Generative Models: Gaussian Discriminative Analysis and Naïve Bayes

Author: Sami Abu-El-Haija (samihaija@umich.edu)

October 10, 2013

In this document, we briefly review the concept of Generative Models, and review the derivation of GDA and Naïve Bayes.

1 Generative vs Discriminative

Generally, there are two wide classes of Machine Learning models: Generative Models and Discriminative Models. Discriminative models aim to come up with a "good separator". Generative Models aim to estimate densities to the training data. Generative Models assume that the data was *generated* from some probability density. The aim of fitting a generative model is to estimate the probability distribution that the data was generated from.

1.1 Recap: Discriminative Model

Recall Logistic Regression, which is a discriminative model. In Logistic Regression, one maximizes likelihood parameters via Maximum Likelihood Estimate (MLE), of the conditional probability:

$$L(\mathbf{w}) = \prod_{i=1}^{N} p(t^{(i)}|\mathbf{x}^{(i)})$$

The above likelihood can be maximized by optimization methods, such as Gradient Ascent or Newton's method. The learned $\mathbf{w}^* = \arg\max_{\mathbf{w}} L(\mathbf{w})$, [1] is then used for classification. The Logisitc classifier takes a test example (unseen example) and computes $p(t=1|\mathbf{x};\mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x})$ and $p(t=0|\mathbf{x};\mathbf{w}) = 1 - p(t=0|\mathbf{x};\mathbf{w})$, then classifies the example to the class that returned a larger probability measure. Please read the Logistic Regression handout if you need a refresher.

Let's dig deeper into what's happening during classification. The argument of the $\sigma(.)$ is the inner product of $\mathbf{w}^T\mathbf{x}$. This inner product can be *visualized* as a linear separator. For instance, $\mathbf{w}$ can be plotted as a straight line if the features are 2-dimensional ($M = 2$), plotted as a plane if $M = 3$, and plotted as a hyper-plane if $M > 3$. The Logistic Classifier classifies a test example depending on the side of the hyperplane that it lies on.

1.2 Generative Models

In contrast, Generative Models, like GDA, fit probability distributions to the input feature vectors. Their Likelihood is the joint distribution [2]:

$$L(parameters) = \prod_{i}^{N} p(\mathbf{x}^{(i)}, t^{(i)}; parameters)$$

[1]it is common denote with * superscript the value returned by an $\arg\max$

[2]As taught later in EECS445, some generative algorithms are used to learn probability of data in a non-classification setting. i.e., optimizing for $\prod_i p(\mathbf{x}^{(i)})$, where as generative models used for classification generally learn $\prod_i p(\mathbf{x}^{(i)}, t^{(i)})$

It is possible in such models to **generate** examples, that look realistic to the training data. Because Generative models fit probability distributions to the data (rather than learning a separating hyperplane, like discriminative models), it is possible to get samples from the fitted probability distributions (similar to how one can sample a number from gaussian distribution with known parameters). it is not possible in discriminative models to *sample* or *generate* realistic examples. Since, there is no plausible way of generating a realistic example using a separating hyperplane (*a.k.a. decision boundary*).

2 Gaussain Discriminative Analysis (GDA)

GDA uses Bayes rule to represent the joint distribution:

$$p(\mathbf{x}, t) = p(\mathbf{x}|t)p(t)$$

When restricted to a binary classification tasks, GDA models $p(t)$ as a Bernoulli Distribution with parameter ϕ. *Note: this ϕ has nothing to do with the feature mapping function $\phi(.)$. We apologize for mixing notations. Throughout this document, we use $\mathbf{x}$ as a feature vector.* GDA models:

$$\begin{aligned} &p(t = 1) = \phi \\ &p(t = 0) = 1 - p(t = 0) = 1 - \phi \end{aligned}$$

Or, combining the two cases into one:

$$p(t) = \phi^t (1 - \phi)^{1-t}$$

Further, GDA models $p(\mathbf{x}|t = c)$ as a Guassian distribution with parameters μ_c, Σ, known as "mean vector for class c" and "covariance matrix":

$$p(\mathbf{x}|t = c) = \frac{1}{(2\pi)^M |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_c)^T \Sigma^{-1} (\mathbf{x} - \mu_c)\right)$$

