Pattern Analysis – PA [2019]

Prof. Juan Rafael Orozco-Arroyave <rafael.orozco@udea.edu.co>
GITA Research Lab
School of Engineering
University of Antioquia

- Note 1: These slides are for a personal use only in order to prepare for the examination.
- Note 2: Publication, reproduction, and distribution of this material is not permitted without prior written approval.
- Note 3: These slides are mainly based on the lecture of Pattern Analysis of the Lehrstuhl für Mustererkennung (LME) at the Friedrich-Alexander-Universtität Erlangen-Nürnberg by Prof. Dr.-Ing. Joachim Hornegger and Dr.-Ing. Christian Riess.
- Note 4: Part of these slides are also based on the lecture of Introduction to Pattern Recognition by Dr.-Ing. Stefan Steidl.

Topics to be covered:

1. Introduction and analytic feature extraction methods: PCA and SVD
2. Bayesian Classifier
3. Logistic Regression
4. Naïve and Gaussian Classifiers
5. Linear Discriminant Analysis and Fisher Transform
6. Linear Regression
7. Norms and Norm Dependent Linear Regression
8. Optimization
9. Rosenblatt's Perceptron
10. Unconstrained optimization
11. Support Vector Machines (SVM)
12. Soft Margin SVM
13. Regression using SVM
14. Kernels
15. Kernel PCA
16. The Expectation Maximization Algorithm
17. Gaussian Mixture Models

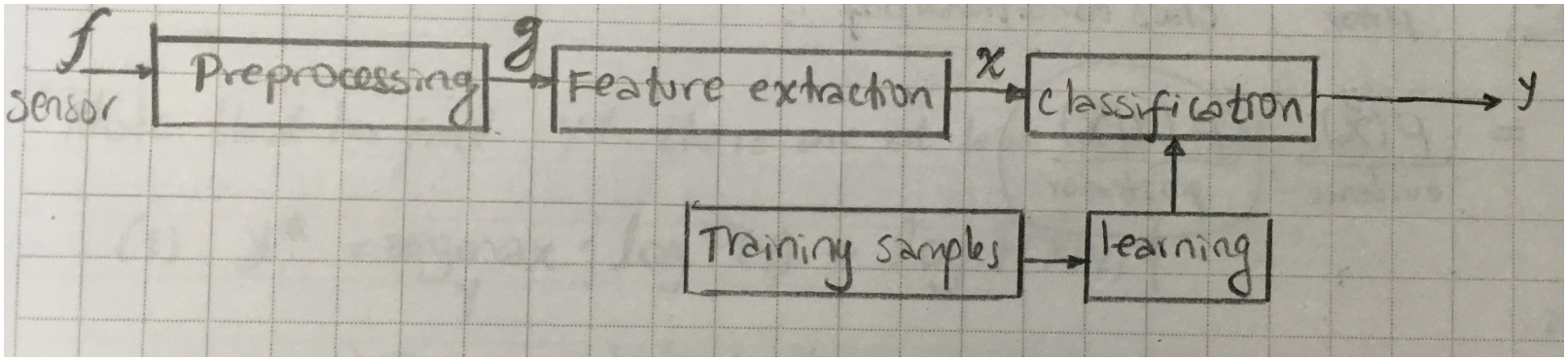
Examination

- There could be short written exams, not planned in advance [20 %].
- There will be an oral exam during the last two/three weeks of the term (exact dates to be defined) [60 %]
- There will be home-works (practical and theoretical) assigned during the term [20 %].

Pattern Analysis: methods and systems that automatically recognize patterns in sensed data.

Postulates of Pattern Recognition (PR)

- Sample set: we have representative samples per class.
- Features: a pattern has features that are characteristic for a certain class that the pattern belongs to.
- Compactness: features of patterns occupy a compact area in the feature space.
- Decomposition: a complex pattern can be decomposed into smaller parts.
- Structure: complex patterns of a specific domain have a certain structure.
- Similarity: two patterns are similar if their features or their components differ only lightly.



Analytic Feature Extraction Methods

Idea: Construction of feature vectors to support the PR postulates.

Approach: find a linear transformation $\mathbf{\Phi} : \mathbf{g} \rightarrow \mathbf{x}$ that maps the pattern $\mathbf{g} \in \mathbb{R}^d$ to a feature vector $\mathbf{x} \in \mathbb{R}^{d'}$, so that an optimality criterion is satisfied.
Dimensionality reduction: $d' \leq d$, ideally $d'' \ll d$
Problem: computation of $\mathbf{\Phi}$ such that the resulting features $\mathbf{x}$ optimize a quality criterion.

Principal Component Analysis

Goal 1: to find the features $\mathbf{x}$ that better describe the variation of the original data.

Goal 2: maximize the square distance S_1 between all pairs of feature vectors:

$$S_1 = \sum_{i=1}^{m}\sum_{j=1}^{m}(\mathbf{x_i}-\mathbf{x_j})^{\mathbf{T}}(\mathbf{x_i}-\mathbf{x_j}) = \sum_{i=1}^{m}\sum_{j=1}^{m}(\mathbf{\Phi g_i}-\mathbf{\Phi g_j})^{\mathbf{T}}(\mathbf{\Phi g_i}-\mathbf{\Phi g_j})$$

m is the number of samples (data points).
Trivial solution: when $\phi \rightarrow \infty$

Idea: bind components of $\mathbf{\Phi}$ to be within a certain range.
Constraint: the Frobenius of $\mathbf{\Phi}$ (Euclidean norm):

$$||\mathbf{\Phi}||_F = \sqrt{\sum_{j=1}^{d'}\sum_{j=1}^{d}|\phi_{ij}|^2} = 1$$

Constraint optimization problem $\rightarrow$ Lagrange multiplier method.

$$\hat{\mathbf{\Phi}} = \underset{\mathbf{\Phi}}{\operatorname{argmax}} \underbrace{\sum_{i=1}^{m}\sum_{j=1}^{m}(\mathbf{\Phi g_i} - \mathbf{\Phi g_j})^{\mathbf{T}}(\mathbf{\Phi g_i} - \mathbf{\Phi g_j})}_{S_1} - \lambda(||\mathbf{\Phi}||_F - 1)$$

Let's try to simplify this equation:

$$S_1 = \sum_{i=1}^{m}\sum_{j=1}^{m}(\mathbf{\Phi g_i} - \mathbf{\Phi g_j})^{\mathbf{T}}(\mathbf{\Phi g_i} - \mathbf{\Phi g_j})$$

factoring out $\mathbf{\Phi}$;

$$S_1 = \sum_{i=1}^{m}\sum_{j=1}^{m}((\mathbf{\Phi}(\mathbf{g_i} - \mathbf{g_j}))^{\mathbf{T}}\mathbf{\Phi}(\mathbf{g_i} - \mathbf{g_j})$$

but in general $(AB)^T = B^T A^T$, then:

$$S_1 = \sum_{i=1}^{m}\sum_{j=1}^{m}\underbrace{(\mathbf{g_i} - \mathbf{g_j})^{\mathbf{T}}}_{\mathbf{g_{ij}^T}}\mathbf{\Phi^T\Phi}\underbrace{(\mathbf{g_i} - \mathbf{g_j})}_{\mathbf{g_{ij}}}$$

Additionally, by construction, $\mathbf{\Phi^T \Phi}$ is a symmetric matrix.
$\Rightarrow$ Property of symmetric matrices $\mathbf{A^T B C} = \mathrm{Tr}\left(\mathbf{B A C^T}\right)$ (Tr is the trace operator).
And two properties of the trace are: $\mathrm{Tr}(\mathbf{A}) = \mathrm{Tr}(\mathbf{A^T}) = \sum_i a_{ii}$ and $\mathrm{Tr}(\mathbf{ABC}) = \mathrm{Tr}(\mathbf{BAC}) = \mathrm{Tr}(\mathbf{BCA})$

$\therefore$

$$S_1 = \sum_{i=1}^{m}\sum_{j=1}^{m} \mathbf{g_{ij}^T} \underbrace{\Phi^T \Phi}_{\mathbf{B}} \mathbf{g_{ij}} = \sum_{\mathbf{i=1}}^{\mathbf{m}}\sum_{\mathbf{j=1}}^{\mathbf{m}} \mathrm{Tr}(\mathbf{\Phi^T \Phi g_{ij} g_{ij}^T})$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{m} \mathrm{Tr}(\underbrace{\mathbf{g_{ij} g_{ij}^T}}_{\mathbf{M_{ij}}} \mathbf{\Phi^T \Phi})$$

$\mathbf{M_{ij}}$: Measured matrix.

