CS229 lecture 2 notes

James Chuang

January 19, 2017

Contents

My notes on Andrew Ng's CS229 lecture 2 notes.

Generative Learning algorithms

discriminative learning algorithms

- learn $p(y \mid x)$ directly
  - e.g. logistic regression
- or, learn mapping from the space of inputs, $\mathcal{X}$ to the labels
  - e.g. perceptron algorithm

generative learning algorithms

- model $p(x \mid y)$ (and $p(y)$, the ***class priors***)
  - e.g. if 0 = dog and 1 = elephant, $p(x \mid y = 0)$ models the distribution of dogs' features, and $p(x \mid y = 1)$ models the distribution of elephants' features
- then use Bayes rule to derive the posterior distribution on y given x:

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

  - to use $p(y \mid x)$ to make a prediction, the denominator $p(x)$ does not need to be calculated:

$$\begin{aligned}\arg\max_y p(y \mid x) &= \arg\max_y \frac{p(x \mid y)p(y)}{p(x)} \\ &= \arg\max_y p(x \mid y)p(y)\end{aligned}$$

Gaussian discriminant analysis

Also known as ***linear discriminant analysis***. Assume that $p(x \mid y)$ is distributed according to a multivariate normal distribution.

The multivariate normal distribution

parameters

- ***mean vector*** $\mu \in \mathbb{R}^n$
- ***covariance matrix*** $\Sigma \in \mathbb{R}^{n \times n}$
  - $\Sigma \geq 0$ is symmetric and positive semi-definite

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

For $X \sim \mathcal{N}(\mu, \Sigma)$:

$$\mathsf{E}[X] = \int_x x p(x; \mu, \Sigma) dx = \mu$$

For a vector-valued random variable Z, $\mathsf{Cov}(Z) = \mathsf{E}\left[(Z - \mathsf{E}[Z])(Z - \mathsf{E}[Z])^T\right]$, generalizing the variance to multiple dimensions.

$$\begin{aligned}
\mathsf{Cov}(Z) &= \mathsf{E}\left[(Z - \mathsf{E}[Z])(Z - \mathsf{E}[Z])^T\right] \\
&= \mathsf{E}\left[(Z - \mathsf{E}[Z])\left(Z^T - \left(\mathsf{E}[Z]^T\right)\right)\right] && \text{transpose respects addition} \\
&= \mathsf{E}\left[(Z - \mathsf{E}[Z])\left(Z^T - \mathsf{E}[Z]\right)\right] && \text{transpose of a scalar: } \mathsf{E}[Z] = \left(\mathsf{E}[Z]\right)^T \\
&= \mathsf{E}\left[ZZ^T - Z\mathsf{E}[Z] - \mathsf{E}[Z]Z^T + \mathsf{E}^2[Z]\right] \\
&= \mathsf{E}\left[ZZ^T\right] - \mathsf{E}[Z\mathsf{E}[Z]] - \mathsf{E}[\mathsf{E}[Z]Z^T] + \mathsf{E}[\mathsf{E}^2[Z]] \\
&= \mathsf{E}\left[ZZ^T\right] - \mathsf{E}^2[Z] - \mathsf{E}[Z]\mathsf{E}\left[Z^T\right] + \mathsf{E}^2[Z] && \text{expectation of a scalar: } \mathsf{E}[\mathsf{E}[Z]] = \mathsf{E}[Z] \\
&= \mathsf{E}\left[ZZ^T\right] - \mathsf{E}[Z]\mathsf{E}\left[Z^T\right] \\
&= \mathsf{E}\left[ZZ^T\right] - \left(\mathsf{E}[Z]\right)\left(\mathsf{E}\left[Z\right]\right)^T
\end{aligned}$$

$$\begin{aligned}
\therefore \mathsf{Cov}(Z) &= \mathsf{E}\left[(Z - \mathsf{E}[Z])(Z - \mathsf{E}[Z])^T\right] \\
&= \mathsf{E}\left[ZZ^T\right] - \left(\mathsf{E}[Z]\right)\left(\mathsf{E}\left[Z\right]\right)^T
\end{aligned}$$

If $X \sim \mathcal{N}(\mu, \Sigma)$, then

$$\mathsf{Cov}(X) = \Sigma$$

$X \sim \mathcal{N}(\mu = \mathbf{0}, \Sigma = I)$ is the ***standard normal distribution***.

