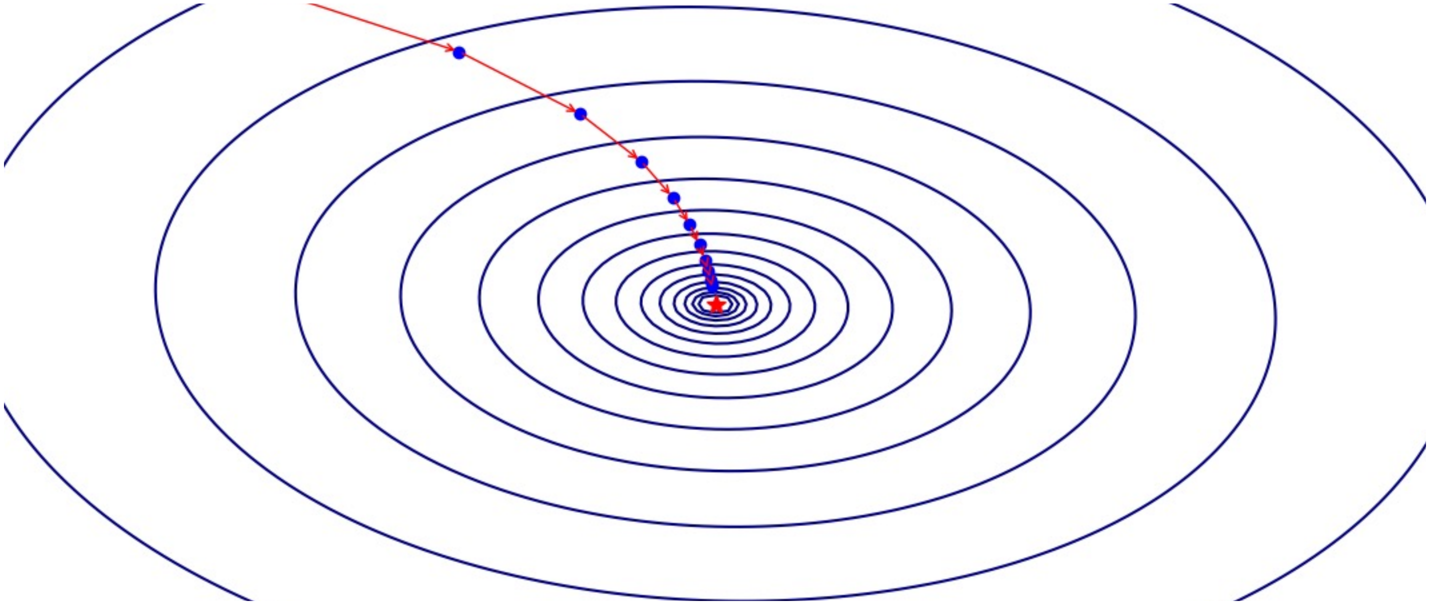


Machine Learning in Signal Processing

Winter Semester 2022/23

4. Linear Classification

07.11.2023

Prof. Dr. Vasileios Belagiannis

Chair of Multimedia Communications and Signal Processing

Friedrich-Alexander-Universität
Technische Fakultät

Course Topics

1. Introduction.
2. Basics and terminology.
3. Linear regression.
4. **Linear classification.**
5. Performance evaluation.
6. Neural networks.
7. Deep neural networks.
8. Decision trees.
9. Ensemble models.
10. Random forests.
11. Clustering / Unsupervised learning.
12. Dimensionality reduction.
13. Support vector machines.
14. Recap and Q&A.

- The exam will be written.
- We will have an exam preparation test before the end of the year.

Acknowledgements

Ideas and inspiration from:

- CSC311 Introduction to Machine Learning, University of Toronto.
- Introduction to Machine Learning: LMU Munich.
- Introduction to Machine Learning, CSAIL, MIT.
- CSE 574 Introduction to Machine Learning, University of Buffalo.
- Special thanks Arij Bouazizi, Julia Hornauer, Julian Wiederer, Adrian Holzbock and Youssef Dawoud for contributing to the lecture preparation.

Last Lecture Recap

- Linear regression.
  - Normal equation.
  - Gradient descent.
- Convergence.
- Polynomial regression.
- Regularisation.
- Ridge regression.
- Lasso regression.

Today’s Agenda and Objectives

- Linear classifiers.
- Grouping.
- Decision Regions and Boundaries.
- Linear Classifiers.
  - Perceptron Learning Rule.
  - Logistic Regression.
  - Naive Bayes Classifier.
  - Linear Discriminant Analysis.

Classification Task

- The goal of classification is to predict a class label y from a finite set of categories $y \in \mathcal{Y} = \{1, \dots, n\}$ for a given input sample. n is the number of classes.
- For example, classifying images with blooms:

Bloom (class 1)

No bloom (class 0)

- We consider the following cases:
  - Binary classification, where $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, +1\}$ and $n = 2$.
  - Multiclass classification, where $\mathcal{Y} = \{1, \dots, n\}$ and $n \geq 3$.

Classification Task (Cont.)

- Binary classification, where $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, +1\}$ and $n = 2$.
  - Medical diagnosis: classification whether a patient has a specific disease based on medical scans (image/volume input).
  - Classification of e-mails into spam and non-spam (text input).
  - Anomalous sound detection (audio input).
- Multiclass classification, where $\mathcal{Y} = \{1, \dots, n\}$ and $n \geq 3$.
  - Handwritten digit recognition (image input)
  - Classifying e-mails into multiple categories spam, work, family, insurance and banking (text input).
  - Voice recognition (audio input).
- We refer to the problem also as statistical classification.

Grouping Classification Algorithms

- A classification algorithm can be defined as a mapping function $f: \mathcal{X} \rightarrow \mathbb{R}^n$ that assign continuous outputs to given data points $x \in \mathcal{X}$.
- *Why do we define the classifier with continuous output?*
- Continuous vs discrete output:
  - Continuous-valued cost functions are easier to optimize, e.g. gradient based optimization.
  - In general, the continuous output is more informative.
  - The continuous output can be transformed to discrete (class) values. The opposite does not work.

Grouping Classification Algorithms (Cont.)

- We differentiate between to types of classifier depending if the outputs are scores or probabilities.
  - Scoring classifiers.
  - Probabilistic classifiers.
- In addition, for every classifier we differentiation between two classification approaches:
  - Generative classifiers.
  - Discriminative classifiers.
  - The classification approach depends on whether the classifier learn the model that generated the data or only separates the specified classes given the input features.