Since we are looking for a decomposition strategy, let's use the column vectors of $\mathbf{\Phi^T}$: $\mathbf{\Phi^T} = (\phi_\mathbf{1}, \phi_\mathbf{2}, \cdots, \phi_{\mathbf{d'}})$.

$$S_1 = \sum_{i=1}^{m}\sum_{j=1}^{m} \text{Tr}\left(\mathbf{M_{ij}\Phi^T\Phi}\right) = \sum_{i=1}^{m}\sum_{j=1}^{m} \text{Tr}\left(\mathbf{M_{ij}} \sum_{\mathbf{k=1}}^{\mathbf{d'}} \underbrace{\phi_k^T \phi_k}_{\mathbf{\Phi^T\Phi}}\right)$$

Remember: $\mathbf{A^TBC} = \text{Tr}(\mathbf{BAC^T})$

$\Rightarrow$

$$S_1 = \sum_{k=1}^{d'} \phi_\mathbf{k} \underbrace{\sum_{i=1}^{m}\sum_{j=1}^{m} \mathbf{M_{ij}}}_{\mathbf{Q}} \phi_\mathbf{k}^\mathbf{T} = \sum_{k=1}^{d'} \phi_\mathbf{k} \mathbf{Q} \phi_\mathbf{k}^\mathbf{T}$$

Where

$$\mathbf{Q} = \sum_{i=1}^{m}\sum_{j=1}^{m} \mathbf{M_{ij}} = \sum_{i=1}^{m}\sum_{j=1}^{m} (\mathbf{g_i} - \mathbf{g_j})(\mathbf{g_i} - \mathbf{g_j})^\mathbf{T}$$

Now we can formulate again the optimization problem:

$$\hat{\mathbf{\Phi}} = \underset{\mathbf{\Phi}}{\operatorname{argmax}} \sum_{k=1}^{d'} \phi_{\mathbf{k}}^{\mathbf{T}} \mathbf{Q} \phi_{\mathbf{k}} - \lambda \left(\sum_{k=1}^{d'} \phi_{\mathbf{k}}^{\mathbf{T}} \phi_{\mathbf{k}} - \mathbf{1} \right)$$

Now let's set the partial derivatives regarding $\phi_{\mathbf{k}}$ to zero:

$$\frac{\partial}{\partial \phi_{\mathbf{k}}} \left(\sum_{k=1}^{d'} \phi_{\mathbf{k}}^{\mathbf{T}} \mathbf{Q} \phi_{\mathbf{k}} \right) - \lambda \left(\sum_{k=1}^{d'} \phi_{\mathbf{k}}^{\mathbf{T}} \phi_{\mathbf{k}} - \mathbf{1} \right) = 0$$

$$2\mathbf{Q}\phi - 2\lambda\phi = 0$$

$\therefore$

$$\mathbf{Q}\phi = \lambda\phi$$

Which is the typical eigenvalue/eigenvector problem, where λ are the eigenvalues and ϕ are the eigenvectors.
Without loss of generality:

$$\mathbf{Q}\phi_{\mathbf{k}} = \lambda_k \phi_{\mathbf{k}}$$

PCA in summary

- Constraint optimization problem: to maximize the overall spread of the data samples for the constraint $||\mathbf{\Phi}||_F = 1$.

$$\hat{\mathbf{\Phi}} = \underset{\mathbf{\Phi}}{\operatorname{argmax}} \sum_{i=1}^{m}\sum_{j=1}^{m} (\mathbf{\Phi g_i} - \mathbf{\Phi g_j})^{\mathbf{T}}(\mathbf{\Phi g_i} - \mathbf{\Phi g_j}) - \lambda(||\mathbf{\Phi}||_F - 1)$$

- Set up the matrix $\mathbf{Q} = m^2\mathbf{\Sigma}_{\text{cov}}$

$$\mathbf{Q} = \underbrace{\sum_{i=1}^{m}\sum_{j=1}^{m}}_{m^2} \underbrace{(g_i - g_j)(g_i - g_j)^T}_{\mathbf{\Sigma}_{\text{cov}}}$$

- Solve the eigenvalue/eigenvector problem:

$$\mathbf{Q}\phi_{\mathbf{k}} = \lambda_k \phi_{\mathbf{k}}$$

- Sort the eigenvectors in descending order of their eigenvalues.
- The first d' eigenvectors are the rows of matrix $\mathbf{\Phi}$

NOTE 1: The transformed features are uncorrelated.

NOTE 2: The eigenvalue/eigenvector problem can be solved via Singular Value Decomposition (SVD).

Singular Value Decomposition (SVD)

Let's consider a data matrix:
$\mathbf{X} = [\mathbf{g_1} - \mu, \mathbf{g_2} - \mu, \mathbf{g_3} - \mu, \cdots, \mathbf{g_m} - \mu] \in \mathbb{R}^{d \times m}$

$\text{Rank}(\mathbf{X}) = r$: number of linearly independent columns.

The main fact is: *every* $d \times m$ *matrix can be factorized into* $\mathbf{X} = \mathbf{U\Sigma V^T}$

$[\mathbf{U}]_{d \times d}$: is an orthogonal matrix $\Rightarrow \mathbf{U^T U} = \mathbf{U U^T} = \mathbf{I}$ and $\mathbf{U^T} = \mathbf{U^{-1}}$

$[\mathbf{\Sigma}]_{d \times m}$: is a diagonal matrix $\Rightarrow$ entries outside its main diagonal are all zero.

$$\begin{bmatrix} \sigma_1 & 0 & 0 & \cdots 0 \\ 0 & \sigma_2 & 0 & \cdots 0 \\ 0 & 0 & \sigma_3 & \cdots 0 \\ 0 & 0 & 0 & \cdots \sigma_d \end{bmatrix}$$

σ_i are singular values.

When the matrix is not square but rectangular $(d \times m)$ it looks like $\Rightarrow$

$$\begin{bmatrix} \sigma_1 & 0 & 0 & \cdots 0 & 0 \\ 0 & \sigma_2 & 0 & \cdots 0 & 0 \\ 0 & 0 & \sigma_3 & \cdots 0 & 0 \end{bmatrix}$$

$[\mathbf{V^T}]_{m \times m}$ is an orthogonal matrix.

Physically $\mathbf{U\Sigma V^T}$ implies (rotation)(stretching)(rotation).

What is the difference between this decomposition $(\mathbf{X = U\Sigma V^T})$ and the decomposition that is performed to diagonalize a matrix $(\mathbf{PAP^{-1}})$?

- In SVD $\mathbf{U}$ and $\mathbf{V^T}$ are orthogonal matrices and we can do SVD for rectangular matrices in general. Eigenvalues really work for square matrices.
- In SVD we have two "spaces", one is $d-$dimensional (for $\mathbf{U}$), and the other one is $m-$dimensional (for $\mathbf{V^T}$).

What are $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V^T}$ such that $\mathbf{X} = \mathbf{U\Sigma V^T}$?