The diagonal of the covariance matrix is the variance of each variable. The off-diagonal elements $\Sigma_{ij}, i \neq j$ are the covariance of variable i with variable j.

The Gaussian Discriminant Analysis model

Classification with continuous-valued input features x, modelling $p(x \mid y)$ using a multivariate normal distribution:

$$\begin{aligned}
y &\sim \text{Bernoulli}(\phi) \\
x \mid y = 0 &\sim \mathcal{N}(\mu_0, \Sigma) \\
x \mid y = 1 &\sim \mathcal{N}(\mu_1, \Sigma)
\end{aligned}$$

, i.e.

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

Parameters of the model: $\phi, \Sigma, \mu_0, \mu_1$. Note the assumption of equal covariance matrices between the classes. Relaxing this assumption leads to quadratic discriminant analysis (QDA).

The log-likelihood of the data:

$$
\begin{aligned}
\ell(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^{m} p\left(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma\right) \\
&= \log \prod_{i=1}^{m} p\left(x^{(i)} \mid y^{(i)}; \mu_0, \mu_1, \Sigma\right) p\left(y^{(i)}; \phi\right) \quad \text{chain rule of conditional prob.}
\end{aligned}
$$

GDA and logistic regression

The GDA model can be thought of as the generative analog to the discriminative logistic regression algorithm:

$$
\begin{aligned}
p(y=1 \mid x; \phi, \mu_0, \mu_1, \Sigma) &= \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right) \\
&= \frac{1}{1+\exp(-\theta^T x)}, \qquad \text{where } \theta = f\left(\phi, \Sigma, \mu_0, \mu_1\right)
\end{aligned}
$$

, which is the form of logistic regression. The above argument says that if $p(x \mid y)$ is multivariate Gaussian (with shared Σ), then $p(y \mid x)$ follows a logistic function. However, the converse is not true: $p(y \mid x)$ being a logistic function does not imply $p(x \mid y)$ is multivariate Gaussian. So, GDA makes *stronger* modeling assumptions about the data than logistic regression. When these assumptions are correct, then GDA will better fit to the data, and is a better model. Specifically, when $p(x \mid y)$ is Gaussian (with shared Σ), then GDA is ***asymptotically efficient***- in the limit of large training sets, there is no algorithm that is strictly better than GDA. In constrast, logistic regression makes weaker assumptions about the data and is thus more *robust* to incorrect modelling assumptions. When the data is non-Gaussian, then in the limit of large datasets, logistic regression will usually outperform GDA.

Naive Bayes

- Consider x_i's which are discrete-valued, e.g. vectors where each element represents the i-th word of a dictionary.
- $x_i = 1$ if the word is included, otherwise $x_i = 0$.
- set of words encoded in the feature vector $\rightarrow$ **vocabulary**.
  - dimension of x = size of vocabulary
- to build a generative model, have to model $p(x \mid y)$
  - consider vocabulary of 50,000 words
  - then, $x \in \{0,1\}^{50000}$
    - modelling x with a multinomial distribution over the 2^{50000} possible outcomes results in a $(2^{50000}-1)$-dimensional parameter vector (way too many.).
    - to make $p(x \mid y)$ tractable, make a strong assumption- the ***Naive Bayes (NB) assumption***, which assumes that the x_i's are conditionally independent given y. The resulting algorithm is called the ***Naive Bayes classifier***. Then,

$$
\begin{aligned}
& p(x_1, \ldots, x_{50000} \mid y) \\
& \quad = p(x_1 \mid y) p(x_2 \mid y, x_1) p(x_3 \mid y, x_1, x_2) \cdots p(x_{50000} \mid y, x_1, \ldots, x_{49999}) && \text{chain rule of probabilities} \\
& \quad = p(x_1 \mid y) p(x_2 \mid y) p(x_3 \mid y) \cdots p(x_{50000} \mid y) && \text{NB assumption} \\
& \quad = \prod_{i=1}^{n} p(x_i \mid y)
\end{aligned}
$$