Scoring Classifiers

- For binary classification ($n = 2$), we have only one mapping function $f(x) = f_{+1}(x) - f_{-1}(x)$ as discriminant / scoring function.
  - The class labels are $\mathcal{Y} = \{-1, +1\}$.
  - The discrete class value is chosen by $g(x) = \mathrm{sgn}\big(f(x)\big)$ or by thresholding $g(x) \coloneqq [f(x) \geq \delta]$ with $\delta = 0$.
  - $|f(x)|$ is referred to as confidence.
- *How many scoring functions do we have for a multi-class classification problem with 5 class categories?*
- For n-class classification ($n \geq 3$), we have n discriminant / scoring functions $f_1, \ldots, f_n \colon \mathcal{X} \to \mathbb{R}$.
  - The class labels are $\mathcal{Y} = \{1, \ldots, n\}$.
  - The scores $f_1(x), \ldots, f_n(x)$ are transformed into discrete class values by choosing the class with the maximum score: $g(x) = \arg \max_{i \in \{1,\ldots,n\}} f_i(x)$.

Probabilistic Classifiers

- Probabilistic classifiers are a special case of scoring classifiers. The mapping function can be denoted as $\mathrm{f}(\mathrm{x})\colon\ \mathcal{X} \rightarrow [0, 1]^{\mathrm{n}}$.
- For binary classification ($\mathrm{n} = 2$), a single function $\mathrm{f}(\mathrm{x})$ is sufficient. The probability output is transformed to a discrete class value with $\mathrm{g}(\mathrm{x}) \coloneqq [\mathrm{f}(\mathrm{x}) \geq \delta]$ with $\delta = 0.5$.
- For n-class classification ($\mathrm{n} \geq 3$), we have n probability functions
  $\mathrm{f}_1, \ldots, \mathrm{f}_\mathrm{n}\colon \mathcal{X} \rightarrow [0, 1]$ with $\sum_\mathrm{i} \mathrm{f}_\mathrm{i} = 1$.
- The probabilities $\mathrm{f}_1(\mathbf{x}), \ldots, \mathrm{f}_\mathrm{n}(\mathbf{x})$ are transformed into discrete class values by choosing the class with the maximum probability: $\mathrm{g}(\mathbf{x}) = \arg\max_{\mathrm{i} \in \{1,\ldots,\mathrm{n}\}} \mathrm{f}_\mathrm{i}(\mathbf{x})$.

Generative Classifiers

- Generative models aim to model both hidden and observed variables, e.g. input / output variables.
- Generative models assume that the distribution underlying the data follows a specific parametric form → parametric <u>probability density estimation</u>.
  - Our continuous (random) variables have a probability distribution. The probability density function (PDF) models the relationship between the input / output variables.
  - The shape of PDF defines the probability distribution function, e.g. Gaussian distribution.
  - Probability density estimation: we rely on the values of the observed variables to estimate the density of the probabilities.
- *Name an example of a generative model relationship.*

Reference to Density Estimation: https://machinelearningmastery.com/probability-density-estimation/

Generative Classifiers (Cont.)

- Generative classifiers model the conditional distribution $p(\mathrm{x}|y = i)$. Therefore, generative classifiers apply the Bayes theorem:
  - $\mathrm{f_i(x)} = \mathbb{P}(\mathrm{y = i|x}) = \frac{\mathbb{P}(\mathrm{x|y{=}i})\mathbb{P}(\mathrm{y{=}i})}{\mathbb{P}(\mathrm{x})} = \frac{\mathrm{p(x|y = i)p(y{=}i)}}{\sum_{\mathrm{j=1}}^{\mathrm{n}} \mathrm{p(x|y = j)p(y{=}j)}}$
- f_i are the class probabilities estimated from the training data.
- <u>Posterior probability</u> $\mathbb{P}(\mathrm{y = i|x})$: the conditional probability of $\mathrm{y = i}$ given that x occurs.
- <u>Likelihood</u> $\mathbb{P}(\mathrm{x|y = i})$: the conditional probability of x given that $\mathrm{y = i}$ occurs.
- <u>Prior</u> $\mathbb{P}(\mathrm{y = i})$: the prior probability of $\mathrm{y = i}$.
- <u>Evidence</u> $\mathbb{P}(\mathrm{x})$: the probability of x occurring.

Discriminative Classifiers

- There is not any assumption on the underlying data distribution → distribution free.
- They focus on a direct mapping between input/output variables, while it also learns the conditional probability (the posterior, as we just saw) $\mathbb{P}(y = i|x)$.
- Discriminative classifiers do not model the underlying probability distributions.
- Discriminative classifiers directly learn the decision boundaries between classes using <u>empirical risk minimization</u>.
  - They rely on empirical risk minimization with a defined loss function to directly learn the quantity of interest.