Let's assume wee need to find $\mathbf{X^T X}$.
$\mathbf{X^T X} = (\mathbf{V\Sigma^T U^T})(\mathbf{U\Sigma V^T})$
$\rightarrow$ remember $\mathbf{U^T U} = \mathbf{I}$ because $\mathbf{U}$ is orthogonal.

$$\Rightarrow \mathbf{X^T X} = \mathbf{V} \underbrace{(\Sigma^T \Sigma)}_{\mathbf{diagonal}} \mathbf{V^T}$$

When we have $\mathbf{X^T X} = \mathbf{V}(\mathbf{\Sigma^T \Sigma})\mathbf{V^T}$ it says:

- $\mathbf{V}$ is the eigenvectors matrix for $\mathbf{X^T X}$
- $\mathbf{\Sigma^T \Sigma}$ is the eigenvalues matrix for $\mathbf{X^T X}$

Now let's assume we need to find $\mathbf{XX^T}$:

$\Rightarrow \mathbf{XX^T} = (\mathbf{U\Sigma V^T})(\mathbf{V\Sigma^T U^T})$,

$\rightarrow \mathbf{V^T V} = \mathbf{I}$ because $\mathbf{V}$ is orthogonal.

$\Rightarrow \mathbf{XX^T} = \mathbf{U\Sigma\Sigma^T U^T}$

- $\mathbf{U}$ is the eigenvectors matrix for $\mathbf{XX^T}$
- $\mathbf{\Sigma\Sigma^T}$ is the eigenvalues matrix for $\mathbf{XX^T}$ which are the same for $\mathbf{X^T X}$

Note: when we find $\mathbf{u_1}\sigma_1\mathbf{v_1^T}$, the largest (actually the first one) sigma gives us the first eigenvector which keeps the greatest variance of $\mathbf{X}$.

Now let's come back to the PCA topic.

PCA can also be studied as a minimization of the mean square error.

Homework: please check it in the literature and write a report (hand-made).

Now let's see the concept of "learning"

Supervised learning: we have features and know which class they belong to.
m training samples with features and their associated classes.

$$\mathbf{S} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$$

where $x_i \in \chi$ are the features and $y_i \in \mathbb{Z}$ are the class numbers. Typically $\chi \in \mathbb{R}^d$.

Unsupervised learning: we have features but no information about the associated classes.
m training samples with only features $\Rightarrow \mathbf{S} = \{x_1, x_2, \cdots, x_m\}$

Bayesian Classifier

Notation:

- $\mathbf{x} \in \mathbb{R}^d$: $d-$dimensional feature vector.
- y: class number, usually $y \in \{0, 1\}$ or $y \in \{-1, +1\}$
- $p(y)$: prior probability of pattern class y. E.g., assume you are analyzing texts from The NYT and ET. $p(y)$ is the probability of finding a certain name (Juanes) in each journal.
- $p(\mathbf{x})$: Evidence $\rightarrow$ distribution of features in the $d-$dimensional feature space. $\Rightarrow$ this tells me what is the probability of observing a certain feature without class number.
- $p(\mathbf{x}, y)$: joint probability of having a certain feature and a class number.
- $p(\mathbf{x}|y)$: class conditional density (CCD). Density function for a given feature vector given a class.

  $\Rightarrow$ We use prior knowledge: the probability of having $\mathbf{x}$ if we have the class y.
- $p(y|\mathbf{x})$: Posterior probability. $\rightarrow$ What is the probability of the class y given the feature vector $\mathbf{x}$.

Bayes rule:

$$\underbrace{p(\mathbf{x}, y)}_{\text{joint pdf}} = \underbrace{p(y)}_{\text{prior}} \underbrace{p(\mathbf{x}|y)}_{\text{CCD}}$$

$$= \underbrace{p(\mathbf{x})}_{\text{evidence}} \underbrace{p(y|\mathbf{x})}_{\text{posterior}}$$

We can find the posteriors:

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

$$= \frac{p(y)p(\mathbf{x}|y)}{\sum_{y'} p(\mathbf{x}, y')}$$

$$= \frac{p(y)p(\mathbf{x}|y)}{\sum_{y'} p(y')p(\mathbf{x}|y')}$$

$p(\mathbf{x}) \rightarrow$ evidence is not necessary. We can find it via marginalization: if we have $p(\mathbf{x}, y)$ and we need $p(\mathbf{x})$, we can do $p(\mathbf{x}) = \sum_y p(\mathbf{x}, y)$.

Notes:

- $p(\mathbf{x}) = \sum_y p(y)p(\mathbf{x}|y)$ is a marginal of $p(\mathbf{x}, y)$
- We get $p(\mathbf{x})$ by marginalizing $p(\mathbf{x}, y)$ over y
- Similarly we can get $p(y)$ by marginalizing $p(\mathbf{x}, y)$ over $\mathbf{x}$, i.e., $p(y) = \int_{\mathbf{x}} p(\mathbf{x}, y)d\mathbf{x}$

Now let's consider the Bayesian decision rule.
We decide for the class y^* according to the decision rule, as follows:

$$y^* = \underset{y}{\operatorname{argmax}}\ p(y|\mathbf{x})$$

Evaluates all of the classes and decide for the class with the higher probability y^*

$$y^* = \underset{y}{\operatorname{argmax}}\ \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})}$$

$p(\mathbf{x})$ does not influence the maximum of y it is just a scale of the function.

$$y^* = \underset{y}{\operatorname{argmax}}\ p(y)p(\mathbf{x}|y)$$

These probabilities can get smaller and smaller due to multiplications. The $\log$ function changes the values but not the position of the maximum.

$$y^* = \underset{y}{\operatorname{argmax}}\ \{\underbrace{\log p(y)}_{\text{prior}} + \underbrace{\log p(\mathbf{x}|y)}_{\text{CCD}}\}$$

$\Rightarrow$ PA is all about modeling the posterior $p(y|\mathbf{x})$
$\Rightarrow$ in PA the set χ is not necessary a subset of $\mathbb{R}^d \rightarrow$ can be sequences or sets and can vary on dimension, longitude, etc.

Note that there are at least two strategies to find y^* in a Bayesian Classifier:

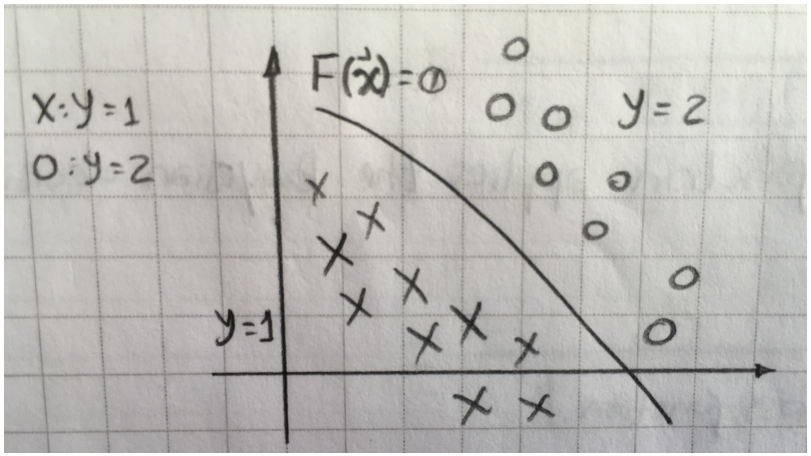
1. $y^* = \underset{y}{\text{argmax}} \{\log p(y) + \log p(\mathbf{x}|y)\}$

   $\Rightarrow$ to compute/find the priors and the class conditional probability.

2. $y^* = \underset{y}{\text{argmax}} \{p(y|\mathbf{x})\}$

Discriminative modeling

Generative modeling



$p(y|\mathbf{x}) =?$ what is the posterior probability assuming that we know that the decision boundary is such that $F(\mathbf{x}) = 0$.

We need to find the optimal decision boundary such that $F(\mathbf{x}) = 0$... there are many, but we need the optimal one!

Optimality of the Bayesian Classifier: why is it optimal?

