Escuela Politécnica de Ingeniería de Gijón EPI Universidad de Oviedo
Grado en Ciencia e Ingeniería de Datos

AA1 Aprendizaje Automático 1

T5: Classification, logistic regression and generative learning

Antonio Bahamonde
Departamento de Informática

Contents

- Classification and logistic regression (Main notes, chapter 2)
- Generative learning algorithms (chapter 4)
  - Naive Bayes

Classification
Logistic regression
(Chapter 2)

Classification

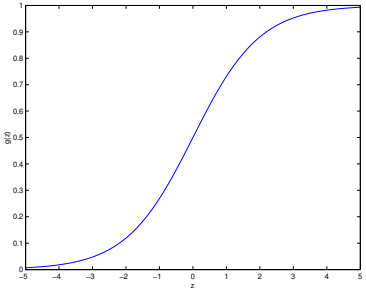
- We will focus on the **binary** classification problem in which y can take on only two values, 0 and 1. (Most of what we say here will also generalize to the multiple-class case.)
  - *For instance, if we are trying to build a spam classifier for email, then $x^{(i)}$ may be some features of a piece of email, and $y^{(i)}$ may be 1 if it is a piece of spam mail, and 0 otherwise. 0 is also called the negative class, and 1 the positive class, and they are sometimes also denoted by the symbols "-" and "+."*
- Given $x^{(i)}$, the corresponding $y^{(i)}$ is also called the **label** for the training example.

Logistic regression

It doesn't make sense for $h_\theta(x)$ to take values larger than 1 or smaller than 0 when we know that $y \in \{0, 1\}$. To fix this, let's change the form of $h_\theta(x)$

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

$$g(z) = \frac{1}{1 + e^{-z}}$$



logistic function or the **sigmoid** function

Logistic regression

Let's endow our classification model with a set of probabilistic assumptions, and then fit the parameters via **maximum likelihood.**

$$\begin{aligned} P(y = 1 \mid x; \theta) &= h_\theta(x) \\ P(y = 0 \mid x; \theta) &= 1 - h_\theta(x) \end{aligned}$$

$$p(y \mid x; \theta) = (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$$

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X; \theta) \\ &= \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{n} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{n} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \end{aligned}$$

Logistic regression

Gradient descent update rule

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{n} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \end{aligned}$$

$$
\begin{aligned}
g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z)).
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial\theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial\theta_j}g(\theta^T x) \\
&= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)g(\theta^T x)(1-g(\theta^T x))\frac{\partial}{\partial\theta_j}\theta^T x \\
&= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right)x_j \\
&= (y-h_\theta(x))\,x_j
\end{aligned}
$$

Logistic regression

Gradient descent update rule

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}$$

it looks identical than LMS; but this is not the same algorithm, because $h_\theta(x^{(i)})$ is now defined as a non-linear function of $\theta^T x^{(i)}$. Nonetheless, it's a little surprising that we end up with the same update rule for a rather different algorithm and learning problem.

Generative learning algorithms

(Chapter 4)

Generative

- So far, we've mainly been talking about learning algorithms that model p(y|x; θ), the conditional distribution of y given x. For instance, logistic regression modeled p(y|x; θ) as $h_\theta(x) = g(\theta^T x)$ where g is the sigmoid function
- Consider a classification problem in which we want to learn to distinguish between elephants (y = 1) and dogs (y = 0), based on some features of an animal.
- to classify a new animal, we can match the new animal against the elephant model, and match it against the dog model, to see whether the new animal looks more like the elephants or more like the dogs we had seen in the training set.

Generative

p(y|x) directly (such as logistic regression), or algorithms that try to learn mappings directly from the space of inputs X to the labels {0, 1}, are called **discriminative** learning algorithms.

Here, we'll talk about algorithms that instead try to model p(x|y) (and p(y)). These algorithms are called **generative** learning algorithms.

Generative

After modeling p(y) (called the class priors) and p(x|y), our algorithm can then use Bayes rule to derive the posterior distribution on y given x:

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

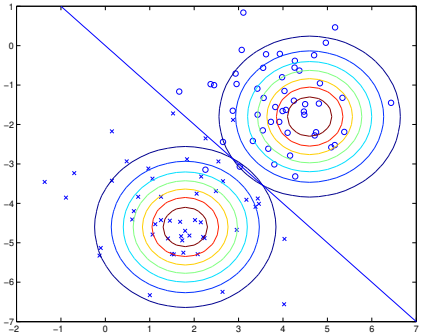
And then

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

Generative: The Gaussian discriminant analysis (GDA) model

When we have a classification problem in which the input features x are continuous-valued random variables, we can then use the Gaussian Discriminant Analysis (GDA) model, which models p(x|y) using a multivariate normal distribution.

