Foundations of Data Science, Fall 2020

9. Generative Models for Classification

Prof. Dan Olteanu

DaST Data • (Systems+Theory)

University of Zurich[UZH]

Oct 27, 2020

https://lms.uzh.ch/url/RepositoryEntry/16830890400

https://uzh.zoom.us/j/96690150974?pwd=cnZmMTduWUtCeWoxYW85Z3RMYnpTZz09

Supervised Learning: Regression vs Classification

The input vector $\mathbf{x} = [x_1, \ldots, x_D]^\mathsf{T}$ consists of categorical and continuous features

- Categorical: $x_i \in \{1, \ldots, K\}$

  Often mapped to real values using, e.g., one-hot encoding

- Continuous: $x_i \in \mathbb{R}$

The target/label/output y

- Classification: y is a category

$$y \in \{1, \ldots, C\}$$

- Regression: y is a real value

$$y \in \mathbb{R}$$

Our focus today: Classification

Discriminative vs Generative Models

Discriminative setting: Model the conditional distribution of the output y given the input $\mathbf{x}$ and the model parameters $\boldsymbol{\theta}$

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

- Our focus so far: Discriminative Linear Model (with Gaussian noise)

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

- Other noise models possible, *e.g.,* Laplace
- Non-linearities using basis expansion
- Regularisation to avoid overfitting: Ridge, Lasso
- Validation to choose hyper-parameters
- Optimisation algorithms for model fitting

Generative setting: Model the full joint distribution of input $\mathbf{x}$ and output y given the model parameters $\boldsymbol{\theta}$

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

Our focus today: Generative models for classification

Predicting using Generative Classification Models

Given generative model $p(\mathbf{x}, y \mid \boldsymbol{\theta})$ representing the joint distribution of $\mathbf{x}$ and y

For a new input $\mathbf{x}_{\text{new}}$, the conditional distribution for y is

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

where $c \in \{1, \ldots, C\}$ are the possible categories/classes for y.

- Numerator is the joint probability distribution $p(\mathbf{x}_{\text{new}}, y = c \mid \boldsymbol{\theta})$
- Denominator is the marginal distribution $p(\mathbf{x}_{\text{new}} \mid \boldsymbol{\theta})$

To predict the most likely class: $\widehat{y} = \operatorname{argmax}_c p(y = c \mid \mathbf{x}_{\text{new}}, \boldsymbol{\theta})$

Defining a Generative Classification Model

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

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

- $p(y \mid \boldsymbol{\pi})$: marginal distribution of outputs y parameterised by $\boldsymbol{\pi}$

  We use parameters π_c such that $\sum_c \pi_c = 1$ and model the distribution as

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

- $p(\mathbf{x} \mid y, \boldsymbol{\theta})$: class-conditional distributions of input $\mathbf{x}$ given the class label y, parameterised by $\boldsymbol{\theta}$

  For each class $c = 1, \ldots, C$, we have the model

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

Toy Example: Dataset

Predict voter preference using in US elections

Voted in 2016?	Annual Income	State	Candidate Choice
Y	50K	OK	Beyoncé
N	173K	CA	Beyoncé
Y	80K	NJ	Borat
Y	150K	WA	Beyoncé
N	25K	WV	Kanye West
Y	85K	IL	Beyoncé
⋮	⋮	⋮	⋮
Y	1050K	NY	Borat
N	35K	CA	Borat
N	**100K**	**NY**	**?**

Toy Example: Classification

Marginal distribution for candidates

$$p(y = \text{Beyoncé} \mid \boldsymbol{\pi}) = \pi_{\text{Beyoncé}}$$
$$p(y = \text{Borat} \mid \boldsymbol{\pi}) = \pi_{\text{Borat}}$$
$$p(y = \text{Kanye West} \mid \boldsymbol{\pi}) = \pi_{\text{Kanye West}}$$

Given that a voter supports Borat

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

models the (class-conditional) distribution over $\mathbf{x}$ given $y = \text{Borat}$ and $\boldsymbol{\theta}_{\text{Borat}}$

