# Learning

# Clustering

**Clustering** is the partitioning of a data set into meaningful or useful groups, based on distance between data points.

> **Example**: Given a set of oranges and lemons, cluster into different orange and lemon varieties.

# K-means clustering

**K-means clustering** is a simple algorithm to find clusters:

1. Pick $K$ random points as cluster centre positions
2. Assign each point to its nearest centre (In the unlikely event of a tie, break the tie in some way).
3. Move each centre to the mean of its assigned points
4. If the centres moved, go back to step 2.

## Evaluation of K-means clustering

To measure the quality of a K-means clustering solution, we can use a **sum-squared error function** — that is, the sum of squared distances of each point from its cluster centre.

Let $z_{kn}$ be defined as:

$$z_{kn} = \begin{cases} 1 & \text{if point } x_n \text{ belongs to cluster } k \\ 0 & \text{otherwise} \end{cases}$$

Then:

$$E = \sum_{k=1}^{K} \sum_{n=1}^{N} z_{kn} || \underbrace{\mathbf{x}_n - \mathbf{m}_k}_{r_D(\mathbf{x}_n, \mathbf{m}_k)} ||^2$$

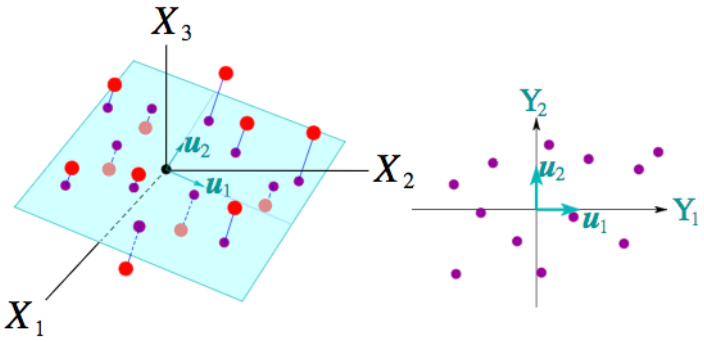
$$\begin{aligned} & & \mathbf{m}_k \text{ is the centre of cluster } k \\ \text{where} & & \mathbf{x}_n = (x_{n1}, \ldots, x_{nD})^\mathsf{T} \\ & & \mathbf{m}_k = (m_{k1}, \ldots, m_{kD})^\mathsf{T} \end{aligned}$$

The sum squared error decreases as we increase $K$. ($E \to 0$ as $K \to N$)

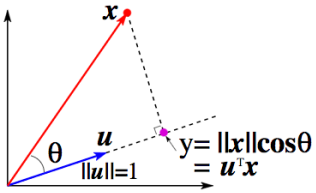
# Dimensionality reduction

High-dimensional data is often difficult to understand and visualise. It is often a good idea to consider dimensionality reduction of data to improve visualisation.

As shown in the example below, **dimensionality reduction** is the process of reducing the number of random variables under consideration. This is done by projecting each sample in 3D onto a 2D plane.




Recall orthogonal projection of data onto an axis:




# Optimal projection of 2D data onto 1D

For the optimal projection of 2D data to 1D, we need a mapping $y_n = \mathbf{u}^\mathsf{T}\mathbf{x}_n = u_1 x_{n1} + u_2 x_{n2}$ where the is optimal $\mathbf{u}$ is when $\mathrm{Var}\,[y]$ is maximised.

This optimal $\mathbf{u}$ is called the **principal component axis**.

# General case

Mapping $D$-dimensional data to a **principal component axis** $\mathbf{u} = (u_1, \ldots, u_D)^\mathsf{T}$ that maximises $\mathrm{Var}\,[y]$:

$$y_n = \mathbf{u}^\mathsf{T}\mathbf{x}_n = u_1 x_{n1} + \cdots + u_D x_{nD}$$

$\mathbf{u}$ is given as the eigenvector with the largest eigenvalue of the **covariance matrix**, $\mathbf{\Sigma}$.

## Covariance matrix

Each entry of the covariance matrix is given by:

$$\begin{aligned}\mathbf{\Sigma}_{ij} &= \mathrm{Cov}\,[\mathbf{x}_i, \mathbf{x}_j] \\ &= \mathbb{E}\,[(\mathbf{x}_i - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_j)] \\ &= \frac{1}{N-1}\sum_{n=1}^{N}(x_{ni} - \mu_i)(x_{nj} - \mu_j), \qquad \mu_i = \frac{1}{N}\sum_{n=1}^{N} x_{ni}\end{aligned}$$

Giving us the following matrix equality:

$$\boldsymbol{\Sigma} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^\mathsf{T}, \qquad \boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

# Classification and nearest neighbours

## Types of learning problems

| Data | Input | Output | Type of problem | Type of learning |
|---|---|---|---|---|
| $\mathbf{x}$ | $\{\mathbf{x}\}$ | groups (subsets) | clustering | unsupervised |
| $(\mathbf{x}, y)$ | $\mathbf{x}$ | $y$: discrete category | classification | supervised |

Where $\mathbf{x} = (x_1, \ldots, x_D)^\mathsf{T}$ is a feature vectors and $y$ is a target vector or scalar.

