Machine Learning 2/3

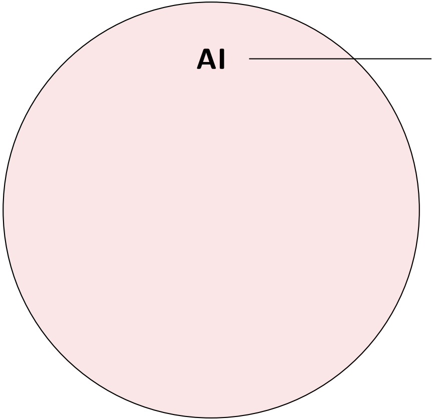
Lecture 08

Computer Vision for Geosciences

2021-04-30

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

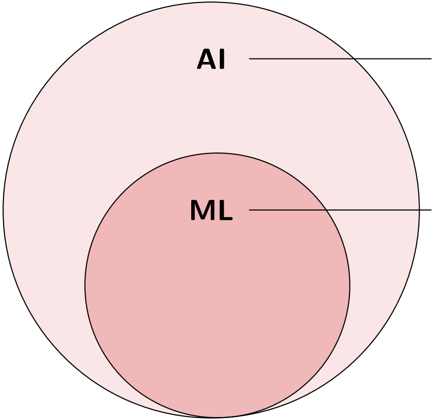
Pinpoint “hot” words



Artificial Intelligence

broad concept, whereby machine mimics human behaviour

Pinpoint "hot" words



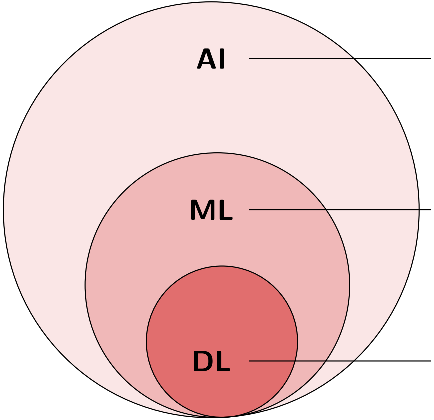
Artificial Intelligence

broad concept, whereby machine mimics human behaviour

Machine Learning *(a.k.a. Statistical Learning, Classical Learning)*

subset of AI which uses **statistical** methods
(features are designed by the user)

Pinpoint "hot" words



Artificial Intelligence

broad concept, whereby machine mimics human behaviour

Machine Learning *(a.k.a. Statistical Learning, Classical Learning)*

subset of AI which uses **statistical** methods

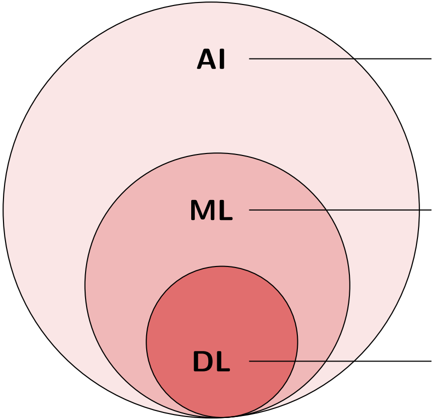
(features are designed by the user)

Deep Learning *(a.k.a. Modern Machine Learning)*

subset of ML, which uses **multi-layered neural networks**

(features are learned by the network)

Pinpoint "hot" words



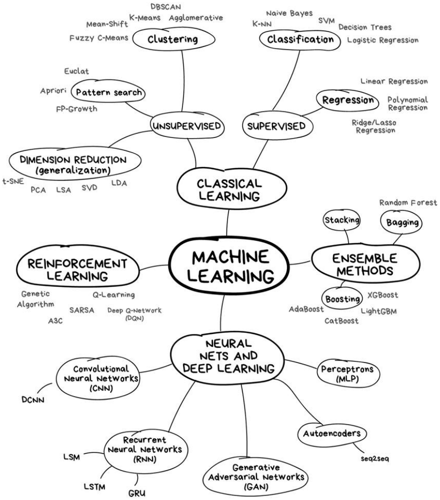
Artificial Intelligence
broad concept, whereby machine mimics human behaviour

Machine Learning *(a.k.a. Statistical Learning, Classical Learning)*
subset of AI which uses **statistical** methods
(features are designed by the user)

Deep Learning *(a.k.a. Modern Machine Learning)*
subset of ML, which uses **multi-layered neural networks**
(features are learned by the network)

