Natural Language Models and Interfaces

BSc Artificial Intelligence

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

2020, week 5, lecture b

Text Classifiers

We have learnt two techniques to design feature-rich models

- naive Bayes classification
- logistic regression

Text Classifiers

We have learnt two techniques to design feature-rich models

- naive Bayes classification
- logistic regression

They are both useful to condition on high-dimensional data

- NBC uses Bayes rule and a conditional independence assumption
- LR uses a linear model and the softmax function

Pros and Cons

NBC

Pros and Cons

NBC

- exact MLE solution
- strong independence assumption

LR

Pros and Cons

NBC

- exact MLE solution
- strong independence assumption

LR

- flexible
- no closed-form MLE
- but gradient ascent converges to global optimum because the log-likelihood function is concave

Applications

1. Text classifiers: predict a categorical target from high-dimensional input
2. Component in a generative model: parameterise a cpd that conditions on high-dimensional input

Text Classification

Task	Description
Sentiment analysis	emotion towards a subject x is a document and y is a binary label
Textual entailment	given a two pieces of text, does one entail or contradict the other? x is a pair of documents and y is a binary label

Text Classification

Task	Description
Sentiment analysis	emotion towards a subject x is a document and y is a binary label
Textual entailment	given a two pieces of text, does one entail or contradict the other? x is a pair of documents and y is a binary label

In both cases, there are extensions to multiple classes

- Stanford sentiment classification: 5 sentiment levels
- Stanford natural language inference: 3 logical entailment relations

Feature functions

A feature function in NBC is a little different from a feature function in LR

- in NBC only the input x is available to the feature function
- in LR both x and y are available

Why?

Feature functions

A feature function in NBC is a little different from a feature function in LR

- in NBC only the input x is available to the feature function
- in LR both x and y are available

Why?

- NBC:

$$P_{Y|X}(y|x) \stackrel{\text{def}}{=} P_{Y|F_1^n}(y|f_1^n = h(x))$$

Feature functions

A feature function in NBC is a little different from a feature function in LR

- in NBC only the input x is available to the feature function
- in LR both x and y are available

Why?

- NBC:

$$P_{Y|X}(y|x) \stackrel{\text{def}}{=} P_{Y|F_1^n}(y|f_1^n = h(x)) \propto P_Y(y) \prod_{i=1}^{n} P_{F|Y}(f_i|y)$$

Feature functions

A feature function in NBC is a little different from a feature function in LR

- in NBC only the input x is available to the feature function
- in LR both x and y are available

Why?

- NBC:

$$P_{Y|X}(y|x) \overset{\text{def}}{=} P_{Y|F_1^n}(y|f_1^n = \color{red}{h(x)}) \propto P_Y(y) \prod_{i=1}^{n} P_{F|Y}(f_i|y)$$

- LR:

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

Feature functions (cont.)

In NBC a feature is a random variable
while in LR a feature is just input to a log-linear model

Example: NBC for sentiment classification

Consider a sentence like

I did not like the acting, but the plot was decent

Example features are:

- unigrams: I, did, not, like, the, acting, but, the plot, was, decent

---

See Lab5

Example: NBC for sentiment classification

Consider a sentence like

I did not like the acting, but the plot was decent

Example features are:

- unigrams: I, did, not, like, the, acting, but, the plot, was, decent
- we can apply certain filters: frequency based, stopwords

See Lab5

Example: NBC for sentiment classification

Consider a sentence like

I did not like the acting, but the plot was decent

Example features are:

- unigrams: I, did, not, like, the, acting, but, the plot, was, decent
- we can apply certain filters: frequency based, stopwords
- we can distinguish some words (e.g. sentiment words): like, decent

See Lab5

Example: NBC for sentiment classification

Consider a sentence like

I did not like the acting, but the plot was decent

Example features are:

- unigrams: I, did, not, like, the, acting, but, the plot, was, decent
- we can apply certain filters: frequency based, stopwords
- we can distinguish some words (e.g. sentiment words): like, decent
- we can use external resources (e.g. POS tagger)