Definition: $l(y_1, y_2)$ is the loss if a feature vector belonging to class y_2 is assigned to class y_1. The $(0-1)-$Loss function is defined by:

$$l(y_1, y_2) = \begin{cases} 0 & \text{if } y_1 = y_2 \\ 1 & \text{otherwise} \end{cases}$$

What is the best classifier? $\rightarrow$ the one that gives you the lowest loss.

The best (optimal) decision rule, according to a classification loss, minimizes the average loss (AL).

$$\text{AL}(\mathbf{x}, y) = \sum_{y'} l(y, y') p(y'|\mathbf{x})$$

which is the loss function weighted by the posterior probability.
y is the assigned class; y' is the original class; $p(y'|\mathbf{x})$ are the posteriors of the observed feature vectors and their associated class.

We decide for the class (y^*) that minimizes the loss:

$\Rightarrow$ using the $(0-1)-$Loss function, the class decision is based on:

$$y^* = \underset{y}{\operatorname{argmin}} \ \mathrm{AL}(\mathbf{x}, y)$$

$$y^* = \underset{y}{\operatorname{argmin}} \ \sum_{y'} l(y, y') p(y'|\mathbf{x}) \quad 1 \cdot p(y_1|\mathbf{x}) + 1 \cdot p(y_2|\mathbf{x}) + 0 \cdot p(\underbrace{y}_{y=y'}|\mathbf{x}) + \cdots + 1 \cdot p(y_i|\mathbf{x})$$

$$y^* = \underset{y}{\operatorname{argmin}} \ (1 - p(y|\mathbf{x}))$$

All posteriors except for the one that was chosen with $l(y, y') = 1$

$$y^* = \underset{y}{\operatorname{argmax}} \ p(y|\mathbf{x})$$

The Bayesian Classifier!!

Conclusion: The optimal classifier according to the $(0-1)-$Loss function applies the Bayesian decision rule.

$\Rightarrow$ The Bayesian Classifier! $\rightarrow$ posteriors $\&$ $(0-1)-$Loss function.

Logistic Regression

It is a generative model because it models the posterior probabilities $p(y|\mathbf{x})$

For two classes $y \in \{0, 1\}$ we get:

$$p(y=0|\mathbf{x}) = \frac{p(y=0)p(\mathbf{x}|y=0)}{p(\mathbf{x})} \text{ we can marginalize over } y$$

$$p(y=0|\mathbf{x}) = \frac{p(y=0)p(\mathbf{x}|y=0)}{p(y=0)p(\mathbf{x}|y=0) + p(y=1)p(\mathbf{x}|y=1)}$$

$$p(y=0|\mathbf{x}) = \frac{1}{1 + \frac{p(y=1)p(\mathbf{x}|y=1)}{p(y=0)p(\mathbf{x}|y=0)}}$$

$$p(y=0|\mathbf{x}) = \frac{1}{1 + e^{\ln\left(\frac{p(y=1)p(\mathbf{x}|y=1)}{p(y=0)p(\mathbf{x}|y=0)}\right)}}$$

$$p(y=0|\mathbf{x}) = \frac{1}{1 + e^{-\ln\left(\frac{p(y=0)p(\mathbf{x}|y=0)}{p(y=1)p(\mathbf{x}|y=1)}\right)}} \rightarrow \text{Decision boundary}$$

$$p(y=0|\mathbf{x}) = \frac{1}{1+e^{-\ln\frac{p(y=0)}{p(y=1)}-\ln\frac{p(\mathbf{x}|y=0)}{p(\mathbf{x}|y=1)}}}$$

$\therefore$ The posterior can be written in terms of a logistic function:

$$p(y=0|\mathbf{x}) = \frac{1}{1+e^{-F(\mathbf{x})}}$$

And for the other posterior we have:

$$p(y=1|\mathbf{x}) = 1 - p(y=0|\mathbf{x}) \text{ two-class problem}$$

$$= \frac{e^{-F(\mathbf{x})}}{1+e^{-F(\mathbf{x})}}$$

$$= \frac{1}{1+e^{+F(\mathbf{x})}}$$

$\Rightarrow$ just by changing a sign we can find the posterior for $y=0$ or $y=1$.

Definition

The logistic function (also called sigmoid function) is defined by:

$$g(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}} \text{ , where } \mathbf{x} \in \mathbb{R}$$

The derivative of this function is very useful, so let's compute it:

$$g'(\mathbf{x}) = \frac{e^{-\mathbf{x}}}{(1+e^{-\mathbf{x}})^2}$$

$$g'(\mathbf{x}) = \frac{1}{(1+e^{-\mathbf{x}})}\frac{e^{-\mathbf{x}}}{(1+e^{-\mathbf{x}})}$$

$$g'(\mathbf{x}) = \frac{1}{(1+e^{-\mathbf{x}})}\frac{1}{(1+e^{\mathbf{x}})}$$

$$g'(\mathbf{x}) = g(\mathbf{x})g(-\mathbf{x})$$

$$\boxed{g'(\mathbf{x}) = g(\mathbf{x})(1-g(\mathbf{x}))}$$

How can we find the decision boundary $\delta(\mathbf{x}) = 0$?
It is the function that separates the two classes... so it is exactly formed with those points where no decision can be made.
$\Rightarrow$ points $\mathbf{x}$ on the decision boundary satisfy:

$$p(y = 0|\mathbf{x}) = p(y = 1|\mathbf{x})$$

Thus, the ratio of the posteriors is like this on the decision boundary:

$$\ln \frac{p(y = 0|\mathbf{x})}{p(y = 1|\mathbf{x})} = \ln 1 = 0$$

Lemma: the decision boundary is given by $F(\mathbf{x}) = 0$, and it is a sigmoid/logistic function.

Proof:

$$\ln \frac{p(y=0|\mathbf{x})}{p(y=1|\mathbf{x})} = F(\mathbf{x}) = 0$$

$$\frac{p(y=0|\mathbf{x})}{p(y=1|\mathbf{x})} = e^{F(\mathbf{x})}$$

$$p(y=0|\mathbf{x}) = e^{F(\mathbf{x})} p(y=1|\mathbf{x}) \text{ or } p(y=0|\mathbf{x}) = e^{F(\mathbf{x})}(1 - p(y=0|\mathbf{x}))$$

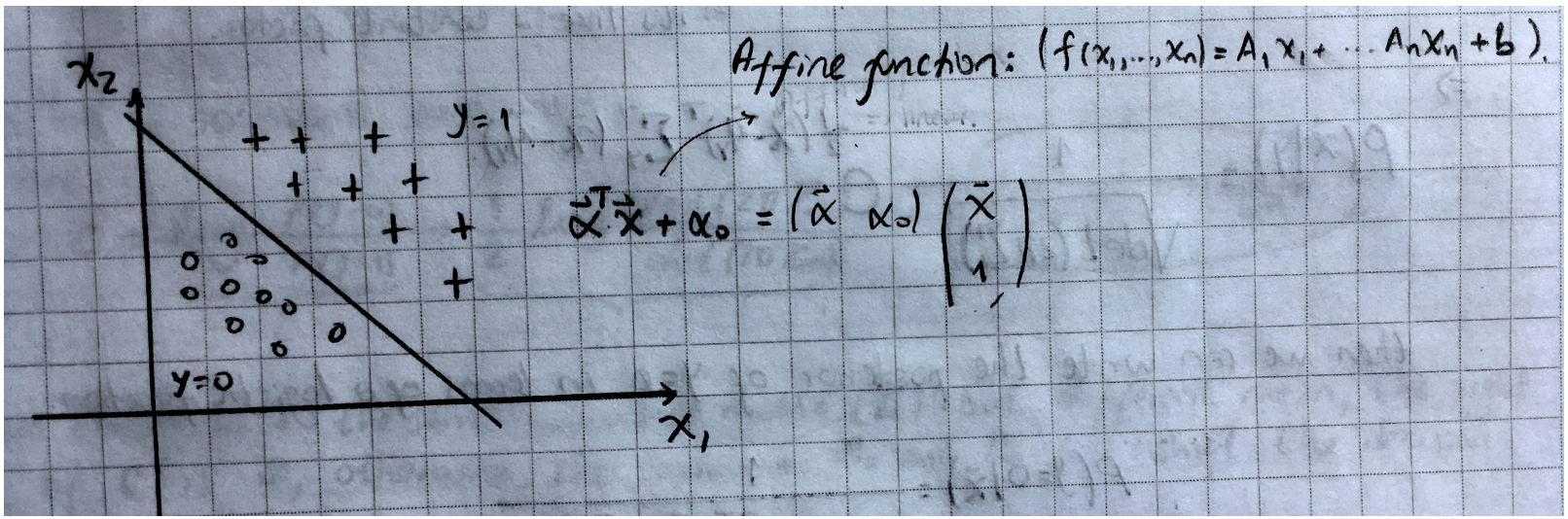
$$\therefore \; p(y=0|\mathbf{x}) = e^{F(\mathbf{x})} - e^{F(\mathbf{x})} p(y=0|\mathbf{x})$$