## Classification

- The data has a feature vector $\mathbf{x} = (x_1, \ldots, x_D)^\mathsf{T}$ and a label $c \in \{1, \ldots, C\}$.
- Given a **training set** - a set of $N$ feature vectors and their labels $(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)$, we have to use a learning algorithm to train a classifier from a training set.
- The **test set** is a set of feature vectors which the classifier must assign labels. It is used for evaluation.
- An **error function** tells us how accurate the classifier is. One option is to count the number of misclassifications:

$$\text{Error rate} = \frac{\text{No. of misclassified samples}}{\text{No. of test samples}}$$

## Nearest-neighbour classifier

**Nearest-neighbour classification** is labeling a test example to have the label of the closest training example.

## K-nearest neighbour classifier

**K-nearest neighbour classification** is finding the K closest points in the training set to the test example, then classifying the test example by using a majority vote of the K class labels.

- Training a K-nearest neighbour classifier is simple — just store the training set.
- Classifying a test example requires finding the K nearest training examples, which can be computationally demanding if the training set is large (since we need to compute the Euclidean distance between the test example and every training example).

### K-NN classification algorithm

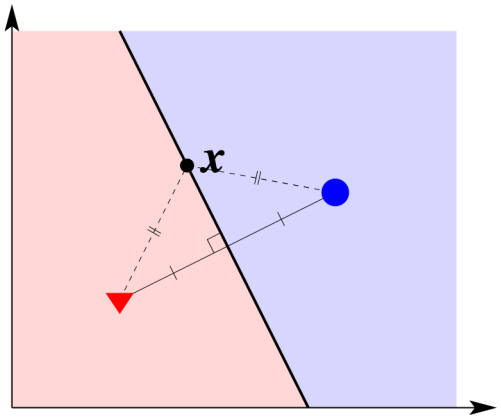
For each test example $\mathbf{z} \in z$:

- Compute the distance $r(\mathbf{z}, \mathbf{x})$ between $\mathbf{z}$ and each training example $(\mathbf{x}, c) \in X$

- Select $U_k(\mathbf{z}) \subseteq X$, the set of the $k$ nearest training examples to $\mathbf{z}$
- Decide the class of $\mathbf{z}$ by the majority voting

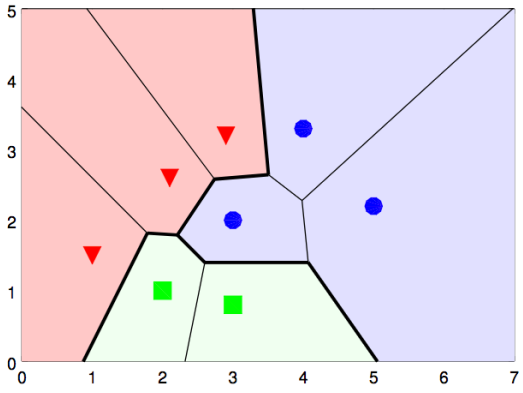
## Geometry of nearest-neighbour classifier

- A **decision boundary** is a boundary that partitions the vector space into subsets of different classes.

> **Example**: The decision boundary for a vector space with two feature vectors in two different classes would look like:
>
> 
> 

- **Decision regions** are the regions that are separated by the decision boundaries.

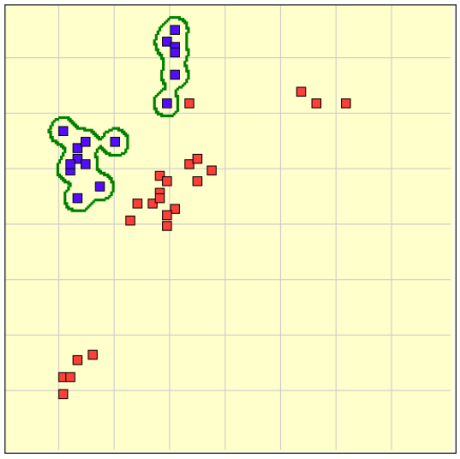
  In the diagram above, the decision regions are the blue and red regions.

> **Example**: In a vector space with more feature vectors and classes, the decision boundaries and decision regions can become more complicated:
>
> 
> 

## Generalisation and over-fitting

- **Overfitting** is the tuning of a classifier too closely to the training set. This can often reduce the accuracy of classification on the test set.

> **Example**: The decision boundary of the below classifier is too overfitting and poorly generalised.