* the Naive Bayes assumption is an extremely strong assumption, but the resulting algorithm works well on many problems

- model parameters:

$$
\begin{aligned}
\phi_{i|y=1} &= p(x_i = 1 \mid y = 1), \\
\phi_{i|y=0} &= p(x_i = 1 \mid y = 0), \\
\phi_y &= p(y = 1)
\end{aligned}
$$

- joint likelihood:

$$
\mathcal{L}(\phi_y, \phi_{i|y=0}, \phi_{i|y=1}) = \prod_{i=1}^{m} p\left(x^{(i)}, y^{(i)}\right)
$$

Maximize joint likelihood w.r.t. ϕ_y, $\phi_{i|y=0}$ and $\phi_{i|y=1}$ to get maximum likelihood estimates:

$$
\begin{aligned}
\phi_{j|y=1} &= \frac{\sum_{i=1}^{m} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\right\}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\}} \\
\phi_{j|y=0} &= \frac{\sum_{i=1}^{m} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\right\}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 0\right\}} \\
\phi_y &= \frac{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\}}{m}
\end{aligned}
$$

Where the $\wedge$ symbol means “and”. The parameters have very natural interpretations:

- $\phi_{j|y=1}$: the fraction of examples of class 1 in which feature j appears
- $\phi_{j|y=0}$: the fraction of examples of class 0 in which feature j appears
- ϕ_y: the fraction of examples which are of class 1

To make a prediction on a new example with features x, calculate:

$$
\begin{aligned}
p(y = 1 \mid x) &= \frac{p(x \mid y = 1) p(y = 1)}{p(x)} \\
&= \frac{\left(\prod_{i=1}^{n} p(x_i \mid y = 1)\right) p(y = 1)}{\left(\prod_{i=1}^{n} p(x_i \mid y = 1)\right) p(y = 1) + \left(\prod_{i=1}^{n} p(x_i \mid y = 0)\right) p(y = 0)}
\end{aligned}
$$

and classify to whichever class has the higher posterior probability.

Naive Bayes can be generalized to the case where $x_i \in \{1, 2, \ldots, k\}$ by modelling $p(x_i \mid y)$ as multinomial instead of Bernoulli.

Continuous input variables can be adapted for Naive Bayes by ***discretizing***, i.e. splitting the continuous value into bins. When continuous input variables are not well-modeled by a multivariate normal distribution, discretizing and using Naive Bayes often results in a better classifier than GDA.

Laplace smoothing

The Naive Bayes algorithm runs into a problem when it encounters examples with features that were not seen in the training set. For such a feature f, maximum likelihood picks the following parameters:

$$\phi_{f|y=1} = 0$$
$$\phi_{f|y=0} = 0$$

If the feature is present in an example to be classified, then the class posterior probability is as follows:

$$\begin{aligned} p(y=1 \mid x) &= \frac{\left(\prod_{i=1}^{n} p(x_i \mid y=1)\right) p(y=1)}{\left(\prod_{i=1}^{n} p(x_i \mid y=1)\right) p(y=1) + \left(\prod_{i=1}^{n} p(x_i \mid y=0)\right) p(y=0)} \\ &= \frac{0}{0} \qquad \text{, since each } \prod_{i=1}^{n} p(x_i \mid y) \text{ contains } p(x_f \mid y) = 0 \end{aligned}$$