$$p(y=0|\mathbf{x})(1 + e^{F(\mathbf{x})}) = e^{F(\mathbf{x})}$$

$$\therefore \; \boxed{p(y=0|\mathbf{x}) = \frac{e^{F(\mathbf{x})}}{1 + e^{F(\mathbf{x})}} = \frac{1}{1 + e^{-F(\mathbf{x})}}}$$

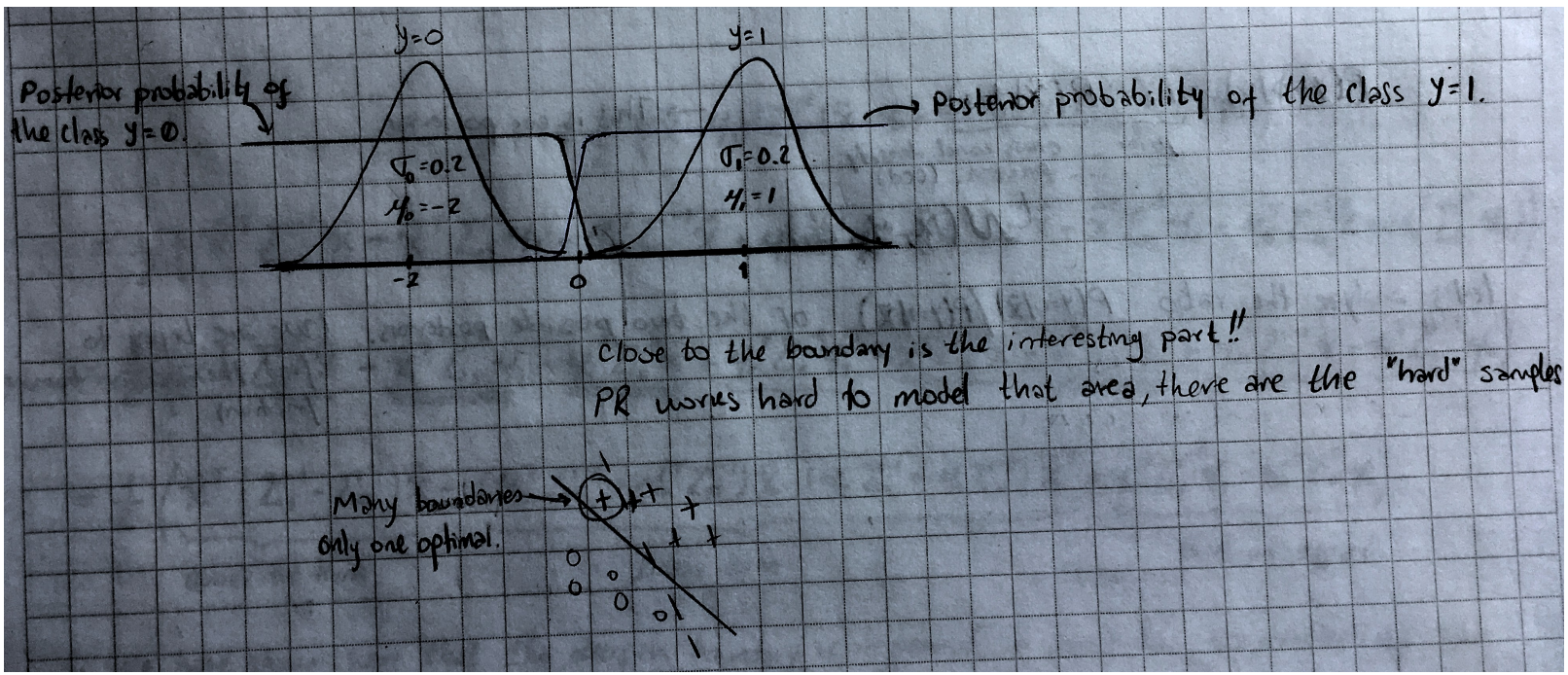
How the feature space looks like when the decision boundary is an *Affine function*?

$$p(y=0|\mathbf{x}) = \frac{1}{1+e^{-F(\mathbf{x})}} = \frac{1}{1+e^{-(\alpha^T\mathbf{x}+\alpha_0)}}$$



The class conditionals (CCD) $p(\mathbf{x}|y)$ can be for instance Gaussians where the two classes share the same covariance matrix $\rightarrow$ linear decision boundary!

$\Rightarrow$ this shows us the relation between Bayesian Classifiers and Linear Classifiers.



Example: Gaussians are usually a good/right choice!

$$p(\mathbf{x}|y) = \frac{1}{\sqrt{(2\pi)^d \det(\mathbf{\Sigma})}} e^{-\frac{1}{2}(\mathbf{x}-\mu_y)^T \mathbf{\Sigma}_y^{-1}(\mathbf{x}-\mu_y)}$$

Where $\mathbf{x}$ is a $d-$component feature vector; μ is the $d-$component mean vector; $\mathbf{\Sigma}$ is a $d \times d$ covariance matrix; and $\det(\mathbf{\Sigma})$ is the determinant of $\mathbf{\Sigma}$.

If we include 2π into the $\det(\cdot)$ operator, we can get-rid-off the $d \rightarrow$ it is like a constant.

$$p(\mathbf{x}|y) = \frac{1}{\sqrt{\det(2\pi\mathbf{\Sigma})}} e^{-\frac{1}{2}(\mathbf{x}-\mu_y)^T \mathbf{\Sigma}_y^{-1}(\mathbf{x}-\mu_y)}$$

If we assume that the decision boundary of a two-class problem with Gaussian CCD is a parabola, the posterior of $y = 0$ can be written in terms of a logistic function as follows:

$$p(y = 0|\mathbf{x}) = \frac{1}{1 + e^{\mathbf{x}^T \mathbf{A}\mathbf{x} + \alpha^T \mathbf{x} + \alpha_0}}$$

where $\mathbf{x}^T \mathbf{A}\mathbf{x}$ is the quadratic component of $\mathbf{x}$.
$\alpha^T \mathbf{x}$ is the linear component of $\mathbf{x}$.
α_0 is a bias (intercept).

$\therefore$ the decision boundary looks like a parabola.

Let's show that this is actually true.

$$p(y=0|\mathbf{x}) = \underbrace{p(y)}_{\text{prior}} \quad \underbrace{p(\mathbf{x}|y=0)}_{\text{CCD: } \mathcal{N}(\mathbf{x},\mu_y,\mathbf{\Sigma}_y)} \quad \text{this is our posterior}$$

Now let's analyze the ratio of the two possible posteriors:

$$\frac{p(y=0|\mathbf{x})}{p(y=1|\mathbf{x})}$$

We want to find the decision boundary function to show that it is a parabola.

$$\frac{p(y=0|\mathbf{x})}{p(y=1|\mathbf{x})} = \frac{p(y=0)p(\mathbf{x}|y=0)}{p(y=1)p(\mathbf{x}|y=1)}$$

Note that the marginalization over y to find $p(\mathbf{x})$ disappears.