See Lab5

Example: NBC for sentiment classification

Consider a sentence like

I did not like the acting, but the plot was decent

Example features are:

- unigrams: I, did, not, like, the, acting, but, the plot, was, decent
- we can apply certain filters: frequency based, stopwords
- we can distinguish some words (e.g. sentiment words): like, decent
- we can use external resources (e.g. POS tagger)

We can pre-process the data to account for negation scope

I did not like-NEG the-NEG acting-NEG, but the plot was decent

See Lab5

Example: LR for sentiment classification

Consider a sentence like

I did not like the acting, but the plot was decent

We can consider the same types of features and more

- we can pair them with the class: like_{+}, like_{-}

Example: LR for sentiment classification

Consider a sentence like

I did not like the acting, but the plot was decent

We can consider the same types of features and more

- we can pair them with the class: like_{+}, like_{-}
- we can use overlapping features: not like, like acting, plot decent

Example: LR for sentiment classification

Consider a sentence like

I did not like the acting, but the plot was decent

We can consider the same types of features and more

- we can pair them with the class: like_{+}, like_{-}
- we can use overlapping features: not like, like acting, plot decent
- with good regularisation, we can have far more features

Evaluation

For binary classification

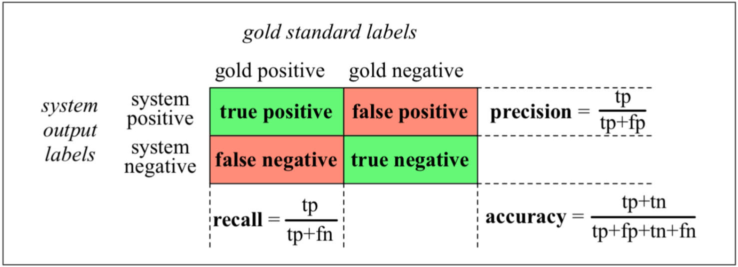

Figure 4.4 Contingency table

- use a training/development/test split
- or, preferably, cross-validation

Image from Ch4 of Jurafsky and Martin (3rd edition)

Component in Generative Model

Recall that we modified the HMM by combining it with a bigram LM?

Component in Generative Model

Recall that we modified the HMM by combining it with a bigram LM?

$$P_{X_1^n C_1^n|N}(x_1^n, c_1^n|n) = \prod_{i=1}^{n} P_{C|C_{\text{prev}}}(c_i|c_{i-1}) P_{X|X_{\text{prev}}C}(x_i|x_{i-1}, c_i)$$

Component in Generative Model

Recall that we modified the HMM by combining it with a bigram LM?

$$P_{X_1^n C_1^n|N}(x_1^n, c_1^n|n) = \prod_{i=1}^{n} P_{C|C_{\text{prev}}}(c_i|c_{i-1}) P_{X|X_{\text{prev}}C}(x_i|x_{i-1}, c_i)$$

Note that we modified emission probabilities to be

$$P_{X|X_{\text{prev}}C}(x|x', c)$$

Component in Generative Model

Recall that we modified the HMM by combining it with a bigram LM?

$$P_{X_1^n C_1^n|N}(x_1^n, c_1^n|n) = \prod_{i=1}^{n} P_{C|C_{\text{prev}}}(c_i|c_{i-1}) P_{X|X_{\text{prev}}C}(x_i|x_{i-1}, c_i)$$

Note that we modified emission probabilities to be

$$P_{X|X_{\text{prev}}C}(x|x', c)$$

Interpolated CPD

This is a heuristic technique whereby we use a convex combination of simpler CPDs:

$$P_{X|X_{\text{prev}}C}(x|x', c) = \alpha \times P_{X|X_{\text{prev}}}(x|x') + (1 - \alpha) \times P_{X|C}(x|c)$$

Interpolated CPD

This is a heuristic technique whereby we use a convex combination of simpler CPDs:

$$P_{X|X_{\text{prev}}C}(x|x', c) = \alpha \times P_{X|X_{\text{prev}}}(x|x') + (1 - \alpha) \times P_{X|C}(x|c)$$