In the Binary classification case:

$$\begin{aligned} p(\mathbf{x}|t = 1) &= \frac{1}{(2\pi)^M |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_1)^T \Sigma^{-1} (\mathbf{x} - \mu_1)\right) \\ p(\mathbf{x}|t = 0) &= \frac{1}{(2\pi)^M |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_0)^T \Sigma^{-1} (\mathbf{x} - \mu_0)\right) \end{aligned}$$

Note that every class has its own mean vector but all share a single covariance matrix. $\mu_c \in \mathbb{R}^M$ and $\Sigma \in \mathbb{R}^{M \times M}$, where M is the number of features (dimension of $\mathbf{x}$). In the binary case, which we restrict ourselves to in this document, $c \in \{0, 1\}$. We use the same trick above to combine the probabilities conditioned on $t = 0$ and $t = 1$ to write:

$$p(\mathbf{x}|t) = p(\mathbf{x}|t = 1)^t p(\mathbf{x}|t = 0)^{1-t}$$

Finally, a GDA classifier takes an example $\mathbf{x}$ and assigns the class that maximizes the joint distribution, like:

$$\arg\max_{t \in \{0,1\}} [p(\mathbf{x}, t; \phi, \mu_0, \mu_1, \Sigma)] = \arg\max_{t \in \{0,1\}} [p(t; \phi) p(\mathbf{x}|t; \mu_0, \mu_1, \Sigma)]$$

2.1 Maximum Likelihood Estimates (MLE)

The parameters of the model are $\phi, \mu_0, \mu_1, \Sigma$. Learning a GDA model corresponds to finding the parameters that maximize the likelihood. i.e. solving for:

$$\begin{aligned}\underset{\{\phi,\mu_0,\mu_1,\Sigma\}}{\arg\max}\ L(\phi,\mu_0,\mu_1,\Sigma) &= \prod_{i=1}^{N} p(\mathbf{x}^{(i)}, t^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \\ &= \prod_{i=1}^{N} p(\mathbf{x}^{(i)} | t^{(i)}; \mu_0, \mu_1, \Sigma)\ p(t^{(i)}; \phi)\end{aligned}$$

Which is equivalent to solving for the parameters that maximize the log-likelihood:

$$\underset{\{\phi,\mu_0,\mu_1,\Sigma\}}{\arg\max}\ l(\phi,\mu_0,\mu_1,\Sigma) = \log \prod_{i=1}^{N} p(\mathbf{x}^{(i)} | t^{(i)}; \mu_0, \mu_1, \Sigma)\ p(t^{(i)}; \phi)$$

For each of the parameters $\phi, \mu_0, \mu_1, \Sigma$, it is possible to take the derivative of the log-likelihood with respect to that parameter, set to zero, and solve for the parameter. Giving the maximum likelihood estimates:

$$\begin{aligned}\phi &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{I}\{t^{(i)} = 1\} \\ \mu_0 &= \frac{\sum_{i=1}^{N} \mathbf{I}\{t^{(i)} = 0\}\mathbf{x}^{(i)}}{\sum_{i=1}^{N} \mathbf{I}\{t^{(i)} = 0\}} \\ \mu_1 &= \frac{\sum_{i=1}^{N} \mathbf{I}\{t^{(i)} = 1\}\mathbf{x}^{(i)}}{\sum_{i=1}^{N} \mathbf{I}\{t^{(i)} = 1\}} \\ \Sigma &= \frac{1}{N}\sum_{i=1}^{N} (\mathbf{x}^{(i)} - \mu_{t^{(i)}})(\mathbf{x}^{(i)} - \mu_{t^{(i)}})^T\end{aligned}$$

Note: The Σ is contains within it the computed estimates for μ_0 and μ_1. The parameters can be learned in the listed order.

2.2 Meaning of the MLE parameters

The expressions above are very interpretable. In particular:

- The maximum likelihood $\phi \in \mathbb{R}$ is equal to the ratio of training examples with class 1.
- $\mu_0 \in \mathbb{R}^M$ is equal to the mean (centroid) of the feature vectors that have the label 0.
- $\mu_1 \in \mathbb{R}^M$ is equal to the mean (centroid) of the feature vectors that have the label 1.
- $\Sigma \in \mathbb{R}^{M \times M}$ is the covariance of features, averaged across training data. If features k, l are correlated in the training data, $\Sigma_{kl} = \Sigma_{lk}$ will be large (large and positive if they are positively correlated, large and negative if they are negatively correlated). If they are uncorrelated, $\Sigma_{kl} = \Sigma_{lk} \approx 0$

2.3 Matrix Identities for deriving MLE of GDA

In order to derive the MLE for Σ, it is necessary to use matrix identities that have not been taught (yet) in the course. All the formulas below are given (and proven) in Stanford's Linear Algebra Review Notes for Machine Learning [3].