Empirical Risk Minimization

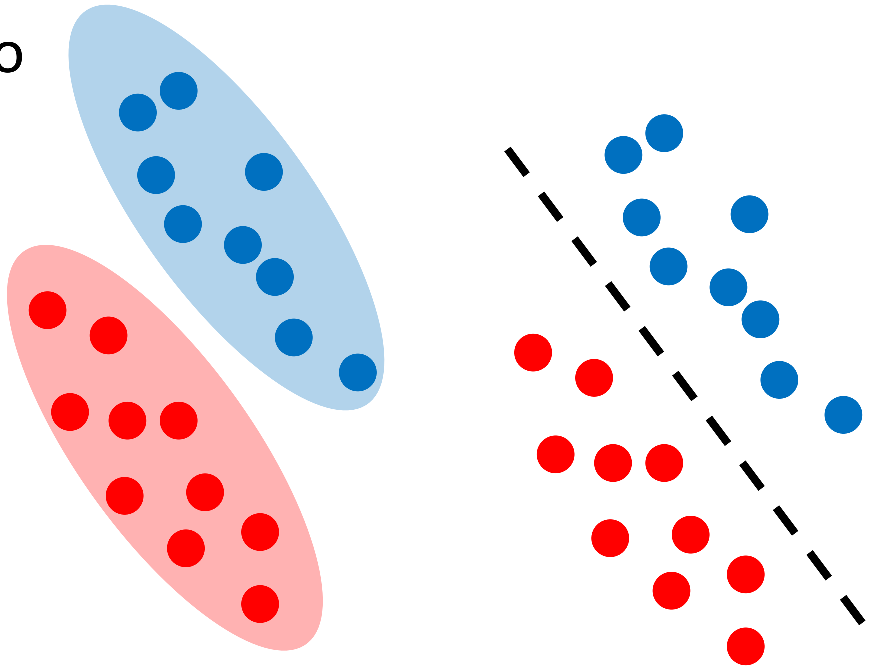
- In supervised learning, the cost function is normally defined as follows:
  - $\mathcal{J}(\mathbf{w}) = \mathbb{E}_{(\mathbf{x},y)\sim\hat{p}_{data}}\mathcal{L}(f(\mathbf{x};\mathbf{w}),y)$, where $\mathcal{L}$ is the loss function,
  - $f(\mathbf{x};\mathbf{w})$ the prediction function, $\mathbf{x}$ the input and y the label.
- $\hat{p}_{data}$ is the <u>empirical data distribution</u>, which corresponds to the training data.
- Assuming access to the data generating distribution p_{data}, the cost function becomes:
  - $\mathcal{J}^*(\mathbf{w}) = \mathbb{E}_{(\mathbf{x},y)\sim p_{data}}\mathcal{L}(f(\mathbf{x};\mathbf{w}),y)$.
- This is an ideal cost function that is not realistic due to the lack of access to p_{data}. It measures <u>the expected generalization error that is usually called risk</u>.

Empirical Risk Minimization (Cont.)

- Our goal is to minimise the risk $\mathcal{J}^*(\mathbf{w})$ but this is not possible.
- Instead we minimize the empirical risk as given by the cost function $\mathcal{J}(\mathbf{w})$. Given the training set $\mathcal{T} = \{\mathbf{x}_i, y_i\}_{i=1}^{m}$, we write the empirical risk as:
  - $\mathcal{J}(\mathbf{w}) = \mathbb{E}_{(\mathbf{x},y)\sim\hat{p}_{data}}\mathcal{L}(f(\mathbf{x};\mathbf{w}), y) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(f(\mathbf{x}_i;\mathbf{w}), y_i)$.
- Training the discriminative classifier or some other machine learning algorithm results in minimizing the average error in the training set.
- In total, we minimize the empirical risk and hope that the risk is minimized too.
- What is a major disadvantage of the empirical risk minimization (ERM)?
- A disadvantage for the ERM is that is can overfit to the training data.

Generative & Discriminative Classifiers

- The discussed ideas generalise to different machine learning algorithms.
- Example:
  - Our task is to recognise the English and German text.
  - The generative classifier will learn the English and German language first and then decide for the language.
  - The discriminative classifier will learn separate English from German without learning the language.



Generative Classifier, e.g., Naive Bayes Classifier.

Discriminative Classifier, e.g., Logistic Regression.

➡ It focuses on learning a decision boundary.

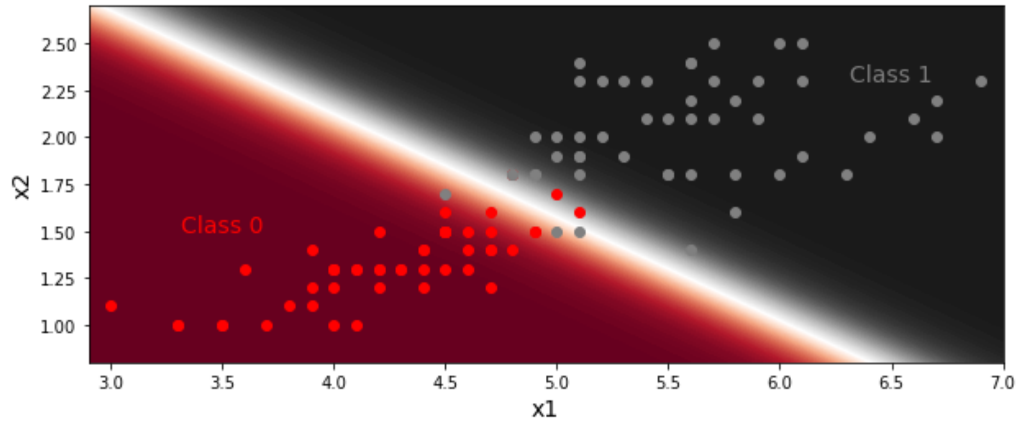
Comparison between the two models: https://www.baeldung.com/cs/ml-generative-vs-discriminative

Decision Regions and Boundaries

- Decision region: set of data points x to which the model assigns the class y: $\mathcal{X}_{\mathrm{y}} = \{\mathrm{x} \in \mathcal{X}: \mathrm{g}(\mathrm{x}) = \mathrm{y}\}$ where $\mathrm{g}(\cdot)$ the mapping function.
- The decision boundary separates the points of two or more classes.
- In the binary case, the decision boundary is given by: $\{\mathbf{x} \in \mathcal{X}: f(\mathbf{x}) = c\}$. It is a threshold.
- In the multiclass case, the decision boundary is defined as: $\{\mathbf{x} \in \mathcal{X}: \exists\, i \neq j \text{ s.t. } f_i(\mathbf{x}) = f_j(\mathbf{x}) \text{ and } f_i(\mathbf{x}), f_j(\mathbf{x}) \geq f_k(\mathbf{x})\ \forall k \neq i, j\}$.

Decision Regions and Boundaries (Cont.)

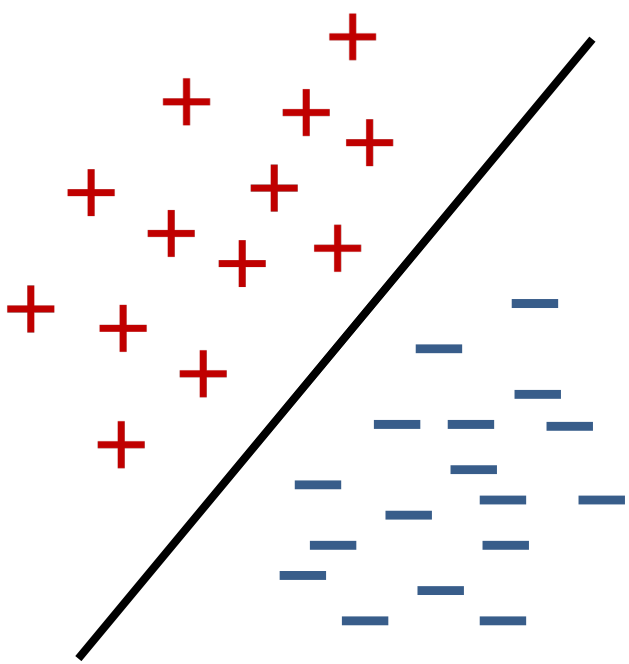
- The figure illustrates the hypersurfaces of two examples classes "0" and "1" with corresponding decision boundary as defined by a linear classifier.
- In the case of two classes, we a have linear binary classifier.