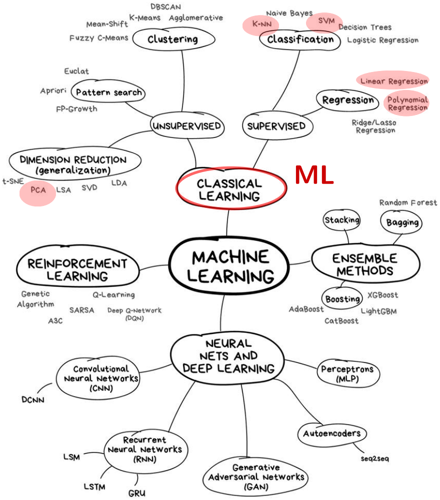
ML: lectures 07, 08 (today), 09
DL: lectures 10, 11, 12

Machine Learning is a huge (and growing) field!



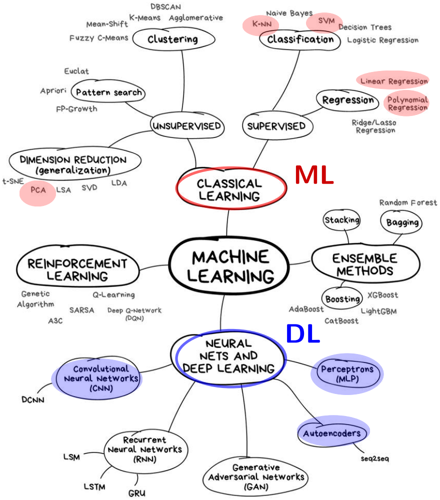
source

Machine Learning is a huge (and growing) field!



source

Machine Learning is a huge (and growing) field!



source

What we will introduce in the ML lectures:

Machine Learning

Supervised Learning

- Learning algorithm is presented inputs and desired outputs: **training data** $D = (in, out)$
- Goal: learn a general rule f that maps inputs to outputs $f(in) = out$

⇒ **Regression task**: *out* is a *continuous* number
e.g. linear regression, polynomial regression

⇒ **Classification task**: *out* is a *nominal* number (class label)
e.g. kNN, SVM, Logistic Regression

Unsupervised Learning

- No training data is given to the learning algorithm
- Goal: find structure data, discover hidden patterns, learn features

⇒ **Dimension reduction**, e.g. PCA
→ also used to craft features

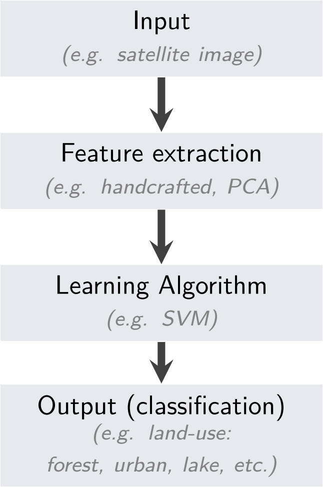
⇒ **Clustering task**, e.g. K-means

Classification task

- Goal:
  Learn the mapping between low level **features**, and **high level information** *(e.g. semantic classes)*

  NB: "features" is here used in a broad sense, not the "descriptors" introduced in lecture 06 (e.g. HOG, SIFT)

- Steps:
  1. features extraction *(e.g. handcrafted, PCA)*
  2. learning algorithm *(e.g. SVM)*



Toy example (courtesy of Andreas Ley & Ronny Hänsch)

- **Task**:
  $\Rightarrow$ classify fruit images into either bananas or apples

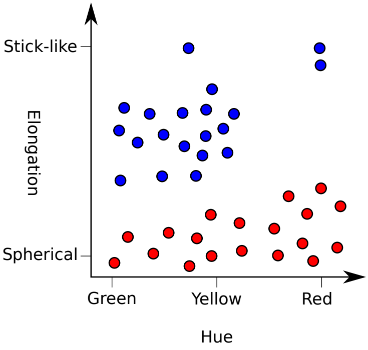
Toy example (courtesy of Andreas Ley & Ronny Hänsch)

- **Task**:
  $\Rightarrow$ classify fruit images into either bananas or apples

- **Features (hand-crafted)**:
  $\Rightarrow$ Hue (yellow to red) & Elongation (max/min extent)

Toy example (courtesy of Andreas Ley & Ronny Hänsch)

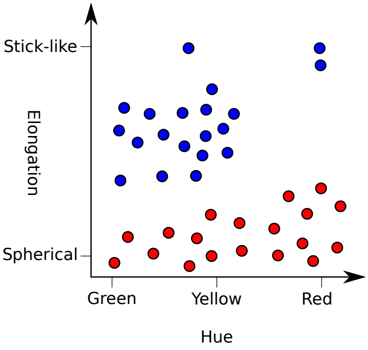
- **Task**:
  ⇒ classify fruit images into either bananas or apples
- **Features (hand-crafted)**:
  ⇒ Hue (yellow to red) & Elongation (max/min extent)
  ⇒ representation of input data in 2D feature space