$\Rightarrow$

$$\ln \frac{p(y=0|\mathbf{x})}{p(y=1|\mathbf{x})} = \ln \frac{p(y=0)}{p(y=1)} + \ln \frac{\frac{1}{\sqrt{\det(2\pi\mathbf{\Sigma}_0)}} e^{-\frac{1}{2}(\mathbf{x}-\mu_0)^T \mathbf{\Sigma}_0^{-1}(\mathbf{x}-\mu_0)}}{\frac{1}{\sqrt{\det(2\pi\mathbf{\Sigma}_1)}} e^{-\frac{1}{2}(\mathbf{x}-\mu_1)^T \mathbf{\Sigma}_1^{-1}(\mathbf{x}-\mu_1)}}$$

If the priors $p(y=0)$ and $p(y=1)$ are NOT equal, they just shift the decision boundary (they are like an offset).

This function has a constant component:

$$c = \ln \frac{p(y=0)}{p(y=1)} + \frac{1}{2} \ln \frac{\det(2\pi\boldsymbol{\Sigma}_1)}{\det(2\pi\boldsymbol{\Sigma}_0)}$$

If the two classes have the same covariance matrices $\boldsymbol{\Sigma}$ but different means, the right hand side of c is 0, otherwise $\boldsymbol{\Sigma}_0$ and $\boldsymbol{\Sigma}_1$ just shift the decision boundary, as the priors.

$\Rightarrow$ what we can do if don't have prior information? we can find the decision boundary and shifts it to gain some prior information.

Now let's continue with the rest of the equation:

$$\ln \frac{e^{-\frac{1}{2}(\mathbf{x}-\mu_0)^T\boldsymbol{\Sigma}_0^{-1}(\mathbf{x}-\mu_0)}}{e^{-\frac{1}{2}(\mathbf{x}-\mu_1)^T\boldsymbol{\Sigma}_1^{-1}(\mathbf{x}-\mu_1)}} = \frac{1}{2}\left((\mathbf{x}-\mu_1)^T\Sigma_1^{-1}(\mathbf{x}-\mu_1) - (\mathbf{x}-\mu_0)^T\Sigma_0^{-1}(\mathbf{x}-\mu_0)\right)$$

$$\frac{1}{2}\left(\mathbf{x}^T\Sigma_1^{-1}\mathbf{x} - \mathbf{x}^T\Sigma_1^{-1}\mu_1 - \mu_1^T\Sigma_1^{-1}\mathbf{x} + \mu_1^T\Sigma_1^{-1}\mu_1 - \left(\mathbf{x}^T\Sigma_0^{-1}\mathbf{x} - \mathbf{x}^T\Sigma_0^{-1}\mu_0 - \mu_0^T\Sigma_0^{-1}\mathbf{x} + \mu_0^T\Sigma_0^{-1}\mu_0\right)\right)$$

$$\frac{1}{2}\left(\mathbf{x}^T\Sigma_1^{-1}\mathbf{x} - \mathbf{x}^T\Sigma_1^{-1}\mu_1 - \mu_1^T\Sigma_1^{-1}\mathbf{x} + \mu_1^T\Sigma_1^{-1}\mu_1 - \mathbf{x}^T\Sigma_0^{-1}\mathbf{x} + \mathbf{x}^T\Sigma_0^{-1}\mu_0 + \mu_0^T\Sigma_0^{-1}\mathbf{x} - \mu_0^T\Sigma_0^{-1}\mu_0\right)$$

$$\frac{1}{2}\left(\underbrace{\mathbf{x}^T\left(\Sigma_1^{-1} - \Sigma_0^{-1}\right)\mathbf{x}}_{\text{Quadratic term}} - \underbrace{\left(\mu_1^T\Sigma_1^{-1} + \mu_0^T\Sigma_0^{-1}\right)\mathbf{x} - \left(\mu_1\Sigma_1^{-1} + \mu_0\Sigma_0^{-1}\right)\mathbf{x}^T}_{\text{Linear term}} + \underbrace{\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0^T\Sigma_0^{-1}\mu_0}_{\text{Bias or offset}}\right)$$

$\therefore$ We have shown that the decision boundary is, in general, a parabola. No matter of the features, labels, classes, etc.

Additionally, if the two classes share the same covariance matrix, i.e., $\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1$, then the decision boundary is **linear** because $\boldsymbol{\Sigma}_0 - \boldsymbol{\Sigma}_1 = 0$ and the quadratic term disappears.

$\rightarrow$ We found the exponent of a sigmoid/logistic function for a quadratic decision boundary as: $\mathbf{x}^T\mathbf{A}\mathbf{x} + \alpha^T\mathbf{x} + \alpha_0$

Note 1: We could have created a decision boundary right away by assuming Gaussianity in the data or we could have computed the decision boundary by finding $\mathbf{A}$, α, and α_0.

- In the second approach we do not assume Gaussianity. We adjust the parameters of the sigmoid function properly to find the decision boundary.
- In the first approach, where Gaussianity is assumed, the results are very likely to be different because only a certain subset of straight lines satisfy the Gaussianity assumption.

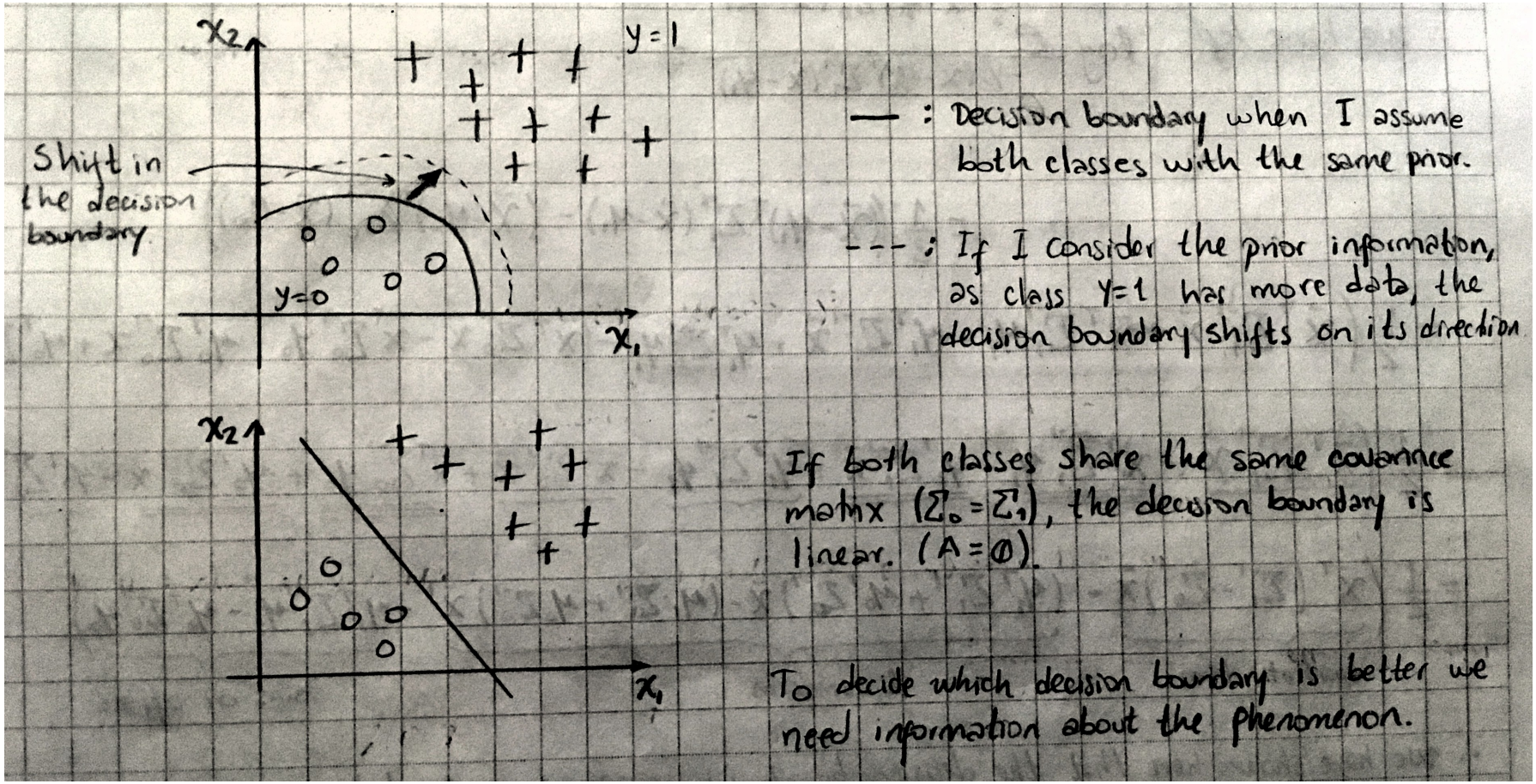
Note 2: To find the better classifier it is necessary to train it and test it... there is no cooking-book.

