Gaussian Discriminant Analysis

CCS Machine Learning, based on Andrew Ng's CS 229 Lecture Notes

Introduction

- We just talked about Generative -vs- Discriminative models
- Although GDA (or NDA) had discriminative in the name, it is a GENERATIVE model.
- What does this mean?
  - We want to model p(x | y)
- Assumption: p(x | y) is distributed according to a multivariate normal distribution

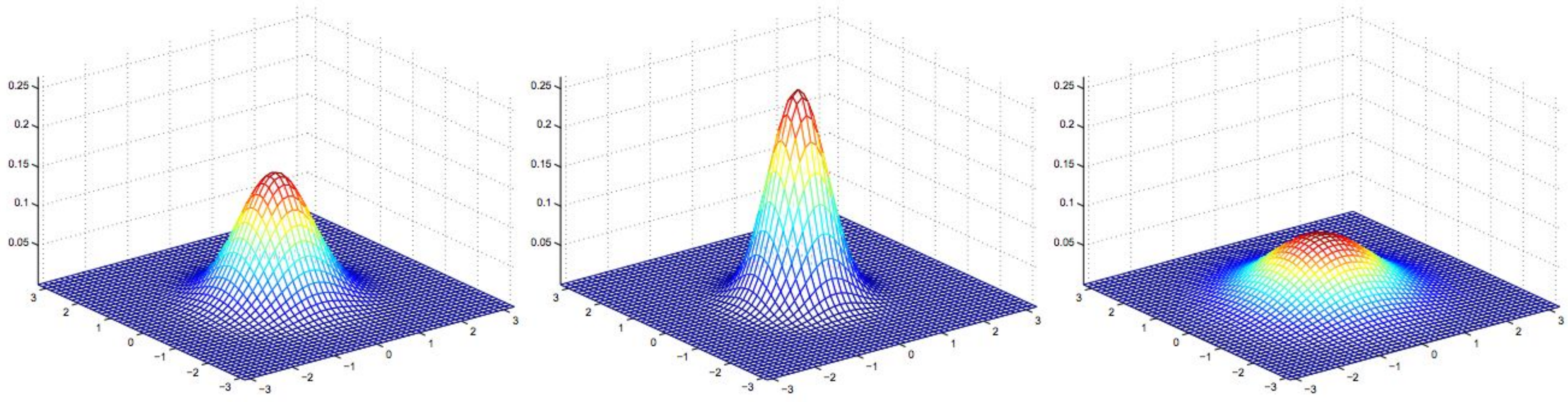
Multivariate Normal Distribution

- MVN in n-dimensions is parametrized by a mean vector $\mu \in \mathbb{R}^n$ and a covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$
- $\Sigma \geq 0$ is symmetric and positive semi-definite
- The density is as follows:

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

- The absolute value indicates the determinant of the matrix in the above equation.
- We will also use the notation $\mathcal{N}(\mu, \Sigma)$.

Examples



- Left is a Gaussian with mean zero and covariance matrix equal to I, i.e. a Standard Normal
- Middle has a gaussian with zero mean and has a covariance matrix of .6*I
- Right is a gaussian with zero mean and has a covariance matrix of 2*I
- Note that the mean is a coordinate pair, and the covariance is 2x2

GDA

- Context:
  - Classification, specifically binary classification
  - Input features x are continuous-valued random variables
- We are modeling p(x | y) using a multivariate normal distribution
- The model:

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

- Explicitly:

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

Log-Likelihood

- Just like in logistic regression and linear regression, we also aim to maximize the likelihood function of our parameters by using the log-likelihood

