Multivariate Normal and Discriminate Analysis
Nicole Beckage

Normal Distribution
\[
x \sim N(\mu, \sigma)
\]
\[
f(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[ -\frac{(x - \mu)^2}{2\sigma^2} \right]
\]
\[
x \sim N(\mu, \Sigma)
\]
\[
f(x | \mu, \Sigma) = \frac{1}{(2\pi)^{p/2}\sqrt{\det(\Sigma)}} \exp \left[ -\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \right]
\]

Bivariate normal
\[
f(x, y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp \left[ -\frac{1}{2(1-\rho^2)} \left( \frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - \frac{2\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} \right) \right]
\]
where \(\rho\) is the correlation between \(X\) and \(Y\) and where \(\sigma_x > 0\) and \(\sigma_y > 0\).

Then we can see that

\[
\mu = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}
\]
\[
\Sigma = \begin{bmatrix} \sigma_x^2 & \rho \sigma_x \sigma_y \\ \rho \sigma_x \sigma_y & \sigma_y^2 \end{bmatrix}
\]

Covariance Matrix
\[
\Sigma = \begin{bmatrix}
\sigma_X^2 & \rho_{X_1 X_2} \sigma_X \sigma_Y & \cdots & \rho_{X_1 X_p} \sigma_X \sigma_Y \\
\rho_{X_2 X_1} \sigma_X \sigma_Y & \sigma_Y^2 & \cdots & \rho_{X_2 X_p} \sigma_Y^2 \\
\vdots & \vdots & \ddots & \vdots \\
\rho_{X_p X_1} \sigma_X \sigma_Y & \rho_{X_p X_2} \sigma_Y \sigma_X & \cdots & \sigma_Y^2
\end{bmatrix}
\]

• Diagonal is the variance
• Off diagonals are covariance of \(i, j\)
• Note the matrix is symmetric and positive semi-definite

Benefits of (MV) Gaussians
• We can talk about Gaussians as ellipses
• That lie along a transformed Euclidean distance coordinate system
  • Eigenvectors determine orientation
  • Eigenvalues determine elongation
• \(u_1, u_2\) are eigenvectors, \(\lambda_1, \lambda_2\) are eigenvalues

Visualization of MVN
\[
\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} ; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} ; \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}.
\]
Visualization of MVN

|\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}| |\Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}| |\Sigma = \begin{bmatrix} 3 & 0.8 \\ 0.8 & 1 \end{bmatrix}|

Benefits of MVNs

• If we want to maximize expressible of a distribution, we want to maximize entropy
• Many times we can only approximate mean and variance from the data directly
  • These are considered the first two moments of a distribution
  • Gaussians maximize entropy given specified means and covariance.
  • You should know this. You won’t need to prove this.
  • Book has the proof. Don’t need to know but should be able to explain why high entropy means high uncertainty which is good for modeling

Gaussian discriminant analysis

• Generative model
  • Name is after mathematical similar model known as Fishers discriminant analysis.
  \[ p(x|y = c, \theta) = \mathcal{N}(x|\mu_c, \Sigma_c) \]
• Class of models. We’ll focus on 2 subtypes
  • Quadratic discriminant analysis
  • Linear discriminant analysis

High level idea

• Let’s assume there is c labels
• Each class generates data that itself is MVN
  • Terms can be correlated (non-independent) with each other (off diagonal entries are non-zero)
  • So each class has its own \( \mu_c, \Sigma_c \)
• Decision criterion:
  • Whatever \( \mu_c, \Sigma_c \) maximizes the log-likelihood is our best class estimate

Example: Height/weight data
Linear Discriminant Analysis

\[ p(y = c|x, \theta) \]

\[ \propto (\pi_c/(2\pi\Sigma))^{-1/2} \exp \left[ -\frac{1}{2} (x - \mu_c)^T \Sigma^{-1} (x - \mu_c) \right] \]

\[ \propto \pi_c \exp \left[ \mu_c^T \Sigma^{-1} x - \frac{1}{2} (x - \mu_c)^T \Sigma^{-1} (x - \mu_c) \right] \]

\[ = \exp \left[ \mu_c^T \Sigma^{-1} x - \frac{1}{2} \mu_c^T \Sigma^{-1} \mu_c + \log \pi_c \right] \exp \left[ -\frac{1}{2} x^T \Sigma^{-1} x \right] \]

\[ \propto \exp \left[ \mu_c^T \Sigma^{-1} x - \frac{1}{2} \mu_c^T \Sigma^{-1} \mu_c + \log \pi_c \right] \]

