Machine Learning - MT 2017
10. Classification : Generative Models

Varun Kanade

University of Oxford
October 30, 2017

Recap: Supervised Learning - Regression

Discriminative Model: Linear Model (with Gaussian noise)

$$p(y \mid \mathbf{w}, \mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + \mathcal{N}(0, \sigma^2)$$

Other noise models possible, *e.g.*, Laplace

Non-linearities using basis expansion

Regularisation to avoid overfitting: Ridge, Lasso

(Cross)-Validation to choose hyperparameters

Optimisation Algorithms for Model Fitting

Least Squares Ridge Lasso

1800 2017

Gauss

Legendre

Supervised Learning - Classification

In classification problems, the target/output y is a category

$$y \in \{1, 2, \dots, C\}$$

The input $\mathbf{x} = (x_1, \dots, x_D)$, where

- Categorical: $x_i \in \{1, \dots, K\}$
- Real-Valued: $x_i \in \mathbb{R}$

Discriminative Model: Only model the conditional distribution

$$p(y \mid \mathbf{x}, \boldsymbol{\theta})$$

Generative Model: Model the full joint distribution

$$p(\mathbf{x}, y \mid \boldsymbol{\theta})$$

Prediction Using Generative Models

Suppose we have a model $p(\mathbf{x}, y \mid \boldsymbol{\theta})$ over the joint distribution over inputs and outputs

Given a new input $\mathbf{x}_{\text{new}}$, we can write the conditional distribution for y

For $c \in \{1, \ldots, C\}$, we write

$$\boxed{p(y = c \mid \mathbf{x}_{\text{new}}, \boldsymbol{\theta}) = \frac{p(y = c \mid \boldsymbol{\theta}) \cdot p(\mathbf{x}_{\text{new}} \mid y = c, \boldsymbol{\theta})}{\sum_{c'=1}^{C} p(y = c' | \boldsymbol{\theta}) p(\mathbf{x}_{\text{new}} \mid y = c', \boldsymbol{\theta})}}$$

The numerator is simply the joint probability $p(\mathbf{x}_{\text{new}}, c \mid \boldsymbol{\theta})$ and the denominator the marginal probability $p(\mathbf{x}_{\text{new}} \mid \boldsymbol{\theta})$

We can pick $\widehat{y} = \operatorname{argmax}_c p(y = c \mid \mathbf{x}_{\text{new}}, \boldsymbol{\theta})$

Toy Example

Predict voter preference using in US elections

Voted in 2012?	Annual Income	State	Candidate Choice
Y	50K	OK	Clinton
N	173K	CA	Clinton
Y	80K	NJ	Trump
Y	150K	WA	Clinton
N	25K	WV	Johnson
Y	85K	IL	Clinton
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Y	1050K	NY	Trump
N	35K	CA	Trump
N	**100K**	**NY**	**?**

Classification : Generative Model

In order to fit a generative model, we'll express the joint distribution as

$$p(\mathbf{x}, y \mid \boldsymbol{\theta}, \boldsymbol{\pi}) = p(y \mid \boldsymbol{\pi}) \cdot p(\mathbf{x} \mid y, \boldsymbol{\theta})$$

To model $p(y \mid \boldsymbol{\pi})$, we'll use parameters π_c such that $\sum_c \pi_c = 1$

$$p(y = c \mid \boldsymbol{\pi}) = \pi_c$$

For class-conditional densities, for class $c = 1, \dots, C$, we will have a model:

$$p(\mathbf{x} \mid y = c, \boldsymbol{\theta}_c)$$

Classification : Generative Model

So in our example,

$$p(y = \text{clinton} \mid \boldsymbol{\pi}) = \pi_{\text{clinton}}$$
$$p(y = \text{trump} \mid \boldsymbol{\pi}) = \pi_{\text{trump}}$$
$$p(y = \text{johnson} \mid \boldsymbol{\pi}) = \pi_{\text{johnson}}$$

Given that a voter supports Trump

$$p(\mathbf{x} \mid y = \text{trump}, \boldsymbol{\theta}_{\text{trump}})$$

models the distribution over $\mathbf{x}$ given $y = \text{trump}$ and $\boldsymbol{\theta}_{\text{trump}}$