- Recall the *trace* operator `tr`(A) which takes a square matrix and returns the sum of the elements along the diagonal.

[3] `http://cs229.stanford.edu/section/cs229-linalg.pdf`

If A was a real number (i.e.: $A \in \mathbb{R}^{1\times 1}$ or simply $A \in \mathbb{R}^{1\times 1}$, then $A = \texttt{tr}(A)$). In other words the trace of a real-number the number itself. More relevant to us, if some matrix multiplication $\mathbf{x}^T A\mathbf{x}$ produces a real number, than we can apply the trace operator since:

$$\texttt{tr}(\mathbf{x}^T A\mathbf{x}) = \mathbf{x}^T A\mathbf{x}$$

- In addition, one is allowed to 'rotate' the arguments of a trace:

$$\texttt{tr}(ABC) = \texttt{tr}(CAB) = \texttt{tr}(BCA)$$

- Further, if A is a square matrix, then the matrix derivative:

$$\nabla_A \texttt{tr}(Ab) = b$$

  where b can be a vector or another matrix.

- The derivative of a determinant of some matrix A is given by:

$$\nabla_A |A| = |A| A^{-T}$$

- Chaining the derivative of log and derivative of determinant (please verify this yourself using chain-rule):

$$\nabla_A \log |A| = A^{-1}$$

2.4 GDA example

Will be added to the document later

2.5 GDA or Logistic Regression?

Will be added to the document later

3 Naïve Bayes (NB)

Naïve Bayes is also a generative model. It generally used for text classification, to classify documents into one of K classes. Lets start by summarizing the model parameters:

- $\phi_1, \dots, \phi_K$. One for each class. These are called the *priors*. ϕ_j equals "the probability of any document belonging to class k". The term *prior* in machine learning refers to "prior knowledge" [4]. In general, $\sum_{k=1}^{K} \phi_k = 1$. Therefore, it is common to estimate $K-1$ priors $\phi_1, \dots, \phi_{K-1}$ since we can define $\phi_K = 1 - \sum_{k=1}^{K-1} \phi_k$

- $\mu_j^k \in \mathbb{R}$ for $j \in M, k \in [1, K]$. this means, every word j and class k have a measure μ_j^k, which is equal to the probability of word j appearing in class k

From this point onwards in the document, we will restrict ourselves to binary classification, where the document class is $\in \{0, 1\}$. Therefore, we use $p(t = 1) = \phi$ to denote the prior of the positive class, and the prior for the negative class is implicitly $p(t = 0) = 1 - \phi$. The derivations are easily extensible to K classes.

[4] Most times, priors are measured from the training set. In some cases where one has prior knowledge that spam occurs 20% of the time, it is possible to assign $\phi_{spam} = 0.2$ and $\phi_{notspam} = 0.8$

3.1 Classification in Naïve Bayes

Given an example document ($\mathbf{x} =$ “life is good”), Naïve Bayes Classifier classifies the example into its class by computing $P(\mathbf{x}, t = 0)$ and $P(\mathbf{x}, t = 1)$ then classifying the example to the class with the larger probability measure. Same as GDA, Naïve Bayes doesn't compute this expression explicitly, but decomposes it into two expressions using Bayes Rule:

$$\begin{aligned} p(\mathbf{x}, t = 1) &= p(t = 1)p(\mathbf{x}|t = 1) \\ &= p(t = 1)p(\texttt{word}_1 = \text{“life”}, \texttt{word}_2 = \text{“is”}, \texttt{word}_2 = \text{“good”}|t = 1) \quad \textit{decomposing document into words} \\ &= p(t = 1)p(\texttt{word}_1 = \text{“life”}|t = 1)p(\texttt{word}_2 = \text{“is”}|t = 1)p(\texttt{word}_3 = \text{“good”}|t = 1) \quad \textit{the } \underline{\textit{Naïve}} \textit{ assumption} \\ &= p(t = 1) \prod_{\text{word} j \in \mathbf{x}} p(\texttt{word} = j|t = 1) \\ &= \phi \prod_{\text{word} j \in \mathbf{x}} \mu_j^1 \end{aligned}$$