Then, we have:

$$\mathbf{A} = \frac{1}{2}\left(\mathbf{\Sigma}_1^{-1} - \mathbf{\Sigma}_0^{-1}\right)$$

$$\alpha^T = \mu^T \mathbf{\Sigma}_1^{-1} + \mu_0^T \mathbf{\Sigma}_0^{-1}$$

$$\alpha_0 = \ln \frac{p(y=0)}{p(y=1)} + \frac{1}{2}\left(\ln \frac{\det(2\pi\mathbf{\Sigma}_1)}{\det(2\pi\mathbf{\Sigma}_0)} + \mu_1^T \mathbf{\Sigma}_1^{-1}\mu_1 - \mu_0^T \mathbf{\Sigma}_0^{-1}\mu_0\right)$$


x_2
$y=1$
Shift in the decision boundary
$y=0$
x_1
x_2
x_1
— : Decision boundary when I assume both classes with the same prior.
--- : If I consider the prior information, as class Y=1 has more data, the decision boundary shifts on its direction
If both classes share the same covariance matrix ($\Sigma_0 = \Sigma_1$), the decision boundary is linear. ($A = \emptyset$).
To decide which decision boundary is better we need information about the phenomenon.

Note 1: if the CCD functions are Gaussians and share the same covariance matrix, the argument of the exponential function in the sigmoid/logistic function is affine (linear) in $\mathbf{x}$.

Note 2: the result above is even true for a more general family of PDFs and it is not limited to Gaussians.

Definition: the exponential family is a class of PDFs that can be written in the following canonical form:

$$p(\mathbf{x}, \theta, \phi) = e^{\frac{\theta^T \mathbf{x} - b(\theta)}{a(\phi)} + c(\mathbf{x}, \phi)}$$

where $\theta \in \mathbb{R}^d$ is the location parameter vector and ϕ is the dispersion parameter.

It is possible to show that for all family of exponential functions like this one above, if the dispersion parameter is constant, they lead to a linear decision boundary.

The proof would be like we did with the Gaussians distributions. Take the CCDs ratio and end up with a linear function.

Lemma: If all CCD are members of the same exponential family distribution with equal dispersion ϕ, the decision boundary $F(\mathbf{x}) = 0$ is linear in the components of $\mathbf{x}$.

$\therefore$ Linear decision boundaries cover much more than Gaussian distributions with the same covariance.

Let's assume we have a logistic regression function as our posterior:

$$\frac{1}{1 + e^{F_\theta}} = p(y|\mathbf{x})$$

The question is how to estimate the parameters of the decision boundary given the observations.
$\Rightarrow$ we apply the Maximum Likelihood Estimation (MLE).

MLE assumes that the training samples are mutually independent.
$\Rightarrow$ Each training sample (e.g., each circle in the previous figure) is independent.

Then we try to maximize the PDF to observe the set of training samples:
$p(\mathbf{x}_1, y_1), p(\mathbf{x}_2, y_2), \ldots, p(\mathbf{x}_m, y_m)$
$\Rightarrow$ Try to find the parameters that maximize this product!

If the decision boundary is linear: $\alpha \mathbf{x}^T + b$, then we need to find α and b such that maximize the PDF above.

We can try with the Log-Likelihood Function... let's see that!

Log-Likelihood Function

Let's assume the posteriors are given by:

$$p(y=0|\mathbf{x}) = 1 - g(\theta^T\mathbf{x}) \text{ and } p(y=1|\mathbf{x}) = g(\theta^T\mathbf{x})$$

where $g(\theta^T\mathbf{x})$ is the sigmoid function $\frac{1}{1+e^{F_\theta(\mathbf{x})}}$, parameterized in θ.

The parameter vector θ has to be estimated from a set $\mathbb{S}$ of m training samples:

$$\mathbb{S} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_m, y_m)\}$$

Method of choice to find $\theta \Rightarrow$ MLE.

Before to start, we can re-write the posteriors using the Bernoulli probability distribution:

$$p(y|\mathbf{x}) = g(\theta^T\mathbf{x})^y \left(1 - g(\theta^T\mathbf{x})\right)^{1-y}$$

Note what happens when $y = 0$ or $y = 1$ (each class).

We can compute the log-likelihood function (assuming mutual independence among all training samples):

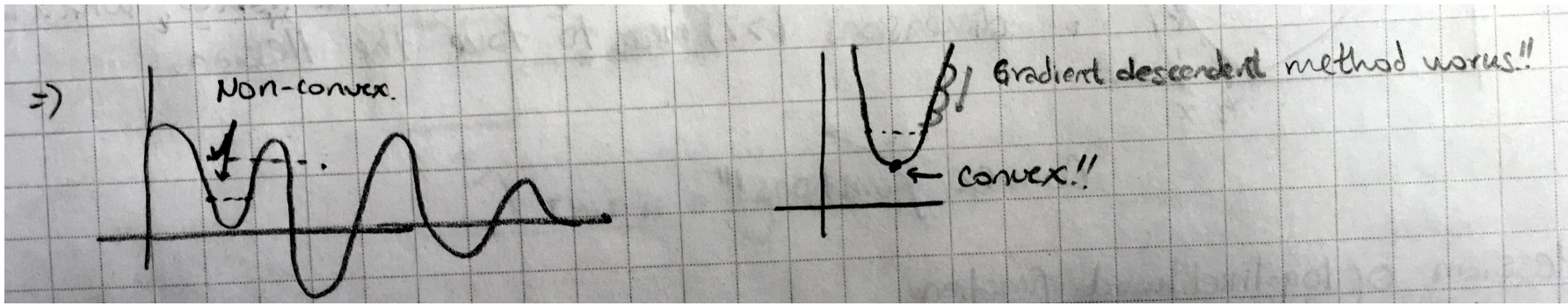
$$l(\theta) = \log \prod_{i=1}^{m} p(y_i|\mathbf{x}_i)$$

$$= \sum_{i=1}^{m} \log \left[g(\theta^T\mathbf{x}_i)^{y_i} \left(1 - g(\theta^T\mathbf{x}_i)\right)^{1-y_i} \right]$$

$$= \sum_{i=1}^{m} \left[y_i \log g(\theta^T\mathbf{x}_i) + (1 - y_i) \log \left(1 - g(\theta^T\mathbf{x}_i)\right) \right]$$

Note 1 (for experts): The negative of the log-likelihood function is the cross-entropy of y and $g(\theta^T \mathbf{x})$.

Note 2: The negative of the log-likelihood function is a convex function.



$g(\theta^T \mathbf{x})$ is a non-linear function $\Rightarrow$ we will need to compute the gradient for a non-linear function regarding θ.