Simplifying the result

- Let \( \gamma_c = -\frac{1}{2} \mu_c^T \Sigma^{-1} \mu_c + \log \pi_c \)
- \( \beta_c = \Sigma^{-1} \mu_c \)
- Then \( p(y = c|x, \theta) = \frac{\exp[\beta_c^T x + \gamma_c]}{\sum_c \exp[\beta_c^T x + \gamma_c]} \)

MLE proof

**MLE: 2 class classification (simplest case)**

- \( y \sim \text{Bernoulli}(\pi) \)
- \( x|y = 0 \sim \mathcal{N}(\mu_0, \Sigma) \)
- \( x|y = 1 \sim \mathcal{N}(\mu_1, \Sigma) \)

\[ p(y) = \pi^y (1 - \pi)^{1-y} \]

\[ p(x|y = 0) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu_0)^T \Sigma^{-1} (x - \mu_0) \right] \]

\[ p(x|y = 1) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu_1)^T \Sigma^{-1} (x - \mu_1) \right] \]

**MLE Linear Discriminant Analysis**

\[ \ell(\pi, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^N p(x_i, y; \pi, \mu, \Sigma) \]

\[ = \log \prod_{i=1}^N \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (x_i - \mu_0)^T \Sigma^{-1} (x_i - \mu_0) \right] \exp \left[ -\frac{1}{2} (x_i - \mu_1)^T \Sigma^{-1} (x_i - \mu_1) \right] \]
MLE Linear Discriminant Analysis

- Maximize loss function with respect to each parameter
  
  \[ \pi_1 = \frac{1}{N} \sum_{i=1}^{N} I(y_i = 1) \]
  
  \[ \mu_0 = \frac{\sum_{i=1}^{N} I(y_i = 0) x_i}{\sum_{i=1}^{N} I(y_i = 0)} \]
  
  \[ \mu_1 = \frac{\sum_{i=1}^{N} I(y_i = 1) x_i}{\sum_{i=1}^{N} I(y_i = 1)} \]
  
  \[ \Sigma = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu_y)(x_i - \mu_y)^T \]

Relation to Naïve Bayes

- They are both generative models!
- But what if our GDA model has no co-variance?
  - E.g. \( \Sigma \) is diagonal
- It’s getting more similar
  - But in the case of Naïve Bayes we only discussed categorical variables

Relation to Naïve Bayes

- \( x \sim \mathcal{N}(\mu, \Sigma^{diag}) \)
- \[ p(X = x) = \sum_{c=1}^{k} p(c) p(x_1, \ldots, x_T | c) \]

  now since we know that \( x_1 \) is conditionally independent of \( x_2 \)
  given a particular class label, we can simplify that further
  
  \[ p(X = x) = \sum_{c=1}^{k} p(c) \prod_{t=1}^{T} p(x_t | c) \]

Neural Networks

- That was our Naïve Bayes model!
- So we’ve extended Naïve Bayes to take real valued input
- There’s some assumption that the real valued input is normally distributed but in practice it doesn’t actually matter
Neural in neural networks

How the brain works (simplified)

- Cascades: Each neuron receives input from (many) other neurons
- If activated enough it activates other neurons
  - "enough" activation indicates that the quantity of activation or the synaptic weights are strong enough to trigger a neuronal spike
- The synaptic weights adapt so the whole network learns
- About $10^{55}$ neurons, with $10^8$ weights per neuron
- Relatively slow, response speed on the order of $10^3 - 10^2$ seconds

(Deep) Neural Networks

Flow of activation

Neural Networks
Neural networks: the role of the hidden layer

• We can increase our predictive accuracy by including features that are more and more relevant
• If we had handpicked features we might be able to have a more accurate model
• Can we automate the design of features? Let the computer design the features?
  • Hidden layers of a neural network
  • Weighting input creates new (hopefully useful) features