# Statistical pattern recognition and optimisation

## Rules of probability

For random variables $X \in \{x_i\}_{i=1}^{L}$ and $Y \in \{y_j\}_{j=1}^{M}$:

- **Product rule**:

$$\begin{aligned}\mathbb{P}(Y = y_j, X = x_i) &= \mathbb{P}(Y = y_j | X = x_i)\,\mathbb{P}(X = x_i) \\ &= \mathbb{P}(X = x_i | Y = y_j)\,\mathbb{P}(Y = y_j)\end{aligned}$$

  Abbreviation:

$$\begin{aligned}\mathbb{P}(Y, X) &= \mathbb{P}(Y|X)\,\mathbb{P}(X) \\ &= \mathbb{P}(X|Y)\,\mathbb{P}(Y)\end{aligned}$$

  $X$ and $Y$ are **independent** iff:

  - $\mathbb{P}(X|Y) = \mathbb{P}(X)$ and $\mathbb{P}(Y|X) = \mathbb{P}(Y)$
  - $\mathbb{P}(X, Y) = \mathbb{P}(X)\,\mathbb{P}(Y)$, which follows from the previous statement.

- **Sum rule**:

$$\mathbb{P}(X = x_i) = \sum_{j=1}^{M} \mathbb{P}(X = x_i, Y = y_j)$$

  Abbreviation:

$$\mathbb{P}(X) = \sum_{Y} \mathbb{P}(X, Y)$$

  - $\mathbb{P}(X)$ is referred to as the **marginal probability** of $X$.
  - $\sum_Y \mathbb{P}(X, Y)$ is the **marginalisation** of the joint probability over $Y$.

## Bayes' theorem

$$\mathbb{P}(H|E) = \frac{\mathbb{P}(E|H)\,\mathbb{P}(H)}{\mathbb{P}(E)}$$

## Bayes' decision rule

Suppose we have a random variable $C \in \{1, \ldots, K\}$, where each element $k$ represents a class label. Let $C_k$ denote $C = k$ and suppose we have an input feature vector matrix $\mathbf{x}$, with features $x_i$.

Then the most probable class (**MAP** - Maximum A Posteriori) for vector $\mathbf{x}$ is given by the class label which generates the maximum posterior probability — that is:

$$\begin{aligned} k_{\max} &= \arg\max_{k \in C} \mathbb{P}\left(C_k|\mathbf{x}\right) \\ &= \arg\max_{k \in C} \mathbb{P}\left(\mathbf{x}|C_k\right)\mathbb{P}\left(C_k\right) \end{aligned}$$

Where:

$$\overbrace{\mathbb{P}\left(C_k|\mathbf{x}\right)}^{\text{posterior}} = \frac{\overbrace{\mathbb{P}\left(\mathbf{x}|C_k\right)}^{\text{likelihood}} \cdot \overbrace{\mathbb{P}\left(C_k\right)}^{\text{prior}}}{\underbrace{\mathbb{P}\left(\mathbf{x}\right)}_{\text{evidence}}}$$

$$= \frac{\mathbb{P}\left(\mathbf{x}|C_k\right) \cdot \mathbb{P}\left(C_k\right)}{\sum_{j=1}^{K} \mathbb{P}\left(\mathbf{x}|C_j\right)\mathbb{P}\left(C_j\right)} \qquad \text{By the law of total probability}$$

Often, the evidence $\mathbb{P}\left(\mathbf{x}\right)$ will be the same for all classes and therefore it might not be necessary to consider it when finding the most probable class since:

$$\mathbb{P}\left(C_k|\mathbf{x}\right) \propto \mathbb{P}\left(\mathbf{x}|C_k\right)\mathbb{P}\left(C_k\right)$$

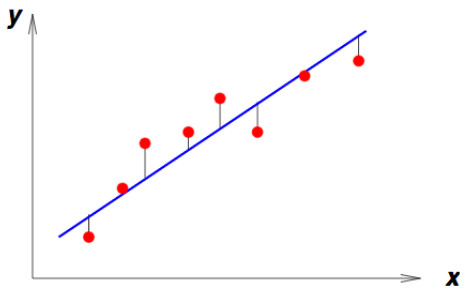
# Least square error line fitting

This is an optimisation problem, also called finding the **line of best fit**.

If the least square error line is given by $\hat{y}_n = ax_n + b$ (also called the **regression line** on $y$ with $x$), then we need to find:

$$\min_{a,b} \frac{1}{N} \sum_{n=1}^{N} (\hat{y}_n - y_n)^2$$

That is, find $a$ and $b$ for the line $\hat{y}_n = ax_n + b$ such that the total variance of all points (along the $y$ axis) is minimized, as depicted here:

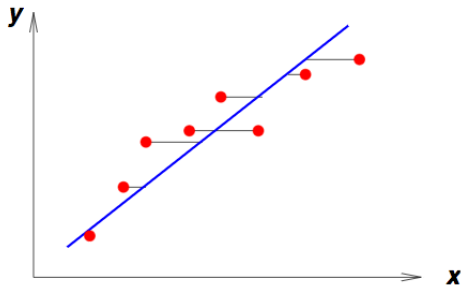



## Regression on $x$ with $y$

Sometimes it may be better to have a least square error line in the form $\hat{x} = cy_n + d$. This is called a regression line on $x$ with $y$, where $c$ and $d$ are given by:

$$\min_{c,d} \frac{1}{N} \sum_{n=1}^{N} (\hat{x}_n - x_n)^2$$

That is, find $c$ and $d$ for the line $\hat{x} = cy_n + d$ such that the total variance of all points (along the $x$ axis) is minimized, as depicted here:




# Iterative optimisation

**Iterative optimisation** is used on optimisation problems that don't have a closed-form solution (might not converge, such as $K$-means clustering)

## Iterative optimisation method

1. Choose an initial point $\mathbf{x}_0$, and make $t = 0$
2. Choose $\mathbf{x}_{t+1}$ based on an update formula for $\mathbf{x}_t$
3. Increment $t$, so $t \leftarrow t + 1$ and go to step 2 unless a stopping criterion is met.

**Gradient descent** is an example of an iterative optimisation method.

## Gradient descent

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_t}$$

# Naïve Bayes classification

For a feature vector $\mathbf{x} = (x_1, \ldots, x_D)^\mathsf{T}$, we wish to know $\mathbb{P}(\mathbf{x}|C_k) = \mathbb{P}(x_1, x_2, \ldots, x_D|C_k)$.

With the chain rule of probabilities:

$$\begin{aligned}\mathbb{P}(\mathbf{x}|C_k) &= \mathbb{P}(x_1, x_2, \ldots, x_D|C_k) \\ &= \mathbb{P}(x_1|C_k) \cdot \mathbb{P}(x_2|x_1, C_k) \cdots \mathbb{P}(x_D|x_{D-1}, \ldots, x_1, C_k)\end{aligned}$$

Naïve Bayes classification **naïvely** assumes that every feature $x_i$ is conditionally independent of every other $x_j : i \neq j$ given $C_k$, i.e:

$$\mathbb{P}(x_i|C_k, x_j) = \mathbb{P}(x_i|C_k), \qquad i \neq j$$

Therefore it follows that after this *naïve* assumption:

$$\mathbb{P}(\mathbf{x}|C_k) = \prod_{i=1}^{D} \mathbb{P}(x_i|C_k)$$

Or:

$$\mathbb{P}(C_k|\mathbf{x}) = \frac{\mathbb{P}(C_k)\prod_{i=1}^{D}\mathbb{P}(x_i|C_k)}{\prod_{i=1}^{D}\mathbb{P}(x_i)}$$

$$\text{Since the denominator remains constant for a given input:}$$

$$\propto \mathbb{P}(C_k)\prod_{i=1}^{D}\mathbb{P}(x_i|C_k)$$

Therefore, when applying **maximum likelihood estimation** (**MLE**) of priors and likelihoods (using Naïve Bayes assumption for the likelihoods), we get:

$$k_{\max} = \arg\max_k \mathbb{P}(\mathbf{x}|C_k)\,\mathbb{P}(C_k)$$

## Advantages and disadvantages

| Advantages | Disadvantages |
|---|---|
| Simplifies calculation of posterior probabilities | If there are no occurences of a class label and a certain feature together, then the entire posterior probability will be $0$, since it is a product of all posteriors. |

# Text classification with Naïve Bayes

Given a document $\mathcal{D}$ with a fixed set of classes $C = \{1, \ldots, K\}$, we classify $\mathcal{D}$ as the MAP class:

$$\begin{aligned} k_{\max} &= \arg\max_{k \in C} \mathbb{P}(C_k|\mathcal{D}) \\ &= \arg\max_{k \in C} \mathbb{P}(\mathcal{D}|C_k)\,\mathbb{P}(C_k) \end{aligned}$$

But how do we represent $\mathcal{D}$, and how do we estimate $\mathbb{P}(\mathcal{D}|C_k)$ and $\mathbb{P}(C_k)$?

## Representation of $\mathcal{D}$

- **Sequence of words** model: $\mathcal{D} = (X_1, X_2, \ldots, X_n)$

  | Advantages | Disadvantages |
  |---|---|
  | Maintains order and position of words | Computationally expensive |
  | | Difficult to train |

- **Set of words** (**Bag-of-words**) model:

  In the bag-of-words model, we only consider the words in a document that appear in a predefined vocabulary $V$.

  > **Example**: If classifying scam emails, we would have a vocabulary consisting of words like:

> *casino, congratulations, draws, lotto, first, winner, true, lottery, million*

| Advantages | Disadvantages |
|---|---|
| Easier to train | Ignores the context of words |
| Not computationally expensive | Loses the ordering and position of words |

## Multinomial document model

- Document represented by an integer feature vector, whose elements indicate the frequency of the corresponding word in the document:

  $$\mathbf{x} = (x_1, \ldots, x_D) \qquad x_i \in \mathbb{N}_0$$

  Where $D$ is either:

  - The number of unique words in the document (**sequence-of-words** model)
  - The number of words in the vocabulary $V$ (**bag-of-words** model), $|V|$

$$\mathbb{P}(\mathcal{D}|C_k) = \mathbb{P}(\mathbf{x}|C_k) \propto \prod_{t=1}^{|V|} \mathbb{P}(w_t|C_k)^{x_t}$$

---

**Note**: Remember that when classifying an unlabelled document $\mathcal{D}$, we actually want the posterior — $\mathbb{P}(C_k|\mathcal{D})$.