Similarly, we have $p(\mathbf{x} \mid y = \text{clinton}, \boldsymbol{\theta}_{\text{clinton}})$ and $p(\mathbf{x} \mid y = \text{johnson}, \boldsymbol{\theta}_{\text{johnson}})$

We need to pick "model" for $p(\mathbf{x} \mid y = c, \boldsymbol{\theta}_c)$

Estimate the parameters π_c, $\boldsymbol{\theta}_c$ for $c = 1, \ldots, C$

Naïve Bayes Classifier (NBC)

Assume that the features are conditionally independent given the class label

$$p(\mathbf{x} \mid y = c, \boldsymbol{\theta}_c) = \prod_{j=1}^{D} p(x_j \mid y = c, \boldsymbol{\theta}_{jc})$$

So, for example, we are ‘modelling’ that conditioned on being a trump supporter, the *state*, *previous voting* or *annual income* are <u>conditionally independent</u>

Clearly, this assumption is ‘‘naïve’’ and never satisfied

But model fitting becomes very very easy

Although the generative model is clearly inadequate, it actually works quite well

Goal is predicting class, not modelling the data!

Naïve Bayes Classifier (NBC)

Real-Valued Features

- x_j is real-valued *e.g.,* annual income
- Example: Use a Gaussian model, so $\boldsymbol{\theta}_{jc} = (\mu_{jc}, \sigma^2_{jc})$
- Can use other distributions, *e.g.,* age is probably not Gaussian!

Categorical Features

- x_j is categorical with values in $\{1, \dots, K\}$
- Use the multinoulli distribution, i.e. $x_j = i$ with probability $\mu_{jc,i}$

$$\sum_{i=1}^{K} \mu_{jc,i} = 1$$

- The special case when $x_j \in \{0, 1\}$, use a single parameter $\theta_{jc} \in [0, 1]$

Naïve Bayes Classifier (NBC)

Assume that all the features are binary, *i.e.*, every $x_j \in \{0, 1\}$

If we have C classes, overall we have only $O(CD)$ parameters, θ_{jc} for each $j = 1, \ldots, D$ and $c = 1, \ldots, C$

Without the conditional independence assumption

- We have to assign a probability for each of the 2^D combination
- Thus, we have $O(C \cdot 2^D)$ parameters!
- The ‘naïve’ assumption breaks the *curse of dimensionality* and avoids overfitting!

Maximum Likelihood for the NBC

Let us suppose we have data $\langle (\mathbf{x}_i, y_i) \rangle_{i=1}^{N}$ i.i.d. from some joint distribution $p(\mathbf{x}, y)$

The probability for a single datapoint is given by:

$$p(\mathbf{x}_i, y_i \mid \boldsymbol{\theta}, \boldsymbol{\pi}) = p(y_i \mid \boldsymbol{\pi}) \cdot p(\mathbf{x}_i \mid \boldsymbol{\theta}, y_i) = \prod_{c=1}^{C} \pi_c^{\mathbb{I}(y_i = c)} \cdot \prod_{c=1}^{C} \prod_{j=1}^{D} p(x_{ij} \mid \boldsymbol{\theta}_{jc})^{\mathbb{I}(y_i = c)}$$

Let N_c be the number of datapoints with $y_i = c$, so that $\sum_{c=1}^{C} N_c = N$

We write the log-likelihood of the data as:

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{c=1}^{C} \sum_{j=1}^{D} \sum_{i : y_i = c} \log p(x_{ij} \mid \boldsymbol{\theta}_{jc})$$

The log-likelihood is easily separated into sums involving different parameters!

Maximum Likelihood for the NBC

We have the log-likelihood for the NBC

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{c=1}^{C} \sum_{j=1}^{D} \sum_{i: y_i = c} \log p(x_{ij} \mid \boldsymbol{\theta}_{jc})$$

Let us obtain estimates for $\boldsymbol{\pi}$. We get the following optimisation problem:

$$\text{maximise} \quad \sum_{c=1}^{C} N_c \log \pi_c$$

$$\text{subject to :} \quad \sum_{c=1}^{C} \pi_c = 1$$

This constrained optimisation problem can be solved using the method of Lagrange multipliers

Constrained Optimisation Problem

Suppose $f(\mathbf{z})$ is some function that we want to maximise subject to $g(\mathbf{z}) = 0$.