Similarly for $p(\mathbf{x} \mid y = \text{Beyoncé}, \boldsymbol{\theta}_{\text{Beyoncé}})$ and $p(\mathbf{x} \mid y = \text{Kanye West}, \boldsymbol{\theta}_{\text{Kanye West}})$

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

To define the model, we estimate the parameters $\pi_c, \boldsymbol{\theta}_c$ for $c \in \{1, \ldots, C\}$

Marginal Distribution of the Target Classes via Maximum Likelihood

Assumptions:

- Dataset $\mathcal{D} = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N))$, where each vector $\mathbf{x}_i$ has dimension D
- Each data point is drawn independently from the same distribution $p(\mathbf{x}, y)$
- Let N_c be the number of data points with $y_i = c$, so that $\sum_{c=1}^{C} N_c = N$

The probability for a single data point $(\mathbf{x}_i, y_i)$ is:

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

The likelihood and the log-likelihood of the data are:

$$p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{\pi}) = \prod_{i=1}^{N} p(\mathbf{x}_i, y_i \mid \boldsymbol{\theta}, \boldsymbol{\pi}) = \prod_{i=1}^{N} \left(\prod_{c=1}^{C} \pi_c^{\mathbb{I}(y_i = c)} \cdot p(\mathbf{x}_i \mid y_i, \boldsymbol{\theta}) \right)$$

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{i=1}^{N} \log p(\mathbf{x}_i \mid y_i, \boldsymbol{\theta})$$

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

Maximum Log-Likelihood Estimator for Class Probability Distribution

We have the log-likelihood for the NBC

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{\pi}) = \underbrace{\sum_{c=1}^{C} N_c \log \pi_c}_{\text{dependent on } \boldsymbol{\pi}} + \underbrace{\sum_{i=1}^{N} \log p(\mathbf{x}_i \mid y_i, \boldsymbol{\theta})}_{\text{independent of } \boldsymbol{\pi}}$$

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

$$\begin{aligned} \text{maximize} \quad & \sum_{c=1}^{C} N_c \log \pi_c \\ \text{subject to :} \quad & \sum_{c=1}^{C} \pi_c = 1 \end{aligned}$$

This problem can be solved using the method of Lagrange multipliers

Recall: Constrained Optimisation Problems using Lagrange Multipliers

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

Constrained Objective

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

Lagrangian (Dual) Form

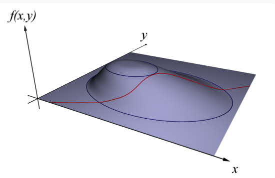
$$\Lambda(\mathbf{z}, \lambda) = f(\mathbf{z}) - \lambda g(\mathbf{z})$$

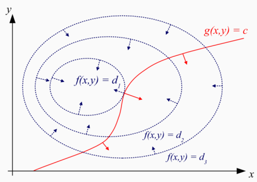
Any optimal solution to the constrained problem is a stationary point of $\Lambda(\mathbf{z}, \lambda)$

Recall: Constrained Optimisation Problems using Lagrange Multipliers

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

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





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

Back to Our Constrained Optimisation Problem

Recall that we want to solve:

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

The Lagrangian formulation of our problem:

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

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

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

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

Summing Up: Marginal Distribution over the Classes

For Generative Classification Models,

the marginal distribution $p(y \mid \pi)$ over the classes is

the empirical distribution over the classes.

Remaining challenges:

- Choose suitable model for the class-conditional distributions $p(\mathbf{x} \mid y, \boldsymbol{\theta})$
- Estimate the parameters $\boldsymbol{\theta}$ of the model

We next consider these challenges for several classification models

- Naïve Bayes
- Gaussian (Quadratic/Linear) Discriminant Analysis

Naïve Bayes Classifier (NBC)

Assume that the features are conditionally independent given the class label c

The *state*, *previous voting* or *annual income* are conditionally independent on being a Beyoncé/Borat/Kanye West supporter

Clearly, this assumption is "naïve" and never satisfied :(

But model fitting becomes 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)