Toy example (courtesy of Andreas Ley & Ronny Hänsch)

- **Task**:
  ⇒ classify fruit images into either bananas or apples

- **Features (hand-crafted)**:
  ⇒ Hue (yellow to red) & Elongation (max/min extent)
  ⇒ representation of input data in 2D feature space
  ⇒ can we "learn" which part of the feature space is bananas/apples?

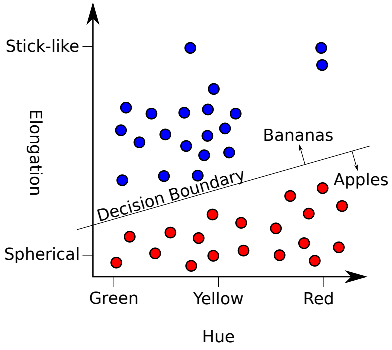


Toy example (courtesy of Andreas Ley & Ronny Hänsch)

- **Task**:
  ⇒ classify fruit images into either bananas or apples

- **Features (hand-crafted)**:
  ⇒ Hue (yellow to red) & Elongation (max/min extent)
  ⇒ representation of input data in 2D feature space
  ⇒ can we "learn" which part of the feature space is bananas/apples?

- **Learning algorithm**:
  ⇒ simple idea: split feature space into two half spaces

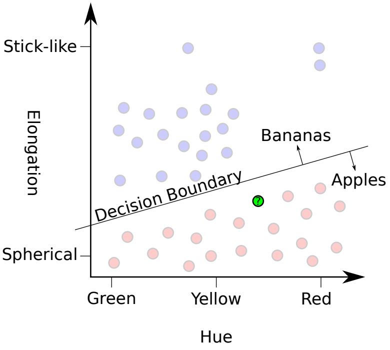


Toy example (courtesy of Andreas Ley & Ronny Hänsch)

- **Task**:
  ⇒ classify fruit images into either bananas or apples

- **Features (hand-crafted)**:
  ⇒ Hue (yellow to red) & Elongation (max/min extent)
  ⇒ representation of input data in 2D feature space
  ⇒ can we “learn” which part of the feature space is bananas/apples?

- **Learning algorithm**:
  ⇒ simple idea: split feature space into two half spaces
  ⇒ classify data based on linear decision boundary

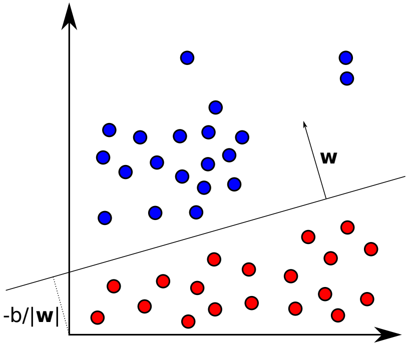


Toy example (courtesy of Andreas Ley & Ronny Hänsch)

- **Task**:
  $\Rightarrow$ classify fruit images into either bananas or apples

- **Features (hand-crafted)**:
  $\Rightarrow$ Hue (yellow to red) & Elongation (max/min extent)
  $\Rightarrow$ representation of input data in 2D feature space
  $\Rightarrow$ can we "learn" which part of the feature space is bananas/apples?

- **Learning algorithm**:
  $\Rightarrow$ simple idea: split feature space into two half spaces
  $\Rightarrow$ classify data based on linear decision boundary
  $\Rightarrow$ **perceptron**: $\boxed{y = \text{sign}(\mathbf{w}^T\mathbf{x} + b)}$
  - $y \in \{-1, 1\}$: predicted class $\rightarrow$ *banana or apple*
  - $\mathbf{x} \in \mathbb{R}^2$: feature vector $\rightarrow$ *[hue, elongation]*
  - $\mathbf{w} \in \mathbb{R}^2$: "weight vector" $\rightarrow$ *needs to be learned*
  - $b \in \mathbb{R}$: "bias" $\rightarrow$ *needs to be learned*
  - *sign*: sign function returning the sign of a real number



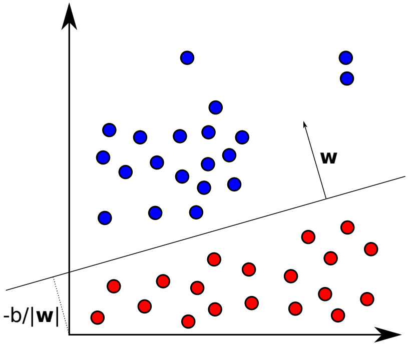
Toy example (courtesy of Andreas Ley & Ronny Hänsch)

- **Task**:
  $\Rightarrow$ classify fruit images into either bananas or apples

- **Features (hand-crafted)**:
  $\Rightarrow$ Hue (yellow to red) & Elongation (max/min extent)
  $\Rightarrow$ representation of input data in 2D feature space
  $\Rightarrow$ can we "learn" which part of the feature space is bananas/apples?