Constrained Objective

$$\underset{\mathbf{z}}{\operatorname{argmax}}\, f(\mathbf{z}), \qquad \text{subject to}: \quad g(\mathbf{z}) = 0$$

Langrangian (Dual) Form

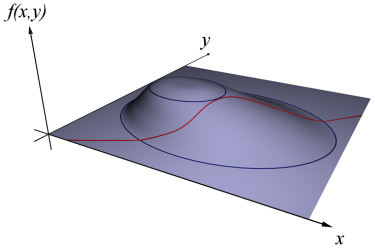
$$\Lambda(\mathbf{z}, \lambda) = f(\mathbf{z}) + \lambda g(\mathbf{z})$$

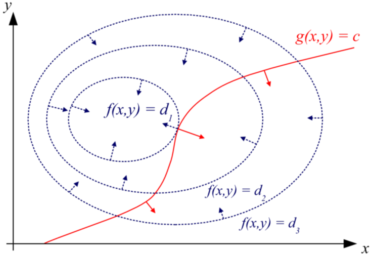
Any optimal solution to the constrained problem is a stationary point of $\Lambda(\mathbf{z}, \lambda)$

Constrained Optimisation Problem

Any optimal solution to the constrained problem is a stationary point of

$$\Lambda(\mathbf{z}, \lambda) = f(\mathbf{z}) + \lambda g(\mathbf{z})$$





$$\nabla_{\mathbf{z}} \Lambda(\mathbf{z}, \lambda) = 0 \Rightarrow \nabla_{\mathbf{z}} f = -\lambda \nabla_{\mathbf{z}} g$$

$$\frac{\partial \Lambda(\mathbf{z}, \lambda)}{\partial \lambda} = 0 \Rightarrow g(\mathbf{z}) = 0$$

Maximum Likelihood for NBC

Recall that we want to solve:

$$\text{maximise}: \quad \sum_{c=1}^{C} N_c \log \pi_c$$
$$\text{subject to}: \quad \sum_{c=1}^{C} \pi_c - 1 = 0$$

We can write the Lagrangean form:

$$\Lambda(\boldsymbol{\pi}, \lambda) = \sum_{c=1}^{C} N_c \log \pi_c + \lambda \left(\sum_{c=1}^{C} \pi_c - 1 \right)$$

We can write the partial derivatives and set them to 0:

$$\frac{\partial \Lambda(\boldsymbol{\pi}, \lambda)}{\partial \pi_c} = \frac{N_c}{\pi_c} + \lambda = 0$$
$$\frac{\partial \Lambda(\boldsymbol{\pi}, \lambda)}{\partial \lambda} = \sum_{c=1}^{C} \pi_c - 1 = 0$$

Maximum Likelihood for NBC

The solution is obtained by setting

$$\frac{N_c}{\pi_c} + \lambda = 0$$

And so,

$$\pi_c = -\frac{N_c}{\lambda}$$

As well as using the second condition,

$$\sum_{c=1}^{C} \pi_c - 1 = \sum_{c=1}^{C} -\frac{N_c}{\lambda} - 1 = 0$$

And thence,

$$\lambda = -\sum_{c=1}^{C} N_c = -N$$

Thus, we get the estimates,

$$\pi_c = \frac{N_c}{N}$$

Maximum Likelihood for the NBC

We have the log-likelihood for the NBC

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{c=1}^{C} \sum_{j=1}^{D} \sum_{i:y_i=c} \log p(x_{ij} \mid \boldsymbol{\theta}_{jc})$$

We obtained the estimates, $\pi_c = \frac{N_c}{N}$

We can estimate $\boldsymbol{\theta}_{jc}$ by taking a similar approach

To estimate $\boldsymbol{\theta}_{jc}$ we only need to use the j^{th} feature of examples with $y_i = c$

Estimates depend on the model, *e.g.*, Gaussian, Bernoulli, Multinoulli, *etc.*

Fitting NBC is very very fast!