The problem is that the prior probability for features that are not seen in the finite training set is set to zero, when it is unlikely to be exactly zero. This can be addressed by ***Laplace smoothing***, which replaces the estimates for ϕ_j with

$$\phi_j = \frac{\sum_{i=1}^{m} 1\left\{z^{(i)} = j\right\} + 1}{m + k}$$

The maximum likelihood estimates of the parameters for Naive Bayes with Laplace smoothing:

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\right\} + 1}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\} + 2}$$
$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\right\} + 1}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 0\right\} + 2}$$

Event models for text classification

Naive Bayes as presented above uses the ***multi-variate Bernoulli event model***. This model assumes that the class priors first determine which class an example comes $(p(y))$. Then, the class-conditional probabilities $p(x_i = 1 \mid y) = \phi_{i|y}$ are used to decide whether each word in the dictionary is added to the email, independently of the other words. The probability of a message is $p(y) \prod_{i=1}^{n} p(x_i \mid y)$.

A different model is the ***multinomial event model***.

- Let x_i denote the identity of the i-th word
  - $x_i \in \{1, \ldots, |V|\}$
  - $|V|$ is the size of the vocabulary (dictionary)
- an email of n words is represented by a vector $(x_1, x_2, \ldots, x_n)$ of length n, where n can vary for different documents

To generate a document. Use class priors to choose which class (same as Naive Bayes). Generate x_1 from a multinomial distribution over words $(p(x_1 \mid y))$, then generate x_2 independently but from the same distribution and so on until x_n. The probability of a message is $p(y) \prod_{i=1}^{n} p(x_i \mid y)$, which looks like the probability under Naive Bayes, but differs since $x_i \mid y$ is multinomial rather than Bernoulli.

Parameters for the multinomial event model:

$$\begin{aligned} \phi_y &= p(y) \\ \phi_{k|y=1} &= p(x_j = k \mid y = 1) \quad \text{(for any } j \text{, i.e. a word's distribution is independent of its position } j) \\ \phi_{i|y=0} &= p(x_j = k \mid y = 0) \end{aligned}$$

Given a training set $\left\{\left(x^{(i)}, y^{(i)}\right); i = 1, \ldots, m\right\}$ where $x^{(i)} = \left(x_1^{(i)}, x_2^{(i)}, \ldots, x_{n_i}^{(i)}\right)$ (where $n_i = $number of words in the ith training example), the likelihood of the data is given by:

$$\begin{aligned}\mathcal{L}\left(\phi, \phi_{k|y=0}, \phi_{k|y=1}\right) &= \prod_{i=1}^{m} p\left(x^{(i)}, y^{(i)}\right) \\ &= \prod_{i=1}^{m}\left(\prod_{j=1}^{n_i} p\left(x_j^{(i)} \mid y; \phi_{k|y=0}, \phi_{k|y=1}\right)\right) p\left(y^{(i)}; \phi_y\right).\end{aligned}$$

Maximize to get the maximum likelihood estimates of the parameters:

$$\begin{aligned}\phi_{k|y=1} &= \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\left\{x_j^{(i)} = k \wedge y^{(i)} = 1\right\}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\} n_i} \\ \phi_{k|y=0} &= \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\left\{x_j^{(i)} = k \wedge y^{(i)} = 0\right\}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 0\right\} n_i} \\ \phi_y &= \frac{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\}}{m}\end{aligned}$$

MLE estimates with Laplace smoothing: Add 1 to numerators and $|V|$ to the denominators to obtain:

$$\begin{aligned}\phi_{k|y=1} &= \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\left\{x_j^{(i)} = k \wedge y^{(i)} = 1\right\} + 1}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\} n_i + |V|} \\ \phi_{k|y=0} &= \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\left\{x_j^{(i)} = k \wedge y^{(i)} = 0\right\} + 1}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 0\right\} n_i + |V|}\end{aligned}$$

Naive Bayes: not necessarily the best classification algorithm, but often works surprisingly well. Often a good “first thing to try”, due to its simplicity and ease of implementation.