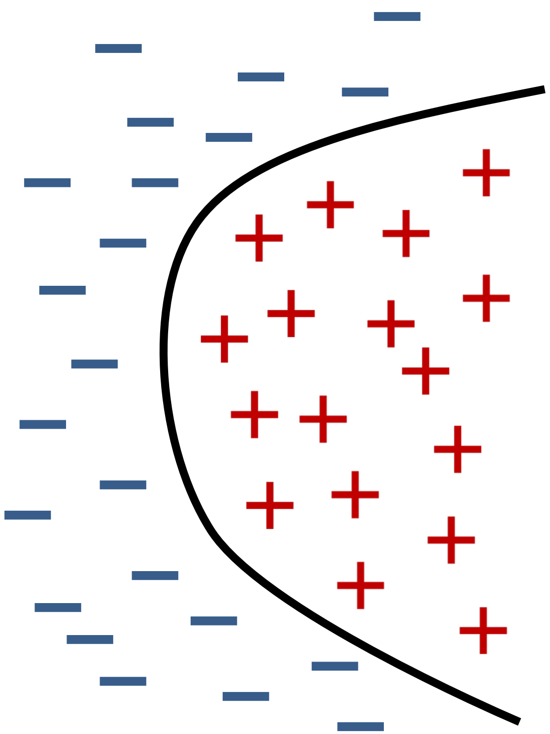
Linear and Non-Linear Decision Boundary

- Binary Classifier.

Linear Decision Boundary



Non-Linear Decision Boundary



Linear Classifiers

- A classifier is linear if the mapping function(s) $f_i(\mathbf{x})$ can be either defined directly as linear function(s) or trough a rank-preserving monotone transformation $g: \mathbb{R} \to \mathbb{R}$.
  - $f_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + \mathbf{b}_i$.
  - $g\big(f_i(\mathbf{x})\big) = \mathbf{w}_i^T \mathbf{x} + \mathbf{b}_i$.
- Rank-preserving implies that the decision region and the decision boundary do not change after applying the transformation g.
- For linear classifiers, the decision boundary between two classes i and j is a hyperplane separating the two classes:
  - $f_i(\mathbf{x}) = f_j(\mathbf{x})$
  - $g\big(f_i(\mathbf{x})\big) = g\left(f_j(\mathbf{x})\right)$
  - $\mathbf{w}_i^T \mathbf{x} + \mathbf{b}_i = \mathbf{w}_j^T \mathbf{x} + \mathbf{b}_j$
  - $\left(\mathbf{w}_i - \mathbf{w}_j\right)^T \mathbf{x} + \left(\mathbf{b}_i - \mathbf{b}_j\right) = 0$
- This hyperplane is $\left(\mathbf{w}_i - \mathbf{w}_j\right)^T \mathbf{x} + \left(\mathbf{b}_i - \mathbf{b}_j\right)$.

Linear Classifiers: Threshold Logic Unit

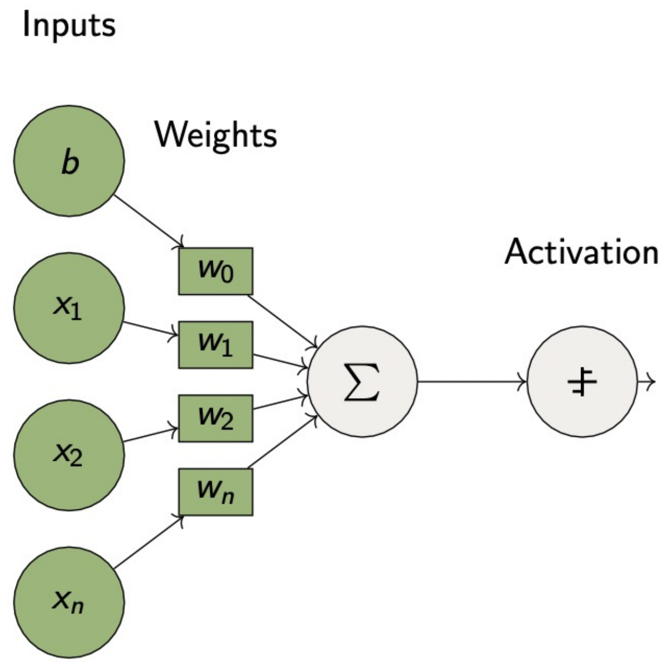
- Binary input $x_i \in \{0,1\}$ and binary output $y \in \{0,1\}$.
- Excitatory or inhibitory input, represented by the not trainable weights $w_i \in \{-1,1\}$.
- The mapping is represented by a threshold function $f: \mathbb{R}^d \rightarrow \mathbb{R}$, where:
  - $f(\mathbf{x}) = \begin{cases} 0, & if\ \mathbf{w} \cdot \mathbf{x} \leq T \\ 1, & otherwise. \end{cases}$
- This is a linear binary classifier that was used to model gate functions.

McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." *The bulletin of mathematical biophysics* 5.4 (1943): 115-133.

Linear Classifiers: Perceptron Learning Rule

- It is a linear classifier for binary problems.
- We have a real-valued input and train-able weights / parameters w_i.
- Training data is the ground-truth is required $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{m}$.
- It is represented by a threshold function $f: \mathbb{R}^d \to \mathbb{R}$, where:

$$f(\mathbf{x}) = \begin{cases} 1, if\ \mathbf{w} \cdot \mathbf{x} + b \geq 0 \\ 0, \qquad otherwise. \end{cases}$$



Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65.6 (1958): 386.

Linear Classifiers: Perceptron Learning Rule (Cont.)