- it requires $0 < \alpha < 1$ for which no closed-form MLE is available
- we need to tune α on held-out data
- and estimate the simpler cpds independently $P_{X|X_{\text{prev}}}(x|x') = \frac{\text{count}_{X_{\text{prev}}X}(x',x)}{\text{count}_{X_{\text{prev}}}(x')}$ and $P_{X|C}(x|c) = \frac{\text{count}_{CX}(c,x)}{\text{count}_{C}(c)}$

Naive Bayes CPD

We can use the Naive Bayes assumption to model CPDs!

$$P_{X|X_{\text{prev}}C}(x|x', c) =$$

Naive Bayes CPD

We can use the Naive Bayes assumption to model CPDs!

$$P_{X|X_{\mathsf{prev}}C}(x|x', c) = \frac{P_X(x)P_{X_{\mathsf{prev}}C|x}(x', c|x)}{P_{X_{\mathsf{prev}}C}(x', c)}$$

Naive Bayes CPD

We can use the Naive Bayes assumption to model CPDs!

$$P_{X|X_{\textsf{prev}}C}(x|x', c) = \frac{P_X(x)P_{X_{\textsf{prev}}C|x}(x', c|x)}{P_{X_{\textsf{prev}}C}(x', c)}$$

$$\overset{\textsf{ind}}{=} \frac{P_X(x)P_{X_{\textsf{prev}}|x}(x'|x)P_{C|X}(c|x)}{P_{X_{\textsf{prev}}C}(x', c)}$$

Naive Bayes CPD

We can use the Naive Bayes assumption to model CPDs!

$$P_{X|X_{\text{prev}}C}(x|x', c) = \frac{P_X(x)P_{X_{\text{prev}}C|x}(x', c|x)}{P_{X_{\text{prev}}C}(x', c)}$$

$$\overset{\text{ind}}{=} \frac{P_X(x)P_{X_{\text{prev}}|x}(x'|x)P_{C|X}(c|x)}{P_{X_{\text{prev}}C}(x', c)}$$

Note the denominator needs to be inferred

$$P_{X_{\text{prev}}C}(x', c) = \sum_{x \in \mathcal{X}} P_{X_{\text{prev}}CX}(x', c, x)$$

Naive Bayes CPD

We can use the Naive Bayes assumption to model CPDs!

$$P_{X|X_{\text{prev}}C}(x|x', c) = \frac{P_X(x) P_{X_{\text{prev}}C|x}(x', c|x)}{P_{X_{\text{prev}}C}(x', c)}$$

$$\overset{\text{ind}}{=} \frac{P_X(x) P_{X_{\text{prev}}|x}(x'|x) P_{C|X}(c|x)}{P_{X_{\text{prev}}C}(x', c)}$$

Note the denominator needs to be inferred

$$P_{X_{\text{prev}}C}(x', c) = \sum_{x \in \mathcal{X}} P_{X_{\text{prev}}CX}(x', c, x)$$

$$= \sum_{x \in \mathcal{X}} P_X(x) P_{X_{\text{prev}}|X}(x'|x) P_{C|X}(c|x)$$

Naive Bayes CPD

We can use the Naive Bayes assumption to model CPDs!

$$P_{X|X_{\text{prev}}C}(x|x', c) = \frac{P_X(x)P_{X_{\text{prev}}C|x}(x', c|x)}{P_{X_{\text{prev}}C}(x', c)}$$

$$\overset{\text{ind}}{=} \frac{P_X(x)P_{X_{\text{prev}}|x}(x'|x)P_{C|X}(c|x)}{P_{X_{\text{prev}}C}(x', c)}$$

Note the denominator needs to be inferred

$$P_{X_{\text{prev}}C}(x', c) = \sum_{x \in \mathcal{X}} P_{X_{\text{prev}}CX}(x', c, x)$$

$$= \sum_{x \in \mathcal{X}} P_X(x)P_{X_{\text{prev}}|X}(x'|x)P_{C|X}(c|x)$$

Pro: MLE is exact (no need to tune heuristic coefficients)
Con: denominator must be computed $O(|\mathcal{X}|)$ (this scales linearly in vocabulary size)