Assume that the features are conditionally independent given the class label c

Class-conditional distribution for data point $(\mathbf{x}_i, y_i)$ becomes:

$$p(\mathbf{x}_i \mid y_i = c, \boldsymbol{\theta}) = p(\mathbf{x}_i \mid y_i = c, \boldsymbol{\theta}_c) = \prod_{j=1}^{D} p(x_{ij} \mid \boldsymbol{\theta}_{jc})$$

The joint probability distribution given the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$ becomes:

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

Log-likelihood for the entire dataset $\mathcal{D}$ in case of NBC becomes:

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

MLE for parameters π_c obtained as before.

Naïve Bayes Classifier: Models for Individual Features

Easy to mix and match different models for different features

Real-Valued Features

- x_j is real-valued *e.g.,* annual income
- Gaussian model: $\boldsymbol{\theta}_{jc} = (\mu_{jc}, \sigma^2_{jc})$

  MLE for $\boldsymbol{\theta}_{jc}$ given by empirical mean and variance
- Can use other distributions, *e.g.,* Laplace

Categorical Features

- x_j is categorical with categories $1, \ldots, K$
- Multinoulli model: $x_j = \ell$ with probability $\theta_{jc,\ell}$ such that $\sum_{\ell=1}^{K} \theta_{jc,\ell} = 1$

  MLE obtained similarly to that for π_c
- The special case when $x_j \in \{0, 1\}$ needs a single parameter $\theta_{jc} \in [0, 1]$
- No need to convert categorical variables to real-valued vectors!

Naïve Bayes Classifier: Number of Parameters

Assumptions:

- All the D features are binary, *i.e.,* every $x_j \in \{0, 1\}$, $j \in [D]$
- There are C target classes

With the naïve conditional independence assumption

- Naïve Bayes has at most $C \cdot D$ parameters $(\theta_{jc})_{j \in [D], c \in [C]}$

Without the conditional independence assumption

- We have to assign a probability for each of the 2^D possible feature vectors
- Thus, we have $C \cdot 2^D$ parameters!
- Either overfits if $N \approx C \cdot 2^D$ or computationally prohibitive if $N \gg C \cdot 2^D$

The 'naïve' assumption breaks the *curse of dimensionality* and avoids overfitting!

NBC: Prediction for Examples With Missing Data

Recall the prediction rule in a generative model:

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

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

Assume our data point is $\mathbf{x}_{\text{new}} = (?, x_2, \ldots, x_D)$, *e.g.,* $(?, 100\text{K}, \text{NY})$

Since x_1 is missing, we skip it:

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

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

NBC: Training With Missing Data

For Naïve Bayes Classifiers, training with missing entries is easy

Voted in 2016?	Annual Income	State	Candidate Choice
Y	50K	OK	Beyoncé
N	173K	CA	Beyoncé
Y	80K	NJ	Borat
Y	150K	WA	Beyoncé
N	25K	WV	Kanye West
Y	85K	IL	Beyoncé
⋮	⋮	⋮	⋮
Y	1050K	NY	Borat
N	35K	CA	Borat
?	**100K**	**NY**	**?**

Assume for Beyoncé voters, 103 had voted in 2016, 54 had not, and 25 didn't answer

We can set $\theta = \frac{103}{103+54}$ as the probability that a voter had voted in 2016, conditioned on being a Beyoncé supporter

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 $\sum_c \pi_c = 1$

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

We model the class-conditional density for class $c \in \{1, \ldots, C\}$, as a multivariate normal distribution with mean $\boldsymbol{\mu}_c$ and covariance matrix $\boldsymbol{\Sigma}_c$

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

Note: Name 'discriminant' unfortunate as this is a *generative* model.

It refers to the shape of the boundaries that discriminate between the classes.