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

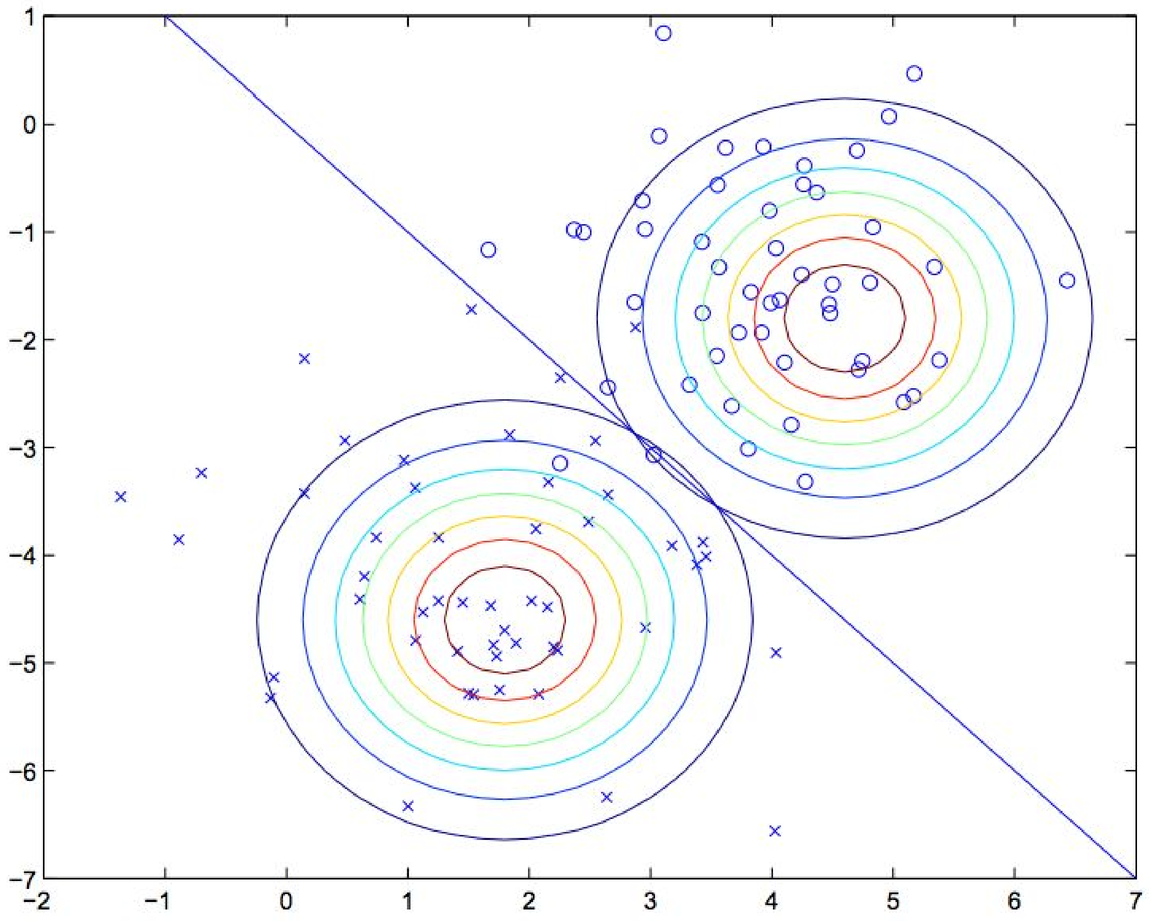
Maximum Likelihood Parameters

- After a bit of arithmetic, the maximum parameters are as follows:

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

Pictorially what is Learned:

Notice that the covariance matrix is shared between each of the classes. This means that we are modeling two separate gaussians, with the same shape. As a result, they will be translations of one another.



Generalizing Gaussian Discriminant Analysis

- Linear Discriminant Analysis (LDA) is another name for GDA because of the decision boundary that is learned
- LDA:
  - class-specific mean, common covariance matrix
- Quadratic Discriminant Analysis (QDA):
  - class-specific mean, class-specific covariance matrix
- General Discriminant Analysis (GDA):
  - Encompasses both LDA and QDA
  - Modeling p(x | y) by multivariate Gaussians
- Note that is why people use NDA or LDA for Gaussian Discriminant Analysis. Acronyms should not overlap.

Relationship to Logistic Regression

- GDA learns p(x | y) and logistic regression learns p(y | x)
- It turns out, with our GDA assumptions, if we view the quantity $p(y=1|x;\phi,\mu_0,\mu_1,\Sigma)$ as a function of x, we find that it can be expressed in the form:

$$p(y=1|x;\phi,\Sigma,\mu_0,\mu_1)=\frac{1}{1+\exp(-\theta^T x)}$$

- This is precisely logistic regression!
- Differences:
  - They learn different decision boundaries for the same set of data
  - If we try to model p(y | x) with a logistic function, it is not necessarily the case that p(x | y) is MVN

Which is Better?