Logistic CPD

This is a direct application of LR, for example:

$$P_{X|X_{\text{prev}}C}(x|x', c; w) = \frac{\exp\left(w^\top f(x, x', c)\right)}{\mathcal{Z}(x|w)}$$

$$\mathcal{Z}(x', c|w) = \sum_{x \in \mathcal{X}} \exp\left(w^\top f(x, x', c)\right)$$

Logistic CPD

This is a direct application of LR, for example:

$$P_{X|X_{\text{prev}}C}(x|x', c; w) = \frac{\exp\left(w^\top f(x, x', c)\right)}{\mathcal{Z}(x|w)}$$

$$\mathcal{Z}(x', c|w) = \sum_{x \in \mathcal{X}} \exp\left(w^\top f(x, x', c)\right)$$

Pro: flexible

Con: no closed-form MLE

Logistic CPD

This is a direct application of LR, for example:

$$P_{X|X_{\text{prev}}C}(x|x', c; w) = \frac{\exp\left(w^\top f(x, x', c)\right)}{\mathcal{Z}(x|w)}$$

$$\mathcal{Z}(x', c|w) = \sum_{x \in \mathcal{X}} \exp\left(w^\top f(x, x', c)\right)$$

Pro: flexible

Con: no closed-form MLE

- but we can start with some random $w \in \mathbb{R}^D$ such that $w_d \sim \mathcal{N}(0, 1)$

Logistic CPD

This is a direct application of LR, for example:

$$P_{X|X_{\text{prev}}C}(x|x', c; w) = \frac{\exp\left(w^\top f(x, x', c)\right)}{\mathcal{Z}(x|w)}$$

$$\mathcal{Z}(x', c|w) = \sum_{x \in \mathcal{X}} \exp\left(w^\top f(x, x', c)\right)$$

Pro: flexible

Con: no closed-form MLE

- but we can start with some random $w \in \mathbb{R}^D$ such that $w_d \sim \mathcal{N}(0, 1)$
- compute the log-likelihood function $\mathcal{L}(w|\mathcal{D})$ for a dataset

Logistic CPD

This is a direct application of LR, for example:

$$P_{X|X_{\text{prev}}C}(x|x', c; w) = \frac{\exp\left(w^\top f(x, x', c)\right)}{\mathcal{Z}(x|w)}$$

$$\mathcal{Z}(x', c|w) = \sum_{x \in \mathcal{X}} \exp\left(w^\top f(x, x', c)\right)$$

Pro: flexible

Con: no closed-form MLE

- but we can start with some random $w \in \mathbb{R}^D$ such that $w_d \sim \mathcal{N}(0, 1)$
- compute the log-likelihood function $\mathcal{L}(w|\mathcal{D})$ for a dataset
- and then $w + \boldsymbol{\nabla}_w \mathcal{L}(w|\mathcal{D})$ takes us closer to the optimum

Logistic CPD

This is a direct application of LR, for example:

$$P_{X|X_{\text{prev}}C}(x|x', c; w) = \frac{\exp\left(w^\top f(x, x', c)\right)}{\mathcal{Z}(x|w)}$$

$$\mathcal{Z}(x', c|w) = \sum_{x \in \mathcal{X}} \exp\left(w^\top f(x, x', c)\right)$$

Pro: flexible

Con: no closed-form MLE

- but we can start with some random $w \in \mathbb{R}^D$ such that $w_d \sim \mathcal{N}(0, 1)$
- compute the log-likelihood function $\mathcal{L}(w|\mathcal{D})$ for a dataset
- and then $w + \boldsymbol{\nabla}_w \mathcal{L}(w|\mathcal{D})$ takes us closer to the optimum
- eventually the log-likelihood function stops improving and we have a global optimum

References I