- Learning rule: the parameter w update is proportional to the input; and the difference between prediction $f(\mathbf{x})$and ground-truth y_i.
- Algorithm for single-layer Perceptron:
  1. Initialize the parameters w, set learning rate η and threshold.
  2. Update the weights: $w_j(t+1) = w_j(t) + \eta\big(y - f(\mathbf{x})\big)\mathbf{x}$
  3. Iterate the update until convergence.

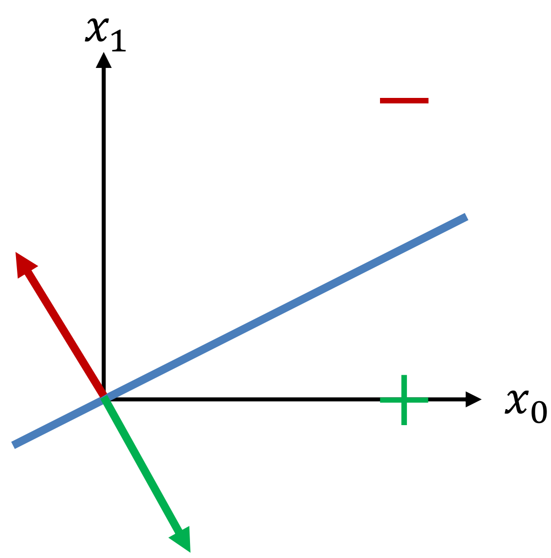
Online example: https://vitalflux.com/perceptron-explained-using-python-example/

NOT Gate

- For the NOT gate only a single input x_1 is required.
- The weights / parameters can be represented by *half-spaces*: $H_+ = \{\mathbf{x}: \mathbf{w}^T\mathbf{x} \geq 0\}$ and $H_- = \{\mathbf{x}: \mathbf{w}^T\mathbf{x} < 0\}$.
  - Half-space: the set of data points from the one side of the hyper-plane.
- The boundary $\mathbf{x}: \mathbf{w}^T\mathbf{x} = 0$ is the decision boundary.
- The training data is linearly separable, since we can find a decision boundary that correctly classifies all data points.
- An auxiliary input x_0 is added for the geometric visualization.

x_1	y
0	1
1	0

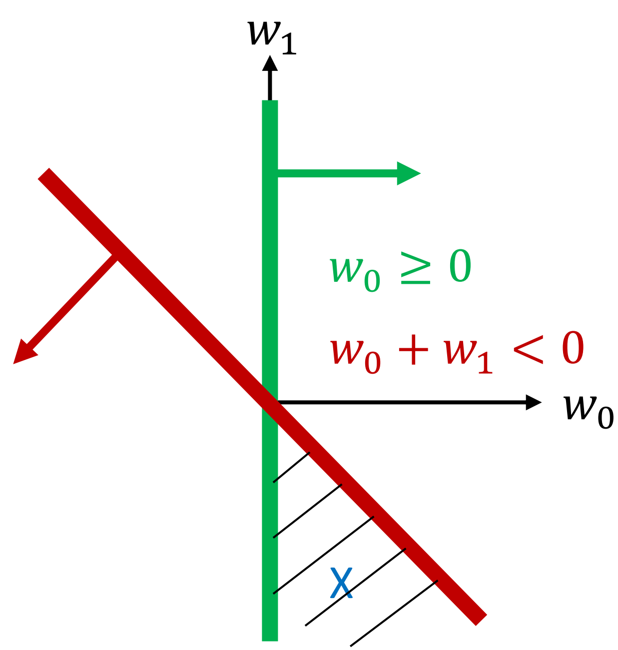
Training Set



Input Space

NOT Gate (Cont.)

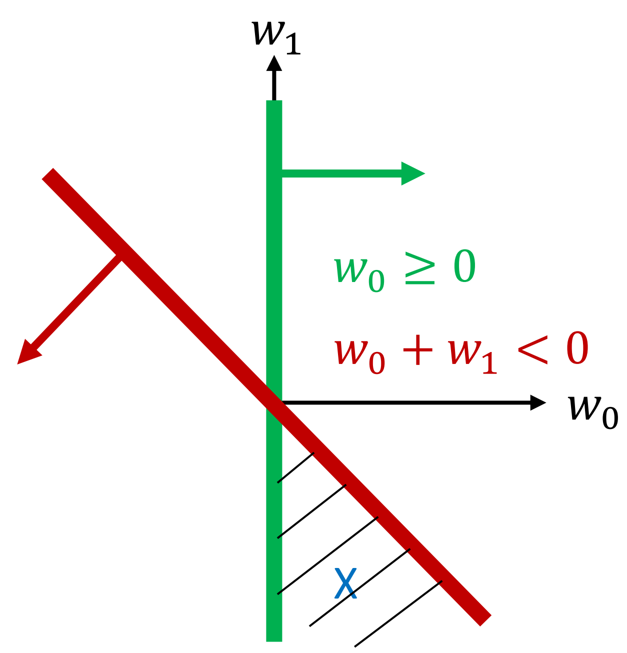
- The weights should fulfill the following conditions:
  - For $x_1 = 0$: $f(\mathbf{x}) = w_0 x_0 + w_1 x_1 \geq 0 \leftrightarrow w_0 \geq 0$
  - For $x_1 = 1$: $f(\mathbf{x}) = w_0 x_0 + w_1 x_1 < 0 \leftrightarrow w_0 + w_1 < 0$
- Solution: $w_0 = 1$ and w_1 = -2.
- Each training sample $\mathbf{x}$ specifies a half-space $\mathbf{w}$. The training example should lie within the half-space to be correctly classified.



Weight Space

NOT Gate (Cont.)

- In our example:
  - For $\mathrm{x}_0 = 1, \mathrm{x}_1 = 0, \mathrm{y} = 1$ we have $(\mathrm{w}_0, \mathrm{w}_1) \in \{\mathrm{w}: \mathrm{w}_0 \geq 0\}$
  - For $\mathrm{x}_0 = 1, \mathrm{x}_1 = 1, \mathrm{y} = 0$ we have $(\mathrm{w}_0, \mathrm{w}_1) \in \{\mathrm{w}: \mathrm{w}_0 + \mathrm{w}_1 < 0\}$
- The region satisfying all above constraints is the feasible region.
  - The weights, which classify correct all the training samples, correspond to the intersection of all half-spaces.