Gradient of the log-likelihood function

For the $i-$th feature vector (training sample) on its $j-$th component, the gradient is as follows:

$$\frac{\partial}{\partial \theta_j} l(\theta) = \frac{\partial}{\partial \theta_j} \left[\sum_{i=1}^{m} \left(y_i \log g(\theta^T \mathbf{x}_i) + (1 - y_i) \log \left(1 - g(\theta^T \mathbf{x}_i) \right) \right) \right]$$

$$= \sum_{i=1}^{m} \left(\frac{y_i}{g(\theta^T \mathbf{x}_i)} - \frac{1 - y_i}{1 - g(\theta^T \mathbf{x}_i)} \right) \underbrace{\frac{\partial}{\partial \theta_j} g(\theta^T \mathbf{x}_i)}$$

Additionally, $g(\theta^T \mathbf{x})$ is a Sigmoid function, then the property $g'(x) = g(x)(1 - g(x))$ holds.

$\therefore$

$$\frac{\partial}{\partial \theta_j} l(\theta) = \sum_{i=1}^{m} \left(\frac{y_i}{g(\theta^T \mathbf{x}_i)} - \frac{1 - y_i}{1 - g(\theta^T \mathbf{x}_i)} \right) g(\theta^T \mathbf{x}_i)(1 - g(\theta^T \mathbf{x}_i)) x_{ij}$$

x_{ij} is the $j-$th component of the $i-$th feature vector (a training sample).

$$\frac{\partial}{\partial \theta_j} l(\theta) = \sum_{i=1}^{m} \left(y_i (1 - g(\theta^T \mathbf{x}_i)) - (1 - y_i) g(\theta^T \mathbf{x}_i) \right) x_{ij}$$

$\therefore$

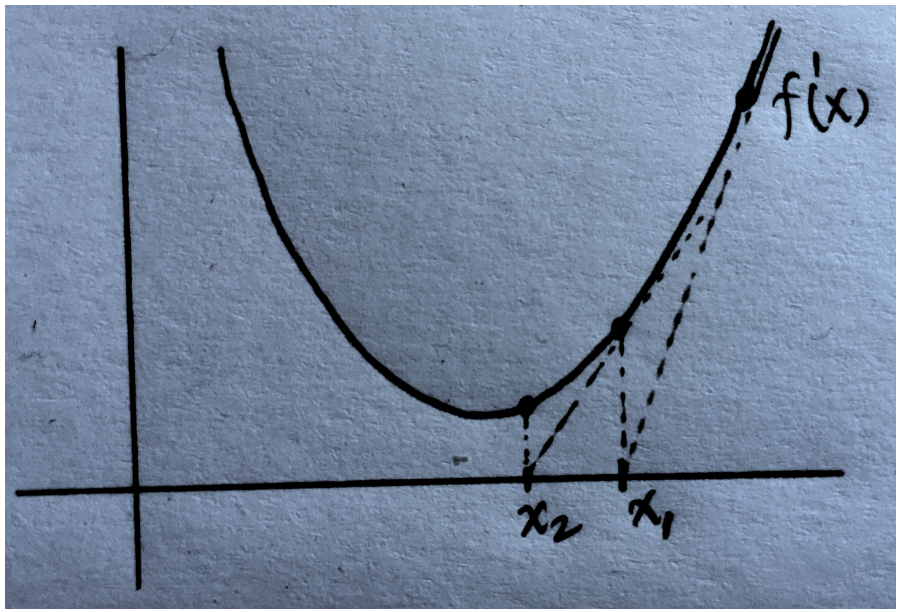
$$\frac{\partial}{\partial \theta_j} l(\theta) = \sum_{i=1}^{m} \left(y_i - g(\theta^T \mathbf{x}_i) \right) x_{ij}$$

or in vector notation:

$$\frac{\partial}{\partial \theta} l(\theta) = \sum_{i=1}^{m} \left(y_i - g(\theta^T \mathbf{x}_i) \right) \mathbf{x}_i$$

The class number y_i is approximated by $g(\theta^T \mathbf{x}_i)$, and their difference weights the x_{ij} feature component.

The iterative computation of the gradient can be done by the Newton-Raphson iteration scheme.



This requires to take second-order derivatives, which in multiple dimensions (more than 2) means to take the Hessian.

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

In our particular case:

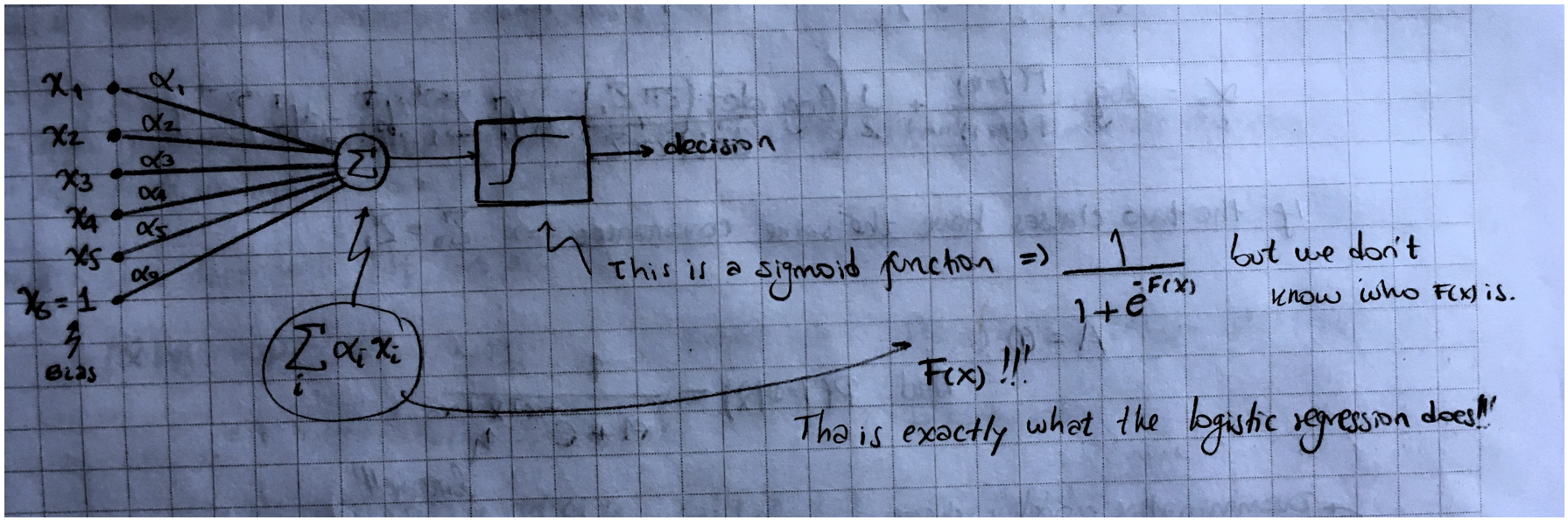
$$\frac{\partial^2}{\partial\theta\partial\theta^T}l(\theta) = \frac{\partial}{\partial\theta^T}\left(\frac{\partial}{\partial\theta}l(\theta)\right) = \frac{\partial}{\partial\theta^T}\sum_{i=1}^{m}\left(y_i - g(\theta^T\mathbf{x}_i)\right)\mathbf{x}_i$$

$\Rightarrow$

$$\frac{\partial^2}{\partial\theta\partial\theta^T}l(\theta) = \sum_{i=1}^{m} g(\theta^T\mathbf{x}_i)(1 - g(\theta^T\mathbf{x}_i))\mathbf{x}_i\mathbf{x}_i^T$$

The Newton-Raphson recursion supports this computation and can be used to find the gradient over convex/concave functions.

Relationship between the Perceptron and Logistic Regression



Lessons learned

- Posteriors can be written in terms of a logistics function.
- Given the decision boundary $F(\mathbf{x}) = 0$, we can write down the posterior $p(y|\mathbf{x})$ right away.
- The decision boundary for normally distributed feature vectors for each class is a quadratic function.
- If Gaussians share the same covariance matrices, the decision boundary is a linear function.
- In general terms Gaussians $\Rightarrow$ 2nd order decision boundary.