Summary: Naïve Bayes Classifier

Generative Model: Fit the distribution $p(\mathbf{x}, y \mid \boldsymbol{\theta})$

Make the naïve and obviously untrue assumption that features are conditionally independent given class!

$$p(\mathbf{x} \mid y = c, \boldsymbol{\theta}_c) = \prod_{j=1}^{D} p(x_j \mid y = c, \boldsymbol{\theta}_{jc})$$

Despite this classifiers often work quite well in practice

The conditional independence assumption reduces the number of parameters and avoids overfitting

Fitting the model is very straightforward

Easy to mix and match different models for different features

NBC: Handling Missing Data at Test Time

Let's recall our example about trying to predict voter preferences

Voted in 2012?	Annual Income	State	Candidate Choice
Y	50K	OK	Clinton
N	173K	CA	Clinton
Y	80K	NJ	Trump
Y	150K	WA	Clinton
N	25K	WV	Johnson
Y	85K	IL	Clinton
⋮	⋮	⋮	⋮
Y	1050K	NY	Trump
N	35K	CA	Trump
?	**100K**	**NY**	**?**

Suppose a voter does not reveal whether or not they voted in 2012

For now, let's assume we had no missing entries during training

NBC: Prediction for Examples With Missing Data

The prediction rule in a generative model is

$$p(y = c \mid \mathbf{x}_{\text{new}}, \boldsymbol{\theta}) = \frac{p(y = c \mid \boldsymbol{\theta}) \cdot p(\mathbf{x}_{\text{new}} \mid y = c, \boldsymbol{\theta})}{\sum_{c'=1}^{C} p(y = c' | \boldsymbol{\theta}) p(\mathbf{x}_{\text{new}} \mid y = c', \boldsymbol{\theta})}$$

Let us suppose our datapoint is $\mathbf{x}_{\text{new}} = (?, x_2, \dots, x_D)$, ***e.g.,*** $(?, \text{100K}, \text{NY})$

$$p(y = c \mid \mathbf{x}_{\text{new}}, \boldsymbol{\theta}) = \frac{\pi_c \cdot \prod_{j=1}^{D} p(x_j \mid y = c, \boldsymbol{\theta}_{cj})}{\sum_{c'=1}^{C} p(y = c' | \boldsymbol{\theta}) \prod_{j=1}^{D} p(x_j \mid y = c', \boldsymbol{\theta}_{jc})}$$

Since x_1 is missing, we can marginalise it out,

$$p(y = c \mid \mathbf{x}_{\text{new}}, \boldsymbol{\theta}) = \frac{\pi_c \cdot \prod_{j=2}^{D} p(x_j \mid y = c, \boldsymbol{\theta}_{cj})}{\sum_{c'=1}^{C} p(y = c' | \boldsymbol{\theta}) \prod_{j=2}^{D} p(x_j \mid y = c', \boldsymbol{\theta}_{jc})}$$

This can be done for other generative models, but marginalisation is requires summation/integration

NBC: Training With Missing Data

For Naïve Bayes Classifiers, training with missing entries is quite easy

Voted in 2012?	Annual Income	State	Candidate Choice
?	50K	OK	Clinton
N	173K	CA	Clinton
?	80K	NJ	Trump
Y	150K	WA	Clinton
N	25K	WV	Johnson
Y	85K	?	Clinton
⋮	⋮	⋮	⋮
Y	1050K	NY	Trump
N	35K	CA	Trump
?	**100K**	**NY**	**?**

Let's say for Clinton voters, 103 had voted in 2012, 54 had not, and 25, didn't answer

You can simply set $\theta = \frac{103}{157}$ as the probability that a voter had voted in 2012, conditioned on being a Clinton supporter

Outline

Generative Model: Gaussian Discriminant Analysis

Recall the form of the joint distribution in a generative model

$$p(\mathbf{x}, y \mid \boldsymbol{\theta}, \boldsymbol{\pi}) = p(y \mid \boldsymbol{\pi}) \cdot p(\mathbf{x} \mid y, \boldsymbol{\theta})$$

For classes, we use parameters π_c such that $\sum_c \pi_c = 1$

$$p(y = c \mid \boldsymbol{\pi}) = \pi_c$$

Suppose $\mathbf{x} \in \mathbb{R}^D$, we model the class-conditional density for class $c = 1, \ldots, C$, as a multivariate normal distribution with mean $\boldsymbol{\mu}_c$ and covariance matrix $\boldsymbol{\Sigma}_c$

$$p(\mathbf{x} \mid y = c, \boldsymbol{\theta}_c) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

Quadratic Discriminant Analysis (QDA)