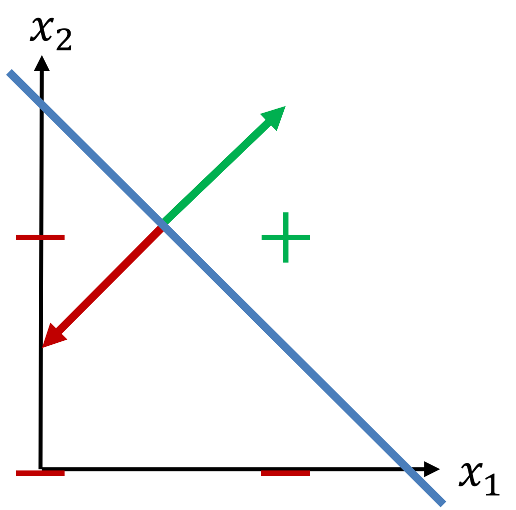
Weight Space

AND Gate

- The AND gate requires two inputs x_1 and x_2.
- To find a solution, a third auxiliary dimension is required.
  - The auxiliary input is always one.

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

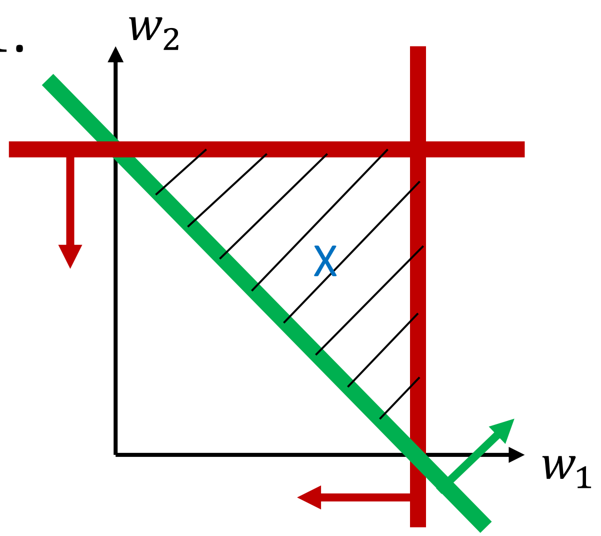
Training Set



Input Space

AND Gate (Cont.)

- Conditions for selecting the weights:
  - *For* $x_1 = 0, x_2 = 0$*:* $f(\boldsymbol{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 < 0 \leftrightarrow w_0 < 0$
  - *For* $x_1 = 0, x_2 = 1$*:* $f(\boldsymbol{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 < 0 \leftrightarrow w_0 + w_2 < 0$
  - *For* $x_1 = 1, x_2 = 0$*:* $f(\boldsymbol{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 < 0 \leftrightarrow w_0 + w_1 < 0$
  - *For* $x_1 = 1, x_2 = 1$*:* $f(\boldsymbol{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 \geq 0 \leftrightarrow w_0 + w_1 + w_2 \geq 0$
- Solution: $w_0 = -1.5$, $w_1 = 1$ and $w_2 = 1$.

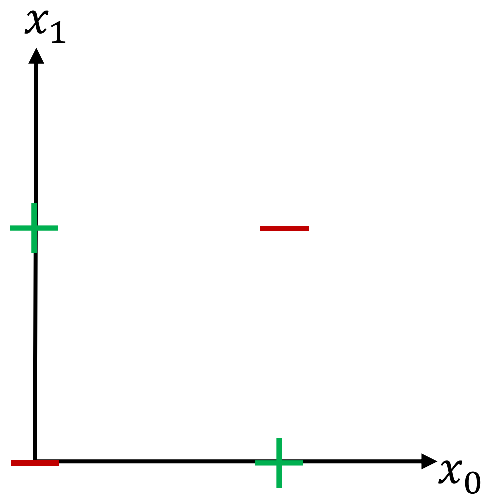


Weight Space

- The auxiliary dimension x_0 is necessary for finding a solution.
- *Is there only a single solution?*

XOR Gate

- Like AND, XOR has two input dimensions x_1 and x_2.
- *Can we rely on a linear classifier for modeling the XOR gate?*



Input Space

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

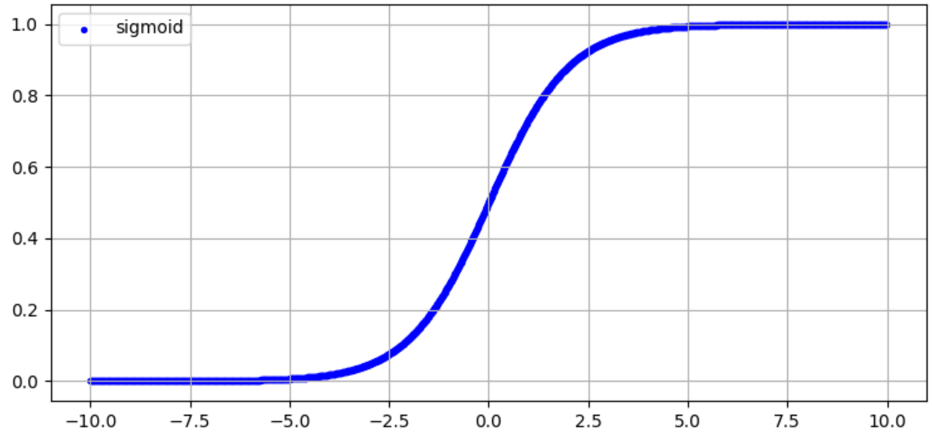
Training Set

Logistic Regression

- Logistic regression transforms the problem of linear regression to classification with the support of a function with bounded output.
- A few examples are spam mail detection, tumor classification in malignant or benign and online transaction classification to fraud or not.
- Motivation: convert continuous output of a bounded function to classification (i.e., category prediction).

Logistic Regression (Two-Class Problem)

- Consider a two-class problem: $y \in \{0, 1\}$.
- We would like to map our input $\mathbf{x}$ to one of this two values.
- Consider also the logistic function $\sigma(\cdot)$.



- The horizontal axis represents the input and the vertical axis the function's response.
- The value at 0.5 would be a well-suited threshold to decide between two classes, i.e., 0 and 1.

Logistic Regression (Cont.)