More suitable than NBC in some cases:

Predict zebra or giraffe based on animal height and weight

MLE for Gaussian Discriminant Analysis

Given data $\mathcal{D} = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N))$, the log-likelihood is:

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

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

For parameters $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$, we have (Section 4.1 from Murphy)

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

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

Quadratic Discriminant Analysis (QDA)

The prediction rule for this model is:

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

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

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

The denominator is the same for all classes $c' \in \{1, \ldots, C\}$.

Quadratic Discriminant Analysis: Decision Boundaries are Quadratic Curves

The boundary between classes c_1 and c_2 is given by the data points $\mathbf{x}$ with

$$p(y = c_1 \mid \mathbf{x}, \boldsymbol{\theta}) = p(y = c_2 \mid \mathbf{x}, \boldsymbol{\theta})$$

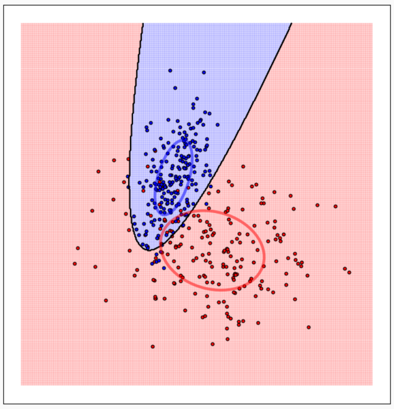
This means:

$$\frac{\pi_{c_1} |2\pi \boldsymbol{\Sigma}_{c_1}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{c_1})^{\mathsf{T}} \boldsymbol{\Sigma}_{c_1}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{c_1})\right)}{\pi_{c_2} |2\pi \boldsymbol{\Sigma}_{c_2}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{c_2})^{\mathsf{T}} \boldsymbol{\Sigma}_{c_2}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{c_2})\right)} = 1$$

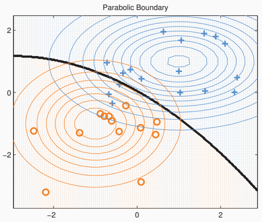
By taking the log:

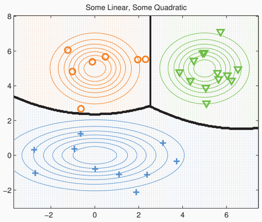
$$\frac{1}{2}\left((\mathbf{x} - \boldsymbol{\mu}_{c_2})^{\mathsf{T}} \boldsymbol{\Sigma}_{c_2}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{c_2}) - (\mathbf{x} - \boldsymbol{\mu}_{c_1})^{\mathsf{T}} \boldsymbol{\Sigma}_{c_1}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{c_1})\right) = \log \left(\frac{\pi_{c_2} |2\pi \boldsymbol{\Sigma}_{c_2}|^{-\frac{1}{2}}}{\pi_{c_1} |2\pi \boldsymbol{\Sigma}_{c_1}|^{-\frac{1}{2}}} \right)$$

Quadratic Discriminant Analysis: Decision Boundary for Two Classes



Quadratic Discriminant Analysis: Quadratic and Linear Decision Boundaries





Not every point $\mathbf{x}$ satisfying

$$p(y = c_1 \mid \mathbf{x}, \boldsymbol{\theta}) = p(y = c_2 \mid \mathbf{x}, \boldsymbol{\theta})$$

lies on the decision boundary between the classes c_1 and c_2!

Point $\mathbf{x}$ belongs to class c_3 if $p(y = c_3 \mid \mathbf{x}, \boldsymbol{\theta}) > p(y = c_i \mid \mathbf{x}, \boldsymbol{\theta}), i \in \{1, 2\}$

Boundaries are given by piecewise quadratic curves

Linear Discriminant Analysis (LDA)

Special case: The covariance matrices are shared or tied across different classes

$$p(y = c \mid \mathbf{x}, \boldsymbol{\theta}) = \frac{\pi_c |2\pi\boldsymbol{\Sigma}_c|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-1}(\mathbf{x} - \boldsymbol{\mu}_c)\right)}{\sum_{c'=1}^{C} \pi_{c'} |2\pi\boldsymbol{\Sigma}_{c'}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{c'})^\top \boldsymbol{\Sigma}_{c'}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{c'})\right)}$$

becomes

$$p(y = c \mid \mathbf{x}, \boldsymbol{\theta}) = \frac{\pi_c |2\pi\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_c)\right)}{\sum_{c'=1}^{C} \pi_{c'} |2\pi\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{c'})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{c'})\right)}$$

$$= \frac{\pi_c \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_c)\right)}{\sum_{c'=1}^{C} \pi_{c'} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{c'})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{c'})\right)}$$