## Bernoulli document model

- Document represented by a binary feature vector, whose elements indicate absence or presence of the corresponding word in the document

  $$\mathbf{b} = (b_1, \ldots, b_D) \qquad b_i \in \{0, 1\}$$

  Where $\mathcal{D}$ is the same as previously described for the multinomial document model.

$$\mathbb{P}(\mathcal{D}|C_k) = \mathbb{P}(\mathbf{b}|C_k) = \prod_{t=1}^{|V|} [b_t \mathbb{P}(w_t|C_k) + (1 - b_t)(1 - \mathbb{P}(w_t|C_k))]$$

  Where $\mathbb{P}(w_t|C_k) = \frac{n_k(w_t)}{N_k}$.

- $n_k(w_t)$ - number of documents of class $k$ in which $w_t$ is observed
- $N_k$ - total documents in class $k$

---

**Note**: Remember that when classifying an unlabelled document $\mathcal{D}$, we actually want the posterior — $\mathbb{P}(C_k|\mathcal{D})$.

# Multivariate Gaussians and classification

## Gaussian distribution

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

- Symmetric and centered about $\mu$
- $\sigma$ increase $\rightarrow$ Gets flatter and wider
- $\sigma$ decrease $\rightarrow$ Gets taller and narrower

# Parameter estimation from data

- Sample mean and sample variance estimates:

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad \sigma^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \mu)^2$$

- Maximum likelihood estimates:

$$\mu_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad \sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{\mathrm{ML}})^2$$

# Multivariate form

For a $D$-dimensional vector $\mathbf{x} = (x_1, \ldots, x_D)^\mathsf{T}$:

$$\mathbb{P}\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \frac{1}{(2\pi)^{\frac{D}{2}}\left|\boldsymbol{\Sigma}\right|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

- $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_D)^T$ is the **mean vector**
- $\boldsymbol{\Sigma} = (\sigma_{ij})$ is the **covariance matrix**

Note that the 1-dimensional Gaussian is a special case of this PDF.

## Parameter estimation

- $\boldsymbol{\Sigma} = \mathbb{E}\left[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\right] = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^\mathsf{T}$
- $\boldsymbol{\Sigma}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu}_{\mathrm{ML}})(\mathbf{x}_n - \boldsymbol{\mu}_{\mathrm{ML}})^\mathsf{T}$

# Covariance

The covariance of variables $X$ and $Y$ is:

$$\mathrm{Cov}\left[X, Y\right] = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

# Covariance matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1D} \\ \vdots & \ddots & \vdots \\ \sigma_{D1} & \cdots & \sigma_{DD} \end{bmatrix} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^\mathsf{T}$$

Where $\mathbf{x}_n = (x_{n1}, \ldots, x_{nD})^\mathsf{T}$ is a feature vector (an observation of $D$ variables) in:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1(D-1)} & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2(D-1)} & x_{2D} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{(N-1)1} & x_{(N-1)2} & \cdots & x_{(N-1)(D-1)} & x_{(N-1)D} \\ x_{N1} & x_{N2} & \cdots & x_{N(D-1)} & x_{ND} \end{bmatrix}$$

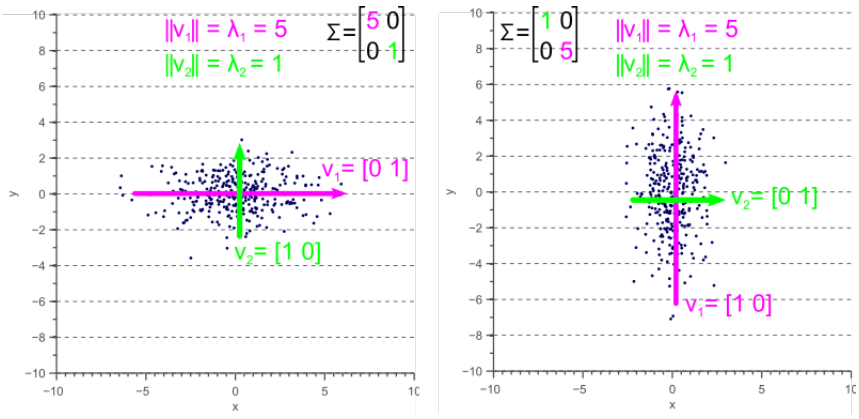
**Note**: Diagonals are the variances.