- Recall the logistic function (also known as sigmoid) equation and its derivative:
  - $\sigma(z) = \frac{1}{1+e^{-z}}$
  - $\frac{\partial \sigma}{\partial z} = \sigma(z)\big(1 - \sigma(z)\big)$, *where z is a scalar.*
- We deal with a regression problem, and we reformulate $\sigma(\cdot)$ to be parameterized by $\mathbf{w}$, i.e., our parameters. We define the modified logistic function now as:
  - $h_{\boldsymbol{w}}(\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{w}\boldsymbol{x}}}$
- The input $\mathbf{x}$ is the feature vector.
- In addition, we interpret the output of the modified logistic function $h_{\mathbf{w}}(\cdot)$ as the probability of being class 1. The function output does not have to be exactly 1 to classify the input $\mathbf{x}$ as class 1.
- We can set a threshold at 0.5.

Logistic Regression (Cont.)

- The probabilistic interpretation of the logistic function is written as:
  - $h_{\boldsymbol{w}}(\boldsymbol{x}) = p(y = 1|\boldsymbol{x}; \boldsymbol{w}) = 1 - p(y = 0|\boldsymbol{x}; \boldsymbol{w})$.
- We could rely on a set of m training pairs (input, ground-truth) and minimize the mean-squared error, like linear regression, where:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{w}}(\boldsymbol{x}) - y_i)^2 \,.$$

- Unfortunately, there is no **analytical solution** as with linear regression.
- We face a non-convex problem because of the sigmoid function.
- Instead, we can use the maximum likelihood estimation.

Logistic Regression (Cont.)

- We want to maximize the conditional probability function $p(y|\mathbf{x};\mathbf{w})$.
- The conditional maximum likelihood estimation is given by:
  - $\boldsymbol{w}^* = arg\,\max_{\boldsymbol{w}} \frac{1}{m}\sum_{i=1}^{m} log\, p(y_i|\boldsymbol{x}_i;\boldsymbol{w})$.
- The probability of $\mathbf{x}_i$ to belong in class 0 or 1, $y_i \in \{0,1\}$, given by:
  - $p(y_i = 1|\boldsymbol{x}_i) = h_{\boldsymbol{w}}(\boldsymbol{x}_i)$
  - $p(y_i = 0|\boldsymbol{x}_i) = 1 - h_{\boldsymbol{w}}(\boldsymbol{x}_i)$
- We can now rewrite the conditional maximum likelihood estimator:
  - $\boldsymbol{w}^* = arg\,\max_{\boldsymbol{w}} \frac{1}{m}\sum_{i=1}^{m} y_i\, log(h_{\boldsymbol{w}}(\boldsymbol{x}_i)) + (1-y_i)\, log\big(1 - h_{\boldsymbol{w}}(\boldsymbol{x}_i)\big)$.

Logistic Regression (Cont.)

- Instead, we minimize the negative log-likelihood (NLL). This is written as:
  - $\mathbf{w}^* = \arg\min_{\mathbf{w}} -\frac{1}{m}\sum_{i=1}^{m} y_i \log\big(h_{\mathbf{w}}(\mathbf{x}_i)\big) + (1-y_i)\log(1-h_{\mathbf{w}}(\mathbf{x}_i)).$
- This is a gradient-based optimization where we can use stochastic gradient descent.
- We assume the loss function:
  - $L(\mathbf{x}, y, \mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} -y_i \log\big(h_{\mathbf{w}}(\mathbf{x}_i)\big) - (1-y_i)\log\big(1-h_{\mathbf{w}}(\mathbf{x}_i)\big).$
- The gradient w.r.t. the parameters is given by:
  - $\nabla_{\mathbf{w}} L(\mathbf{x}, y, \mathbf{w}) = \sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)\mathbf{x}_i .$
- Logistic regression applies on the same way to multiclass problems as well.

Online example: https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

Bayes Optimal Classifier

- It is a probabilistic model that performs the most probable prediction for new sample, given the training data.
- $\mathbb{P}(y = i|x) = \frac{\mathbb{P}(x|y=i)\mathbb{P}(y=i)}{\mathbb{P}(x)} = \frac{p(x|y = i)p(y=i)}{\sum_{j=1}^{n} p(x|y = j)p(y=j)}.$
- In classification, we are interested in the class prediction. We aim for:
  - $\arg\max_{k \in \mathcal{Y}} \mathbb{P}(y = k|\mathbf{x})$.
  - We can thus drop the evidence $\mathbb{P}(\mathrm{x})$ and the posterior solution is still the same.
  - In general, we want find the x estimates that maximise the the posterior. This is known as Maximum a posteriori estimation (MAP).

Information on MAP: https://www.probabilitycourse.com/chapter9/9_1_2_MAP_estimation.php

Bayes Optimal Classifier

- Given a data point represent as feature vector $\mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$ and label space $\mathcal{Y} = \{0,1\}$, we assume that x_i is in the label space.
- The optimal Bayes classifier is defined as:
  - $f_{Bayes}(\mathbf{x}) = \arg\max_{\mathrm{k}\in\mathcal{Y}} \mathbb{P}(y = k|\mathbf{x})$
- The number of parameters increases with the number of features d, since 2^d parameters are needed to describe the probability function $p(y = k|\mathbf{x})$, for example, when $\mathbf{x} \in \{0,1\}^d$.
- With the Bayes' theorem we only need to compute the posterior distribution $p(\mathbf{x}|y = k)$:
  - $f(\mathbf{x}) = \mathbb{P}(y = k|\mathbf{x}) = \frac{\mathbb{P}(\mathbf{x}|y = k)\mathbb{P}(y=k)}{\mathbb{P}(\mathbf{x})} = \frac{p(\mathbf{x}|y = k)p(y=k)}{\sum_{j=1}^{n} p(\mathbf{x}|y = j)p(y=j)}.$
- The optimal Bayes classifier can be written as:
  - $f_{Bayes}(\mathbf{x}) = \arg\max_{k\in\mathcal{Y}} p(\mathbf{x}|y = k)p(y = k).$

Naive Bayes Classifier

- The Naive Bayes classifier assumes that the features are conditionally independent given the class:
  - $p(\mathbf{x}|y=k) = p((x_1, x_2, \dots, x_d)|y=k) = \prod_{j=1}^{d} p(x_j|y=k)$.
- Based on this assumption the Optimal Bayes classifier can be reformulated as:
  - $f_{Bayes}(x) = p(y=k) \prod_{j=1}^{d} p(x_j|y=k)$.
- Then, only $2d + 1$ parameters must be estimated.
- The required parameters are significantly reduced.