Linear Discriminant Analysis: Further Simplifications

Simplify the numerator:

$$\pi_c \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_c)\right)$$

$$= \exp\left(\underbrace{\boldsymbol{\mu}_c^\top \boldsymbol{\Sigma}^{-1}}_{\boldsymbol{\beta}_c^\top} \mathbf{x} \underbrace{- \frac{1}{2}\boldsymbol{\mu}_c^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c + \log \pi_c}_{\gamma_c}\right) \cdot \exp\left(-\frac{1}{2}\mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\mathbf{x}\right)$$

$$= \exp\left(\boldsymbol{\beta}_c^\top \mathbf{x} + \gamma_c\right) \cdot \exp\left(-\frac{1}{2}\mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\mathbf{x}\right)$$

After expansion, the term $\exp\left(-\frac{1}{2}\mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\mathbf{x}\right)$ also appears at denominator.

$$p(y = c \mid \mathbf{x}, \boldsymbol{\theta}) = \frac{\exp\left(\boldsymbol{\beta}_c^\top \mathbf{x} + \gamma_c\right)}{\sum_{c'} \exp\left(\boldsymbol{\beta}_{c'}^\top \mathbf{x} + \gamma_{c'}\right)} = \mathrm{softmax}(\boldsymbol{\eta})_c$$

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

Softmax Function

$$\mathrm{softmax}([z_1, \ldots, z_n])_i = \frac{e^{z_i}}{\sum_{j \in [n]} e^{z_j}}$$

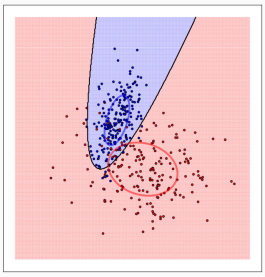
Normalises the vector values into a probability distribution consisting of probabilities proportional to their exponentials.

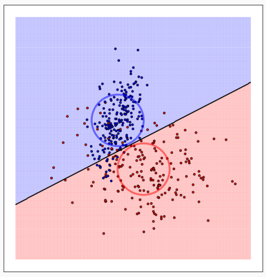
- Smooth approximation of argmax and not max!
- Used in statistical mechanics as the Boltzmann distribution
- Large inputs $\rightarrow$ large probabilities
- Translation invariant: Multiplying all values by the same quantity does not change the ranking

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

- Not scale invariant: De-emphasises the max value for values $\in [0, 1]$
- Often used as the last activation function of a neural network to normalise the network output to a probability distribution over predicted output classes

QDA and LDA





Two-Class LDA

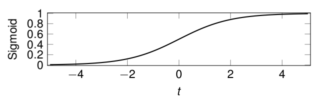
When we have only 2 classes, say 0 and 1:

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

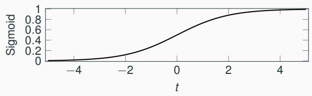
Sigmoid Function

The sigmoid function is defined as:

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



The Sigmoid Functions



- In general: Functions with S-shaped curves
- Examples: logistic (previous slide), hyperbolic tangent
- Logistic function: special case of softmax for 1D axis in 2D space
- Properties:
  - differentiable
  - non-negative derivative at each point
  - monotonic
  - convex for $t < 0$ and concave for $t > 0$

How to Prevent Overfitting

- The number of parameters in QDA is roughly $C \cdot D^2$
- In high-dimensions this can lead to overfitting
- Use diagonal covariance matrices – this corresponds to Naïve Bayes!
- Use weight tying/parameter sharing: LDA vs QDA
- Use a discriminative classifier (+ regularisation if needed)