> **Example**: Consider the following test data with variables length, width and height of a certain object:
>
> $$\mathbf{X} = \begin{bmatrix} \text{Length} & \text{Width} & \text{Height} \\ 4.0 & 2.0 & 0.60 \\ 4.2 & 2.1 & 0.59 \\ 3.9 & 2.0 & 0.58 \\ 4.3 & 2.1 & 0.62 \\ 4.1 & 2.2 & 0.63 \end{bmatrix}$$
>
> Then we have $\boldsymbol{\mu} = (4.10, 2.08, 0.604)^{\mathsf{T}}$ and
>
> $$\boldsymbol{\Sigma} = \begin{bmatrix} & \text{Length} & \text{Width} & \text{Height} \\ \text{Length} & 0.025 & 0.0075 & 0.00175 \\ \text{Width} & 0.0075 & 0.0070 & 0.00135 \\ \text{Height} & 0.00175 & 0.00135 & 0.00043 \end{bmatrix}$$
>
> Which tells us that the covariance between the length and height is $0.00175$, etc.

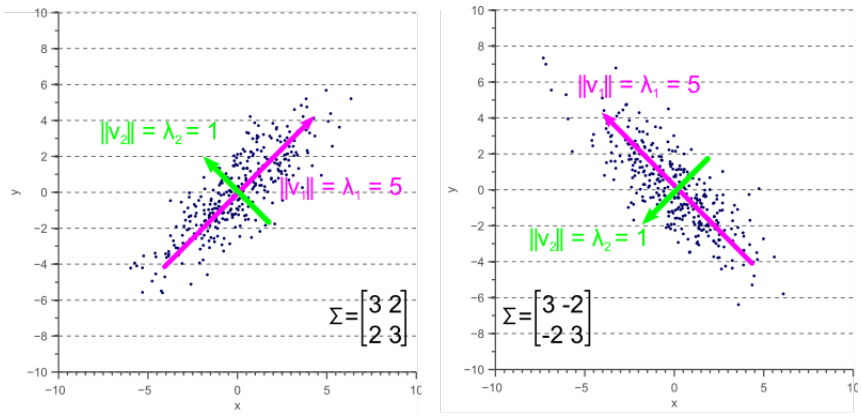
## Geometric interpretation

The covariance matrix defines the shape of the data.

- Diagonal spread of the data is captured by the covariance, while axis-aligned spread is captured by the variance.
- The center of the spread is given by the mean vector.

> Example: If $\boldsymbol{\Sigma}$ is diagonal (such that the covariances are $0$), then the variances must be equal to the eigenvalues:
>
> 
> 

> But if $\Sigma$ isn't diagonal, the eigenvalues still represent the variance magnitude in the direction of the largest spread of the data, and the variance components of the covariance matrix still represent the variance magnitude in the direction of the $x$ and $y$ axis.
>
> But since the data isn't axis-aligned, these values aren't the same anymore:




# Problems with estimation of covariance matrix

- $\mathbf{\Sigma}^{-1}$ becomes unstable when $|\mathbf{\Sigma}|$ is small.
  - One solution is the reduce the dimensionality with PCA
  - Another solution is regularisation — adding a small positive number to the diagonal elements:

$$\mathbf{\Sigma} \leftarrow \mathbf{\Sigma} + \epsilon \mathbf{I}$$

# Diagonal covariance matrix

If $\Sigma$ is diagonal, then:

$$\begin{aligned}
\mathbb{P}\left(\mathbf{x}|\boldsymbol{\mu}, \mathbf{\Sigma}\right) &= \frac{1}{(2\pi)^{\frac{D}{2}} |\mathbf{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}} \mathbf{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})} \\
&= \prod_{i=1}^{D} \mathcal{N}\left(x_i; \mu_i, \sigma_{ii}^2\right) \\
&= \prod_{i=1}^{D} \frac{1}{\sqrt{2\pi\sigma_{ii}^2}} e^{\frac{-(x_i-\mu_i)^2}{2\sigma_{ii}^2}}
\end{aligned}$$

# Bayes' theorem and univariate Gaussians

Remember that:

$$\begin{aligned}
\mathbb{P}\left(C_k|x\right) &\propto \mathbb{P}\left(x|C_k\right)\mathbb{P}\left(C_k\right) \\
&\propto \mathcal{N}\left(x; \mu, \sigma^2\right) \cdot \mathbb{P}\left(C_k\right) \\
&\propto \frac{\mathbb{P}\left(C_k\right)}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}
\end{aligned}$$

## Log likelihood

$$
\begin{aligned}
LL(x|C_k) &= LL(x; \mu, \sigma^2) \\
&= \ln \mathcal{N}\left(x; \mu, \sigma^2\right) \\
&= -\frac{1}{2}\left[\ln(2\pi) + \ln(\sigma^2) + \frac{(x-\mu)^2}{\sigma^2}\right]
\end{aligned}
$$