Let's first see what the prediction rule for this model is:

$$p(y = c \mid \mathbf{x}_{\text{new}}, \boldsymbol{\theta}) = \frac{p(y = c \mid \boldsymbol{\theta}) \cdot p(\mathbf{x}_{\text{new}} \mid y = c, \boldsymbol{\theta})}{\sum_{c'=1}^{C} p(y = c' | \boldsymbol{\theta}) p(\mathbf{x}_{\text{new}} \mid y = c', \boldsymbol{\theta})}$$

When the densities $p(\mathbf{x} \mid y = c, \boldsymbol{\theta}_c)$ are multivariate normal, we get

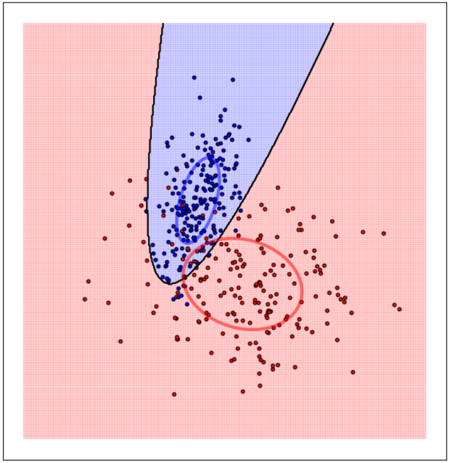
$$p(y = c \mid \mathbf{x}, \boldsymbol{\theta}) = \frac{\pi_c |2\pi \boldsymbol{\Sigma}_c|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^{\mathsf{T}} \boldsymbol{\Sigma}_c^{-1} (\mathbf{x} - \boldsymbol{\mu}_c)\right)}{\sum_{c'=1}^{C} \pi_{c'} |2\pi \boldsymbol{\Sigma}_{c'}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{c'})^{\mathsf{T}} \boldsymbol{\Sigma}_{c'}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{c'})\right)}$$

The denominator is the same for all classes, so the boundary between class c and c' is given by

$$\frac{\pi_c |2\pi \boldsymbol{\Sigma}_c|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^{\mathsf{T}} \boldsymbol{\Sigma}_c^{-1} (\mathbf{x} - \boldsymbol{\mu}_c)\right)}{\pi_{c'} |2\pi \boldsymbol{\Sigma}_{c'}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{c'})^{\mathsf{T}} \boldsymbol{\Sigma}_{c'}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{c'})\right)} = 1$$

Thus the boundaries are quadratic surfaces, hence the method is called quadratic discriminant analysis

Quadratic Discriminant Analysis (QDA)



Linear Discriminant Analysis

A special case is when the covariance matrices are <u>shared</u> or <u>tied</u> across different classes

We can write

$$p(y = c \mid \mathbf{x}, \boldsymbol{\theta}) \propto \pi_c \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_c)\right)$$
$$= \exp\left(\boldsymbol{\mu}_c^\mathsf{T} \boldsymbol{\Sigma}^{-1} \mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_c^\mathsf{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c + \log \pi_c\right) \cdot \exp\left(-\frac{1}{2}\mathbf{x}^\mathsf{T} \boldsymbol{\Sigma}^{-1} \mathbf{x}\right)$$

Let us set

$$\gamma_c = -\frac{1}{2}\boldsymbol{\mu}_c^\mathsf{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c + \log \pi_c \qquad\qquad \boldsymbol{\beta}_c = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c$$

and so

$$p(y = c \mid \mathbf{x}_{\text{new}}, \boldsymbol{\theta}) \propto \exp\left(\boldsymbol{\beta}_c^\mathsf{T} \mathbf{x} + \gamma_c\right)$$

Linear Discriminant Analysis (LDA) & Softmax

Recall that we wrote,

$$p(y = c \mid \mathbf{x}, \boldsymbol{\theta}) \propto \exp\left(\boldsymbol{\beta}_c^\mathsf{T}\mathbf{x} + \gamma_c\right)$$

And so,

$$p(y = c \mid \mathbf{x}, \boldsymbol{\theta}) = \frac{\exp\left(\boldsymbol{\beta}_c^\mathsf{T}\mathbf{x} + \gamma_c\right)}{\sum_{c'} \exp\left(\boldsymbol{\beta}_{c'}^\mathsf{T}\mathbf{x} + \gamma_{c'}\right)} = \mathrm{softmax}(\boldsymbol{\eta})_c$$

