Classifying Text

Nandan Rao

April, 2019

Outline

- Emprical risk minimization and $p(y, x)$
- $p(x)$ when X is language
- Generative vs discriminative classifiers
- Hyperparameters in BOW framework
- Towards supervising the embedded space

Emprical Risk Minimization

Consider an input space X, a discrete output space Y and a function, $g : X \to Y$ which predicts a value for y given an input x. Consider also a *loss function* $\ell : Y, Y \to \mathbb{R}$.

The *risk* of the classifier g is defined as:

$$R(g) = \mathbb{E}_{X,Y}[\ell(g(x), y)]$$

$$R(g) = \sum_{y \in Y} \int_X \ell(g(x), y)\, p(x, y)\, dx$$

Emprical Risk Minimization

We estimate the risk with its finite-sample approximation, the *empirical risk*:

$$R(g) = \frac{1}{n} \sum_{i=1}^{n} \ell(g(x_i), y_i)$$

Note, that this is only an approximation of $\mathbb{E}_{X,Y}$ if all the pairs:

$$x_i, y_i \in p(X, Y)$$

Emprical Risk Minimization

$$x_i, y_i \in p(X, Y)$$

Implies that $p(Y|X)$ and $p(X)$ are the same in our validation distributions as they are in our target distribution.

Ways to deal with changes in any of the component parts of the joint distribution are covered in the literature of *domain adaptation*.

Language

What is $p(X)$ when we are dealing with language?

Again, we will consider that each x_i is a document.

Is the document space continuous?

Hopefully you see why for most tasks related to NLP, we want the space to be continuous.

Thus, in order to classify, we need to embed the documents into a continous space, with a continuous distance metric.

Generative classifiers

Generative classifiers seek to model the join probability $p(X, Y)$ directly.

With a model built of the joint probability, the classification simply consists of applying bayes rule to the joint probability and picking the class $y \in Y$ with the highest $p(y|x)$

Discriminative classifiers

Discriminative classifiers either:

- Directly model the posterior, $p(Y|X)$
- Learn a direct mapping $g : X \rightarrow Y$

Logistic Regression

Logistic regression models the posterior probability as:

$$p(y|x;\beta) = \sigma(\beta^T x)$$

where the logistic function σ is given by:

$$\sigma(n) = \frac{1}{1 + \exp^{-n}}$$

This can be fit via maximum likelihood ($\ell := -p(y|x;\beta)$) or by any other convex surragote of the 0-1 error ($\ell := 1\{g(x) \neq y\}$)

Naive Bayes

As mentioned, Naive Bayes predicts based on the posterior calculated from the modelled joint distribution:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

It's called "naive" because we make the (extreme) simplifying assumption that all the $p(x_i|y)$ are independent and thus $p(x|y) = \prod_i p(x_i|y)$.

Naive Bayes

What does this independence assumption amount to saying about language?

Naive Bayes

Let's see how a multinomial naive Bayes can be related to logistic regression:

$$p(y = 1|x) = \frac{p(x, y = 1)}{\sum_y p(x, y = y_i)}$$

$$p(y = 1|x) = \sigma\left(\log \frac{p(x, y_1)}{p(x, y_0)}\right)$$

$$p(y = 1|x) = \sigma\left((\pi_1 - \pi_0)^T x + \log \frac{p(y_1)}{p(y_0)}\right)$$

Where π_i denotes the log probability parameters of the multinomial distribution representing $p(X|y_i)$

Naive Bayes

Thus the binary multinomial naive Bayes is a linear classifier and, in particular, one in which the coefficients are equal to the log difference in probabilities that a particular feature shows up in each class.

Asymptotic Properties

Logistic regression searches the space of linear classifiers in the feature space, finding the linear classifier with the lowest empirical risk in the training set.

Naive Bayes is a specific linear classifier that makes distributional assumptions over the data generating process $p(X|Y)$.

It should be clear that, asymptotically, logistic regression will converge to the optimal linear classifier. It's not clear that Naive Bayes will do the same, if the distributional assumptions do not hold. Thus:

$$R(g_{LR,\infty}) \leq R(g_{NB,\infty})$$

Asymptotic Properties

However, making more assumptions allows Naive Bayes to reach it's asymptotic risk quicker, as shown by Andrew Ng and Michael Jordan:

https://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf

Let's try this on our data.

Hyperparameter Optimization

A hyperparameter is any parameter that the optimization process of the our learning algorithm does not optimize over.

Thus in logistic regression, the regularization term, λ, is a hyperparameter.

Our training algorithm will pick the parameters that minimize the in-sample empirical risk.

We pick hyperparameters to minimize the out-of-sample empirical risk.

This usually is done either by changing the class of models searched by the algorithm or by changing the loss function to reduce the generalization error.

Hyperparameter Optimization

In our simple language models, however, it should be clear that we have hyperparameters that we want to tune that seem to have nothing to do with our model.

For example, all the parameters in the vectorizing step.

These parameters guide the way that our documents are embedded in the metric space.

We want this embedding to be the best that it can be, such that it minimizes our expected risk.

Hyperparameter Tuning

How can we pick good hyperparameters?

Hyperparameter Tuning

The simplest way to tune hyperparameters is to make no assumptions about them and just search the entire space, hoping to find ones that are relatively good.

This is, interestingly enough, very common!

It's called grid search.

Tuning an embedding

It should be clear, however, that this is not ideal.

Is it possible to jointly optimize the way that we embed language into a metric space AND the classification function at the same time?

Barcelona GSE
Data Science Center