### Log posterior probability

$$
\begin{aligned}
\ln \mathbb{P}\left(C_k|x\right) &\propto \ln \mathbb{P}\left(x|C_k\right) + \ln \mathbb{P}\left(C_k\right) \\
&\propto LL(x|C_k) + \ln \mathbb{P}\left(C_k\right) \\
&\propto -\frac{1}{2}\left[\ln(2\pi) + \ln(\sigma^2) + \frac{(x-\mu)^2}{\sigma^2}\right] + \ln \mathbb{P}\left(C_k\right)
\end{aligned}
$$

## Multivariate Gaussian classifier

### Log likelihood

$$
\begin{aligned}
LL(\mathbf{x}|C_k) &= LL(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\
&= \ln \mathbb{P}\left(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \\
&= -\frac{1}{2}\left[D\ln(2\pi) + \ln|\boldsymbol{\Sigma}| + (\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]
\end{aligned}
$$

### Log posterior probability

$$
\begin{aligned}
\ln \mathbb{P}\left(C_k|\mathbf{x}\right) &\propto \ln \mathbb{P}\left(\mathbf{x}|C_k\right) + \ln \mathbb{P}\left(C_k\right) \\
&\propto LL(\mathbf{x}|C_k) + \ln \mathbb{P}\left(C_k\right) \\
&\propto -\frac{1}{2}\left[D\ln(2\pi) + \ln|\boldsymbol{\Sigma}| + (\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] + \ln \mathbb{P}\left(C_k\right)
\end{aligned}
$$

## Performance measures

- Accuracy (correct classification rate)

  $\text{accuracy} = 1 - \text{error rate}$
- Confusion matrix

# Discriminant functions

## Decision regions

- Given an unseen point $\mathbf{x}$, we assign the class $k = \arg\max_k \mathbb{P}\left(C_k|\mathbf{x}\right)$.
- The $\mathbf{x}$-space can be regarded as being divided into decision regions $\mathcal{R}_k$ such that a point falling in $\mathcal{R}_k$ is assigned to class $C_k$.

  These decision regions $\mathcal{R}_k$ can consist of several disjoint regions, each associated with class $C_k$.
- The boundaries between these regions are called decision boundaries.

## Misclassification

To estimate the probability of misclassification, we need to consider the two ways that a points can be classified wrongly:

1. Assigning $x$ to $C_1$ when it belongs to $C_2$
2. Assigning $x$ to $C_2$ when it belongs in $C_1$

The probability of the total error is then:

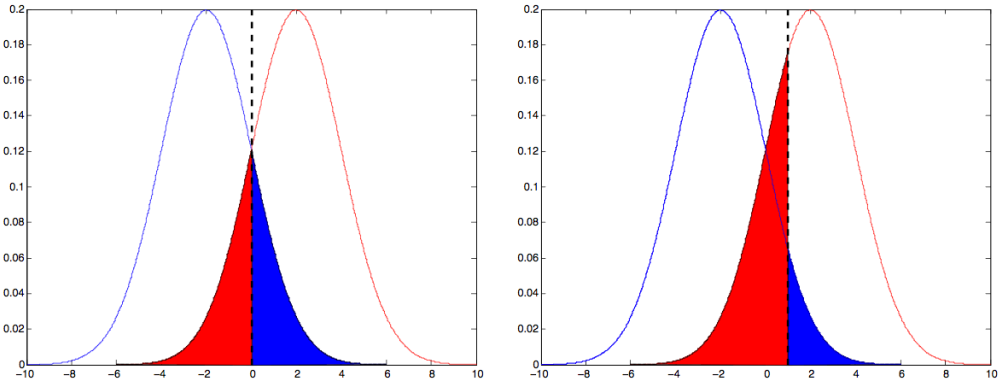
$$\mathbb{P}\left(\text{error}\right) = \mathbb{P}\left(x \in \mathcal{R}_2, C_1\right) + \mathbb{P}\left(x \in \mathcal{R}_1, C_2\right)$$

With conditional probabilities, we get:

$$\mathbb{P}\left(\text{error}\right) = \mathbb{P}\left(x \in \mathcal{R}_2 | C_1\right) \mathbb{P}\left(C_1\right) + \mathbb{P}\left(x \in \mathcal{R}_1 | C_2\right) \mathbb{P}\left(C_2\right)$$

# Overlapping Gaussians

Connsider the overlapping Gaussian PDFs shown below.



Two possible decision boundaries are shown by the dashed line. The decision boundary on the left hand plot is optimal, assuming equal priors.

**Note**: The overall probability of error is given by the area of the shaded regions under the PDFs — that is:

$$\mathbb{P}\left(\text{error}\right) = \int_{\mathcal{R}_2} \mathbb{P}\left(x | C_1\right) \mathbb{P}\left(C_1\right) \, \mathrm{d}x + \int_{\mathcal{R}_1} \mathbb{P}\left(x | C_2\right) \mathbb{P}\left(C_2\right) \, \mathrm{d}x$$

# Discriminant functions

For a set of $K$ classes, we have a set of $K$ discriminant functions $y_k(\mathbf{x})$, one for each class. Data point $\mathbf{x}$ is assigned to class $C_k$ if:

$$y_k(\mathbf{x}) > y_l(\mathbf{x}) \qquad \forall l \neq k$$

In other words, assign $\mathbf{x}$ to the class $C_k$ whose discriminant function $y_k(\mathbf{x})$ is biggest.