The model is:



Generative: The Gaussian discriminant analysis (GDA) model

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

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

$$\begin{aligned} \ell(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^n p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \\ &= \log \prod_{i=1}^n p(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi). \end{aligned}$$

$$\begin{aligned} \phi &= \frac{1}{n}\sum_{i=1}^n 1\{y^{(i)}=1\} \\ \mu_0 &= \frac{\sum_{i=1}^n 1\{y^{(i)}=0\}x^{(i)}}{\sum_{i=1}^n 1\{y^{(i)}=0\}} \\ \mu_1 &= \frac{\sum_{i=1}^n 1\{y^{(i)}=1\}x^{(i)}}{\sum_{i=1}^n 1\{y^{(i)}=1\}} \\ \Sigma &= \frac{1}{n}\sum_{i=1}^n (x^{(i)}-\mu_{y^{(i)}})(x^{(i)}-\mu_{y^{(i)}})^T. \end{aligned}$$

GDA and logistic regression

- If p(x|y) is multivariate gaussian (with shared Σ), then p(y|x) necessarily follows a logistic function. The converse, however, is not true; i.e., p(y|x) being a logistic function does not imply p(x|y) is multivariate gaussian.
- This shows that GDA makes stronger modeling assumptions about the data than does logistic regression. When these modeling assumptions are correct, then GDA will find better fits to the data
- Informally, this means that in the limit of very large training sets, there is no algorithm that is strictly better than GDA
- GDA will be a better algorithm than logistic regression; and more generally, even for small training set sizes, we would generally expect GDA to better. In contrast, by making significantly weaker assumptions, logistic regression is also more robust and less sensitive to incorrect modeling assumptions

Generative: Naive Bayes

- A learning algorithm in which the x_j's are discrete-valued.

Text classification: Naive Bayes

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \text{a} \\ \text{aardvark} \\ \text{aardwolf} \\ \vdots \\ \text{buy} \\ \vdots \\ \text{zygmurgy} \end{matrix}$$

- Training set (a set of emails labeled as spam or non-spam)
- if an email contains the j-th word of the dictionary, then we will set $x_j = 1$; otherwise, we let $x_j = 0$

Text classification: Naive Bayes

- To model p(x|y) we use a very strong assumption: x_i's are conditionally independent given y. This assumption is called the **Naive Bayes (NB) assumption**, and the resulting algorithm is called the Naive Bayes classifier.

$$\begin{aligned} & p(x_1, \ldots, x_{50000}|y) \\ &= p(x_1|y)p(x_2|y, x_1)p(x_3|y, x_1, x_2)\cdots p(x_{50000}|y, x_1, \ldots, x_{49999}) \\ &= p(x_1|y)p(x_2|y)p(x_3|y)\cdots p(x_{50000}|y) \\ &= \prod_{j=1}^{d} p(x_j|y) \end{aligned}$$

$$\phi_{j|y=1} = p(x_j = 1|y = 1)$$

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

Text classification: Naive Bayes

- To make a prediction on a new example with features x, we then simply calculate and pick whichever class has the higher posterior probability

$$
\begin{aligned}
p(y=1|x) &= \frac{p(x|y=1)p(y=1)}{p(x)} \\
&= \frac{\left(\prod_{j=1}^{d} p(x_j|y=1)\right) p(y=1)}{\left(\prod_{j=1}^{d} p(x_j|y=1)\right) p(y=1) + \left(\prod_{j=1}^{d} p(x_j|y=0)\right) p(y=0)},
\end{aligned}
$$

Naive Bayes with Laplace smoothing

To estimate the mean of a multinomial random variable z taking values in {1, . . . , k}

$$\phi_j = \frac{\sum_{i=1}^{n} 1\{z^{(i)} = j\}}{n}. \qquad \phi_j = \frac{1 + \sum_{i=1}^{n} 1\{z^{(i)} = j\}}{k + n}.$$

$$
\begin{aligned}
\phi_{j|y=1} &= \frac{1 + \sum_{i=1}^{n} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\}}{2 + \sum_{i=1}^{n} 1\{y^{(i)} = 1\}} \\
\phi_{j|y=0} &= \frac{1 + \sum_{i=1}^{n} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\}}{2 + \sum_{i=1}^{n} 1\{y^{(i)} = 0\}}
\end{aligned}
$$

Más información

- Classification and logistic regression (Main notes, chapter 2)
- Chapter 4
- Libro de Tom Mitchell chapter 3: GENERATIVE AND DISCRIMINATIVE CLASSIFIERS: NAIVE BAYES AND LOGISTIC REGRESSION

http://www.cs.cmu.edu/%7Etom/mlbook/NBayesLogReg.pdf