Going from the second line to the third line is the core of the Naïve Bayes algorithm. It is a very strong assumption (known as the Naïve assumption, giving rise to the name of the model). Consecutive words in language are important. The Naïve Bayes model assumes that consecutive words are all independent from one another. Nonetheless, this over simplified model of the language does reasonably well on some tasks. Similarily,

$$p(\mathbf{x}, t = 0) = (1 - \phi) \prod_{\text{word} j \in \mathbf{x}} \mu_j^0$$

3.2 Event Models for Naïve Bayes

There are two types of Naïve Bayes Models, each has a different interpretation and a different way in modeling and computing μ_j^k

3.2.1 Multiomial Event Model

Here, documents are represented as integer vectors of size M, where M equals to the size of the vocabulary of the English language. Conceretely, $\mathbf{x} \in \mathbb{Z}^M$. The j-th entry of the document vector (x_j) represents the number of times that the j-th word appears in the document.

Here, the maximum likelihood estimate for parameter μ_j^k gets assigned to:

$$\mu_j^k = \frac{\text{the number of times word } j \text{ appears in class} k}{\text{total number of words in class } k}$$

Note: This is the model in the lecture slides and also in the homework

3.2.2 Multivariate Bernoulli Event Model

Here, documents are represented as binary vectors of size M, where M equals to the size of the vocabulary of the English language. Conceretely, $\mathbf{x} \in \{0, 1\}^M$. The j-th entry of the document vector (x_j) is set to 1 if the j-th word appears in the document.

Here, the maximum likelihood estimate for parameter μ_j^k gets assigned to the fraction of documents from class k that contain word j:

$$\mu_j^k = \frac{\text{the number of documents in class } k \text{ containing word } j}{\text{number of documents in class } k}$$

3.3 Maximum Likelihood Estimates

The likelihood of the generative Naïve Bayes classifier is:

$$\begin{aligned} L(\phi, \mu_1^0, \mu_2^0, \dots, \mu_M^0, \mu_1^1, \mu_2^1, \dots, \mu_M^1,) &= \prod_{i=1}^{N} p(\mathbf{x}^{(i)}, t^{(i)}; \phi, \mu_1^0, \mu_2^0, \dots, \mu_M^0, \mu_1^1, \mu_2^1, \dots, \mu_M^1) \\ &= \prod_{i=1}^{N} p(t^{(i)}; \phi) p(\mathbf{x}^{(i)} | t^{(i)}; \mu_1^0, \dots, \mu_M^0, \mu_1^1, \dots, \mu_M^1) \\ &= \prod_{i=1}^{N} \left(\phi \prod_{\text{word} j \in \mathbf{x}} \mu_j^1 \right)^{t^{(i)}} \left((1-\phi) \prod_{\text{word} j \in \mathbf{x}} \mu_j^0 \right)^{1-t^{(i)}} \end{aligned}$$

Taking the derivative of the log-likelihood with respect to the ϕ, setting to zero, and solving for ϕ yields:

$$\phi = \frac{\sum_{i=1}^{N} \mathbf{I}[\mathbf{t^{(i)}} = \mathbf{1}]}{|N|}$$

Similarly, taking derivative with respect to each μ_j^k yield the estimates described in the two previous subsections (depending on which event model is being used)

3.4 Laplace Smoothing

In the current construction of Naïve Bayes, if there was a case that some word (say "homework") was never observed in the spam class during training, then some obvious spam email that contains the word "homework" will be classified as spam with zero probability. This is because $\mu_{\text{homework}}^{\text{spam}} = 0$, and the product of zero times something gives a result of zero. Laplace smoothing removes this problem by adding a "fake document" to both classes that contains every word exactly once. Adding this fake document is equivalent to modifying the MLE estimates μ_j^k (of multinomial event model) to:

$$\mu_j^k = \frac{1 + \text{the number of times word } j \text{ appears in class} k}{M + \text{total number of words in class } k}$$

This removes the bogus assumption that *based on my training set, it is impossible to find the word homework in the spam class*

4 Conclusion

In this document, we briefly introduced the difference between discriminative and generative models. We also discussed the formulation of GDA and Naïve Bayes.