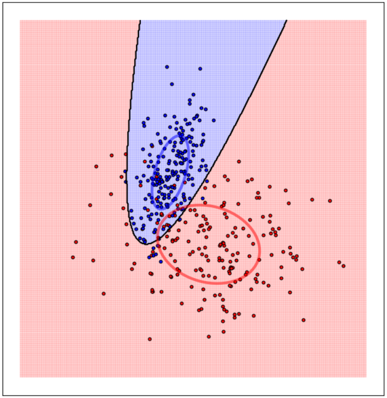
where, $\boldsymbol{\eta} = [\boldsymbol{\beta}_1^\mathsf{T}\mathbf{x} + \gamma_1, \cdots, \boldsymbol{\beta}_C^\mathsf{T}\mathbf{x} + \gamma_C]$.

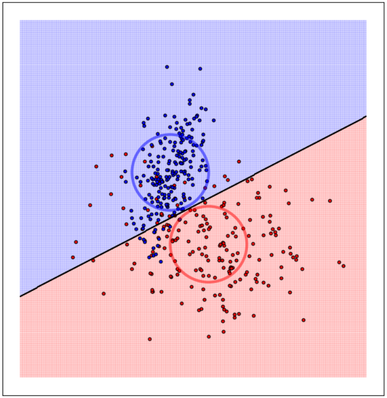
Softmax

Softmax maps a set of numbers to a probability distribution with mode at the maximum

$$\mathrm{softmax}([1, 2, 3]) \approx [0.090, 0.245, 0.665]$$
$$\mathrm{softmax}([10, 20, 30]) \approx [2 \times 10^{-9}, 4 \times 10^{-5}, 1]$$

QDA and LDA





Two class LDA

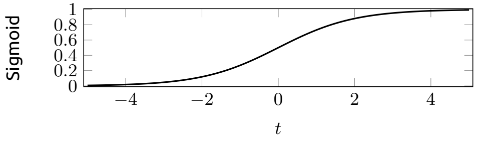
When we have only 2 classes, say 0 and 1,

$$p(y = 1 \mid \mathbf{x}, \theta) = \frac{\exp\left(\boldsymbol{\beta}_1^{\mathsf{T}}\mathbf{x} + \gamma_1\right)}{\exp\left(\boldsymbol{\beta}_1^{\mathsf{T}}\mathbf{x} + \gamma_1\right) + \exp\left(\boldsymbol{\beta}_0^{\mathsf{T}}\mathbf{x} + \gamma_0\right)}$$
$$= \frac{1}{1 + \exp\left(-((\boldsymbol{\beta}_1 - \boldsymbol{\beta}_0)^{\mathsf{T}}\mathbf{x} + (\gamma_1 - \gamma_0))\right)}$$
$$= \mathrm{sigmoid}((\boldsymbol{\beta}_1 - \boldsymbol{\beta}_0)^{\mathsf{T}}\mathbf{x} + (\gamma_1 - \gamma_0))$$

Sigmoid Function

The sigmoid function is defined as:

$$\mathrm{sigmoid}(t) = \frac{1}{1 + e^{-t}}$$



MLE for QDA (or LDA)

We can write the log-likelihood given data $\mathcal{D} = \langle (\mathbf{x}_i, y_i) \rangle_{i=1}^{N}$ as:

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{c=1}^{C} \left(\sum_{i:y_i=c} \log \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \right)$$

As in the case of Naïve Bayes, we get $\pi_c = \frac{N_c}{N}$

For other parameters, it is possible to show that,

$$\widehat{\boldsymbol{\mu}}_c = \frac{1}{N_c} \sum_{i:y_i=c} \mathbf{x}_i$$

$$\widehat{\boldsymbol{\Sigma}}_c = \frac{1}{N_c} \sum_{i:y_i=c} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_c)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_c)^{\mathsf{T}}$$

(See Chap 4.1 from Murphy for details)

How to Prevent Overfitting

- The number of parameters in the model is roughly $C \cdot D^2$
- In high-dimensions this can lead to overfitting
- Use diagonal covariance matrices (basically Naïve Bayes)
- Use weight tying a.k.a. parameter sharing (LDA vs QDA)
- Bayesian Approaches
- Use a discriminative classifier (+ regularize if needed)