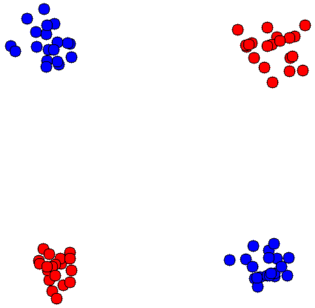
- **Learning algorithm**:
  $\Rightarrow$ simple idea: split feature space into two half spaces
  $\Rightarrow$ classify data based on linear decision boundary
  $\Rightarrow$ **perceptron**: $\boxed{y = \text{sign}(\mathbf{w}^T \mathbf{x} + b)}$
  - $y \in \{-1, 1\}$: predicted class $\rightarrow$ *banana or apple*
  - $\mathbf{x} \in \mathbb{R}^2$: feature vector $\rightarrow$ *[hue, elongation]*
  - $\mathbf{w} \in \mathbb{R}^2$: "weight vector" $\rightarrow$ *needs to be learned*
  - $b \in \mathbb{R}$: "bias" $\rightarrow$ *needs to be learned*
  - *sign*: sign function returning the sign of a real number

NB: **Support Vector Machine (SVM)**
can be used to find the best decision boundary
(i.e. which maximizes distance to data points)
$\rightarrow$ **next lecture!**



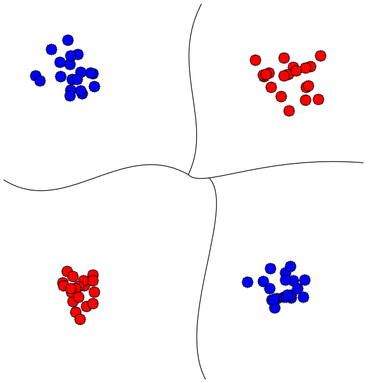
What if this linear separability does not exists? (courtesy of Andreas Ley & Ronny Hänsch)

- **Problem**:
  ⇒ feature space often not linearly separable



What if this linear separability does not exists? (courtesy of Andreas Ley & Ronny Hänsch)

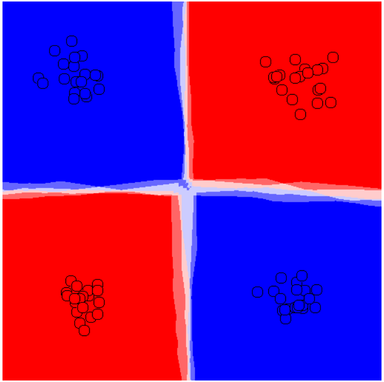
- **Problem**:
  ⇒ feature space often not linearly separable
  ⇒ needs non-linear decision boundary



What if this linear separability does not exists? (courtesy of Andreas Ley & Ronny Hänsch)

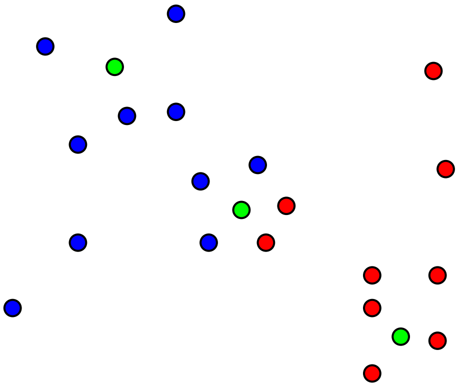
- **Problem**:
  $\Rightarrow$ feature space often not linearly separable
  $\Rightarrow$ needs non-linear decision boundary

- **Classification algorithm**:
  $\Rightarrow$ **k-Nearest-Neighbors (KNN)**



What if this linear separability does not exists? (courtesy of Andreas Ley & Ronny Hänsch)

- **Problem**:
  ⇒ feature space often not linearly separable
  ⇒ needs non-linear decision boundary
- **Classification algorithm**:
  ⇒ **k-Nearest-Neighbors (KNN)**
  1. take random data points in the training dataset