- GDA makes stronger modeling assumptions, but when these modeling assumptions are correct it is better, in fact it is **asymptotically efficient**
  - In the limit of large training sets (large m) no algorithm is strictly better than GDA
  - By better we mean how accurately they estimate p(y | x)
- Even for small training sets we expect GDA to be better
- However, logistic regression makes weaker assumptions, which means that it is more robust and less sensitive to incorrect modeling assumptions
- E.g.
  - x | y =0 ~ Poisson(L_0), x | y = 1 ~ Poisson(L_1)
  - p(y | x) is logistic
  - GDA might not do as well

To Summarize:

- GDA:
  - makes stronger modeling assumptions
  - More data efficient
    - Requires less data to learn “well”
- Logistic Regression:
  - Makes weaker assumptions
  - More robust to deviations from modeling assumptions
- If the data is not Gaussian:
  - In the limit of large datasets, logistic regression will almost always perform better than GDA
  - This is why logistic regression is used more in practice

Moving From Continuous Features to Discrete

- In GDA the feature vectors x were continuous, real-valued vectors
- What if the xi’s are discrete-valued?
- We will build a model for p(x | y) where the xi’s are discrete
- We will use a classifier called a Naive Bayes Classifier to do this

Spam Filter

- Classify spam -vs- not spam
- We build feature vectors whose length is equal to the number of words in a dictionary
- Using our training set, we create a vocabulary, V
- The dimension of x is equal to |V|
- If we have a vocabulary of 50000 words,

$$x \in \{0,1\}^{50000}$$

$$x = \begin{array}{c} \left[\begin{array}{c} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{array}\right] \end{array} \begin{array}{l} \text{a} \\ \text{aardvark} \\ \text{aardwolf} \\ \vdots \\ \text{buy} \\ \vdots \\ \text{zygmurgy} \end{array}$$

Naive Bayes (NB)

- To model p (x | y), we will make a very strong assumption
- We assume the xi's are conditionally independent given y
  - This is the Naive Bayes assumption
- E.g.
  - If y = 1 means spam and "buy" is word 2087 and "price" is word 39831; then we are assuming that if y = 1, then knowing the value of word 2087 will have no effect on word 39831
  - This is probably not true
- Thus, we are learning:

$$
\begin{aligned}
& p(x_1, \ldots, x_{50000} | y) \\
& \quad = p(x_1|y)p(x_2|y, x_1)p(x_3|y, x_1, x_2) \cdots p(x_{50000}|y, x_1, \ldots, x_{49999}) \\
& \quad = p(x_1|y)p(x_2|y)p(x_3|y) \cdots p(x_{50000}|y) \\
& \quad = \prod_{i=1}^{n} p(x_i|y)
\end{aligned}
$$

The Joint Likelihood

- Given our training set in the same notation as we always do:
- The Likelihood is:

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

- Maximizing the likelihood with respect to our parameters yields:

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

Making a Prediction

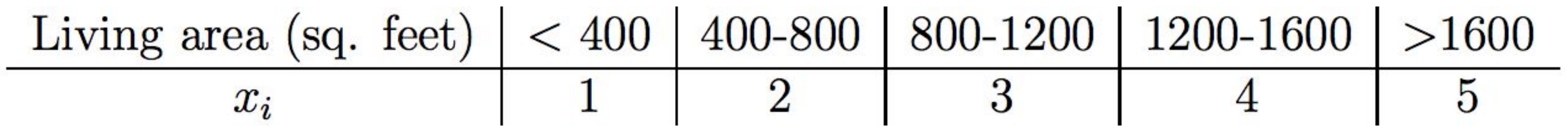
- Given a new example with features x:

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

- Repeat for all classes, and choose the class with the highest probability

Naive Bayes -vs- GDA

- What if our data was not Gaussian? GDA would perform poorly but if we discretize our data, we can still apply Naive Bayes
- E.g.
  - If we use some feature xi to represent living area, we can discretize it as follows:



Living area (sq. feet)	< 400	400-800	800-1200	1200-1600	>1600
x_i	1	2	3	4	5

- Now we can model p (xi | y) with a multinomial distribution (instead of a Bernoulli)

Generative -vs- Discriminative

- Easy to fit?
  - Generative are typically easier as they sometimes only require counting and averaging
  - Discriminative requires solving an optimization problem
- Fit classes separately?
  - In generative, we do not need to retrain the model when adding additional classes
  - In discriminative, all the model parameters interact, so retraining is a must
- Handle missing features easily?
  - Generative classifiers handle this by marginalizing the missing points out
  - Discriminative does not have an easy way to do this
- Symmetric in inputs and outputs?
  - We can run a generative model “backwards” and infer probable inputs given the output by computing $p(x \mid y)$. This is because we are modeling the joint probability
  - Not possible in discriminative

Generative -vs- Discriminative Continued

- Can handle feature preprocessing?
  - New features can be correlated in complex ways, making it tough to do with a generative model
  - This can be done arbitrarily with discriminative models. Just replace x with f(x), with f some basis function
- Well-calibrated probabilities?
  - Generative models such as Naive Bayes make strong independence assumptions which are often not valid
  - Discriminative models are better calibrated as they do not make as many assumptions about the underlying data

Done