \mathcal{Y}} \exp\Big(w^\top f(x,y)\Big) f(x,y)$
- $\ell(w|x,y) = w^\top f(x,y) - \log \mathcal{Z}(x|w)$
- $\nabla_w \ell(w|x,y) = f(x,y) - \nabla_w \log \mathcal{Z}(x|w)$
- and $\nabla_w \log \mathcal{Z}(x|w) = \mathbb{E}[f(X,Y)|X=x;w]$

Then

$$\nabla_w \ell(w|x,y) = f(x,y) - \mathbb{E}[f(X,Y)|X=x;w]$$

There is no closed-form solution to $\nabla_w \ell(w|x,y) = 0$, but there is an iterative algorithm that converges to the solution

$$w^{(t+1)} = w^{(t)} + \gamma \nabla_{w^{(t)}} \ell(w^{(t)}|x,y)$$

$\gamma > 0$ is called the *learning rate* (a hyperparameter)

Gradient descent

Maximum likelihood estimation for logistic regression

We look for w that is solution to $\nabla_w \mathcal{L}(w|\mathcal{D}) = 0$ where

$$\mathcal{L}(w|\mathcal{D}) = \sum_{k=1}^{N} \underbrace{\log P_{Y|X}(y^{(k)}|x^{(k)}, w)}_{\ell(w|x^{(k)}, y^{(k)})}$$

There is no closed-form solution $\nabla_w \mathcal{L}(w|\mathcal{D})$, but there is an iterative algorithm that converges to the solution

$$w^{(t+1)} = w^{(t)} + \gamma \underbrace{\sum_{k=1}^{N} \nabla_{w^{(t)}} \ell(w^{(t)}|x^{(k)}, y^{(k)})}_{\nabla_w \mathcal{L}(w|\mathcal{D})}$$

Stochastic gradient ascent

We can use unbiased *stochastic gradient estimates* instead of the full gradient

$$w^{(t+1)} = w^{(t)} + \gamma^{(t)} \frac{M}{N} \sum_{s=1}^{M} \boldsymbol{\nabla}_{w^{(t)}} \ell(w^{(t)} | x^{(s)}, y^{(s)})$$

where $S \sim \mathcal{U}(1/N)$ selects training instances uniformly at random

Stochastic gradient ascent

We can use unbiased *stochastic gradient estimates* instead of the full gradient

$$w^{(t+1)} = w^{(t)} + \gamma^{(t)} \frac{M}{N} \sum_{s=1}^{M} \boldsymbol{\nabla}_{w^{(t)}} \ell(w^{(t)} | x^{(s)}, y^{(s)})$$

where $S \sim \mathcal{U}(1/N)$ selects training instances uniformly at random
The learning rate $\gamma > 0$ must follow a particular schedule, e.g.

$$\gamma^{(t)} = \frac{\gamma^{(t)}}{1 + \gamma^{(0)} \alpha t}$$

where the initial learning rate $\gamma^{(0)} > 0$ and the rate of decay $\alpha > 0$ are hyperparameters

Regularisation

To avoid overfitting to training instances, we place a penalty on awkwardly large weights, our objective becomes

$$\operatorname*{argmax}_{w \in \mathbb{R}^D} \mathcal{L}(w|\mathcal{D}) - \frac{\lambda}{2}||w||^2$$

where λ is the weight of the L_2 regulariser

Our gradient becomes

$$\begin{aligned}
\boldsymbol{\nabla}_w \left(\mathcal{L}(w|\mathcal{D}) - \lambda||w||^2\right) &= \boldsymbol{\nabla}_w \mathcal{L}(w|\mathcal{D}) - \frac{\lambda}{2}\boldsymbol{\nabla}_w \sum_{d=1}^{D} w_d^2 \\
&= \boldsymbol{\nabla}_w \mathcal{L}(w|\mathcal{D}) - \lambda \sum_{d=1}^{D} w_d
\end{aligned}$$

Regularisation

To avoid overfitting to training instances, we place a penalty on awkwardly large weights, our objective becomes

$$\underset{w \in \mathbb{R}^D}{\text{argmax}} \;\; \mathcal{L}(w|\mathcal{D}) - \frac{\lambda}{2}||w||^2$$

where λ is the weight of the L_2 regulariser

Our gradient becomes

$$\begin{aligned} \boldsymbol{\nabla}_w \left(\mathcal{L}(w|\mathcal{D}) - \lambda||w||^2 \right) &= \boldsymbol{\nabla}_w \mathcal{L}(w|\mathcal{D}) - \frac{\lambda}{2} \boldsymbol{\nabla}_w \sum_{d=1}^{D} w_d^2 \\ &= \boldsymbol{\nabla}_w \mathcal{L}(w|\mathcal{D}) - \lambda \sum_{d=1}^{D} w_d \end{aligned}$$

Note that regularisation leads to *dense* gradients!

Summary

Logistic regression allows us to express statistical dependencies between two variables through a finite set of features

- we can directly model a conditional probability using rich features of a high-dimensional conditioning context (this is called a *logistic cpd*)
- without the need for strong independence assumptions
- we have to estimate D parameters (the weights of a log-linear model)
- MLE does not have a closed-form solution, but gradient ascent gives us an iterative algorithm

Next class we will see how this can be used for various tasks
e.g. sentiment classification, language identification,
POS tagging, language modelling

References I