Vikramkumar, Vijaykumar, B., & Trilochan (2014). Bayes and Naive Bayes Classifier. *ArXiv, abs/1404.0933*.

Naive Bayes Classifier (Cont.)

- Numerical feature:
  - The distribution $p(x_j|y = k)$ is modelled as univariate Gaussian distribution.
  - Then, only the parameters μ_j and σ_j^2 must be estimated.
- Categorical features:
  - The distribution $p\left(x_j \middle| y = k\right) = \prod_g p_{kjg}^{[x_j=g]}$ is modelled as categorical distribution.
  - p_{kjg} is the probability that the j-th feature hast the value g in class k.
  - Therefore, the frequency of $x_j = g$ can be counted.

Online example: https://www.datacamp.com/tutorial/naive-bayes-scikit-learn

Naive Bayes Classifier (Cont.)

- E-mail spam detector example.
  - Single words can be counted as feature to detect spam mails.
  - The classifier can be trained for each user individually. The probabilities assigned to each word are different for each user.
  - <u>Independence assumption</u>: the occurrence of two words in an e-mail is not correlated.
  - For example, the words “free money” occur with a high probability in spam e-mails, but the single words “free” and “money” also occur in non-spam e-mails with a high probability.
  - Individual letters of words that are frequently used in spam e-mails can be replaced with similar letters so that they are readable by humans but not recognized by spam filters.

Linear Discriminant Analysis

- Again, a data point represent as feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ and label space $\mathcal{Y} = \{0,1\}$ is given.
- We now make the following assumptions:
  - $p(y=1) = p(y=0) = \frac{1}{2}$.
  - $p(\mathbf{x}|y=k)$ is a Gaussian distribution, where the covariance matrix is the same for both class labels, i.e. $\Sigma_k = \Sigma \; \forall k$.
  - $p(\mathbf{x}|y=k) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu}_k)\right)$ with $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1 \in \mathbb{R}^d$ and Σ is the covariance matrix.

Tharwat, A., Gaber, T., Ibrahim, A., & Hassanien, A.E. (2017). Linear discriminant analysis: A detailed tutorial. *AI Commun., 30*, 169-190.
Online reference: https://people.revoledu.com/kardi/tutorial/LDA/Numerical%20Example.html

Linear Discriminant Analysis (Cont.)

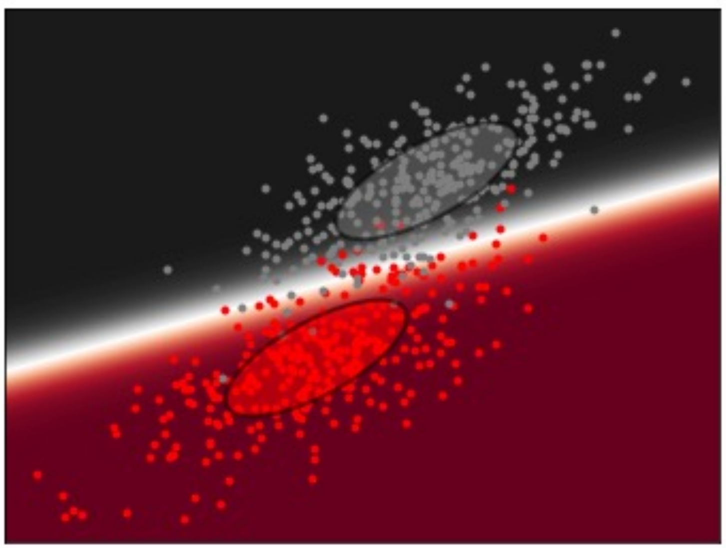
- Remember with Bayes rule the Bayes Optimal Classifier can be written as: $f_{Bayes}(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} p(\mathbf{x}|y = k)p(y = k)$.
- In the binary case, $f_{Bayes}(x) = 1$ under the following condition:
  - $\log\left(\frac{p(x|y = +1)p(y=1)}{p(x|y = -1)p(y=0)}\right) > 0$, which is called the *log-likelihood ratio*.
- Under the made assumptions the log-likelihood ratio becomes:
  - $\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_0)^T\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_0) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^T\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_1)$.
- This can be reformulated as a linear classifier:
  - $f(x) = w^T x + b$ with $w = (\mu_1 - \mu_0)^T\Sigma^{-1}$ and $b = \frac{1}{2}(\mu_0^T\Sigma^{-1}\mu_0 - \mu_1^T\Sigma^{-1}\mu_1)$.
- Now only the parameters μ_0, μ_1 and Σ must be estimated from the training data.

Online example: https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/

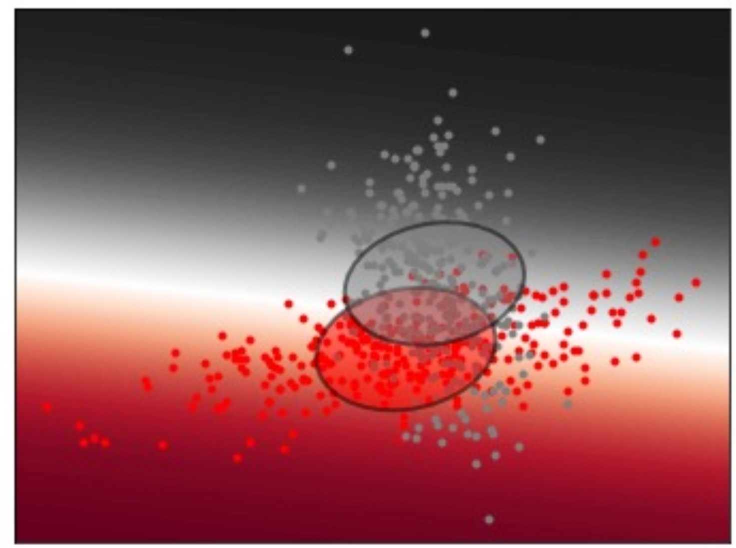
Linear Discriminant Analysis (Cont.)

- Covariance matrix comparison

Same covariance matrix for both classes.



Different covariance matrix for both classes.



Study Material

- *The Elements of Statistical Learning, Trevor Hastie et. al, Chapter 3.*
- *Understanding Machine Learning, Shai Shalev-Schwarz, Chapter 9, Section 9.2.*
- *Pattern Recognition and Machine Learning, Christopher Bishop, Chapter 3.*

Next Lecture

Performance Evaluation