One example of a discriminant function we've seen is simply the log posterior probability of class $C_k$ given a data point $\mathbf{x}$.

## Optimal discriminant functions

- Classifying a point as the class with the largest log posterior probability corresponds to the decision rule which minimises the probability of misclassification.

  In this sense, MAP classification forms an optimal discriminant function.

- A **decision boundary** occurs at points in the input space where discriminant functions for each class are equal.

  If the region of input space classified as class $C_k$ and the region classified as $C_l$ are contiguous, then the decision boundary separating them is given by:

$$y_k(\mathbf{x}) = y_l(\mathbf{x})$$

> **Example**: The form of the discriminant function when using a Gaussian PDF is given as:
>
> $$\begin{aligned} y_k(\mathbf{x}) &= \ln \mathbb{P}(C_k|\mathbf{x}) = \ln \mathbb{P}(\mathbf{x}|C_k) + \ln \mathbb{P}(C_k) \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\mathsf{T}\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_k| + \ln \mathbb{P}(C_k) \underbrace{- \frac{D}{2}\ln(2\pi)}_{\text{constant}} \\ &\quad \text{Since the term is constant for each class, we can ignore it:} \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\mathsf{T}\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_k| + \ln \mathbb{P}(C_k) \end{aligned}$$
>
> Sometimes all of the classes may share the same covariance matrix — that is, $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma} \quad \forall C_k$.
>
> If this is the case, the term $-\frac{1}{2}\ln|\boldsymbol{\Sigma}_k|$ can be ignored since it is constant for all classes:
>
> $$\begin{aligned} y_k(\mathbf{x}) &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) + \ln \mathbb{P}(C_k) \\ &\quad \text{Matrix-vector expansion gives:} \\ &= -\frac{1}{2}\left[\mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\mathbf{x} - \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \boldsymbol{\mu}_k^\mathsf{T}\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_k^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k\right] + \ln \mathbb{P}(C_k) \\ &\quad \text{Note that } \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\mathbf{x} \text{ is constant for all classes, and can be ignored.} \\ &\quad \text{Additionally, for symmetric matrix } \mathbf{M} \text{ and vectors } \mathbf{a} \text{ and } \mathbf{b}, \\ &\quad \text{we have } \mathbf{a}^\mathsf{T}\mathbf{M}\mathbf{b} = \mathbf{b}^\mathsf{T}\mathbf{M}\mathbf{a}. \\ &\quad \text{Therefore, } \mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k = \boldsymbol{\mu}_k^\mathsf{T}\boldsymbol{\Sigma}^{-1}\mathbf{x} \text{ and we can simplify further:} \\ &= \boldsymbol{\mu}_k^\mathsf{T}\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln \mathbb{P}(C_k) \\ &\quad \text{If we define } \mathbf{w}_k^\mathsf{T} = \boldsymbol{\mu}_k^\mathsf{T}\boldsymbol{\Sigma}^{-1} \text{ and } w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln \mathbb{P}(C_k), \\ &\quad \text{we can see that the discriminant function is linear in } \mathbf{x}\text{:} \\ &= \mathbf{w}_k^\mathsf{T}\mathbf{x} + w_{k0} \\ &\quad \text{Where } \mathbf{w}_k \text{ is called the weight vector and } w_{k0} \text{ is the bias for class } C_k. \end{aligned}$$

## Log odds and two-class problems

For two-class problems, we want to find a single decision boundary between the class regions and hence a single discriminant function $y(\mathbf{x})$ such that:

- $y(\mathbf{x}) = 0$ is the decision boundary
- $y(\mathbf{x}) > 0$ means that the feature vector $\mathbf{x}$ is assigned to class $C_1$

- $y(\mathbf{x}) < 0$ means that the feature vector $\mathbf{x}$ is assigned to class $C_2$

A suitable discriminant function in this case is the log ratio of posterior probabilities (**log odds**):

$$\begin{aligned} y(\mathbf{x}) &= \ln \frac{\mathbb{P}(C_1|\mathbf{x})}{\mathbb{P}(C_2|\mathbf{x})} = \ln \frac{\mathbb{P}(\mathbf{x}|C_1)}{\mathbb{P}(\mathbf{x}|C_2)} + \ln \frac{\mathbb{P}(C_1)}{\mathbb{P}(C_2)} \\ &= \ln \mathbb{P}(C_1|\mathbf{x}) - \mathbb{P}(C_2|\mathbf{x}) + \ln \mathbb{P}(C_1) - \ln \mathbb{P}(C_2) \end{aligned}$$