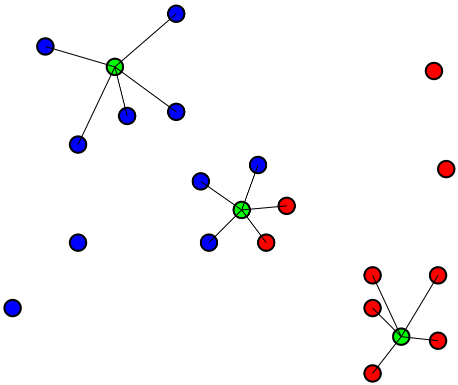


What if this linear separability does not exists? (courtesy of Andreas Ley & Ronny Hänsch)

- **Problem**:
  $\Rightarrow$ feature space often not linearly separable
  $\Rightarrow$ needs non-linear decision boundary

- **Classification algorithm**:
  $\Rightarrow$ **k-Nearest-Neighbors (KNN)**

  1. take random data points in the training dataset
  2. for a sample find the k (e.g. 5) closest data points in the training dataset

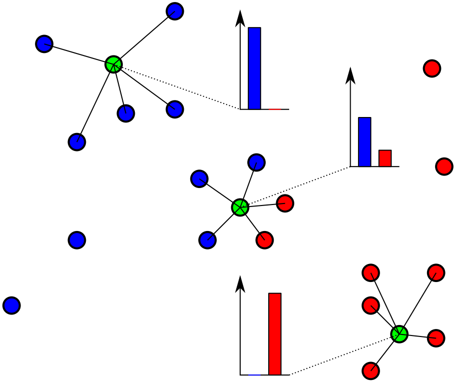


What if this linear separability does not exists? (courtesy of Andreas Ley & Ronny Hänsch)

- **Problem**:
  ⇒ feature space often not linearly separable
  ⇒ needs non-linear decision boundary

- **Classification algorithm**:
  ⇒ **k-Nearest-Neighbors (KNN)**

  1. take random data points in the training dataset
  2. for a sample find the k (e.g. 5) closest data points in the training dataset
  3. look at the neighbor labels, return/assign the mode

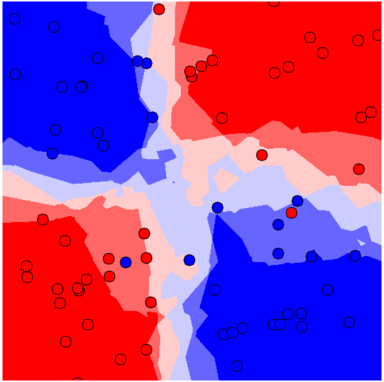


What if this linear separability does not exists? (courtesy of Andreas Ley & Ronny Hänsch)

- **Problem**:
  ⇒ feature space often not linearly separable
  ⇒ needs non-linear decision boundary

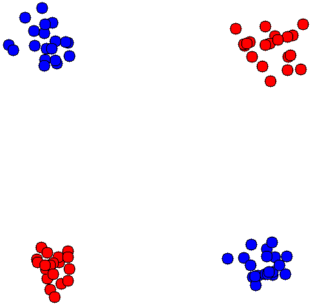
- **Classification algorithm**:
  ⇒ **k-Nearest-Neighbors (KNN)**

  1. take random data points in the training dataset
  2. for a sample find the k (e.g. 5) closest data points in the training dataset
  3. look at the neighbor labels, return/assign the mode
  4. decision boundary can be designed as probability meshgrids

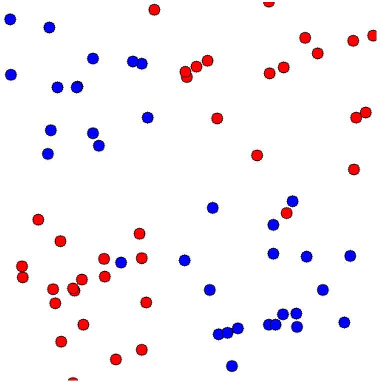


kNN examples

simple case

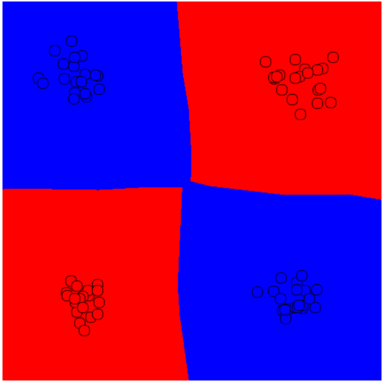


hard case

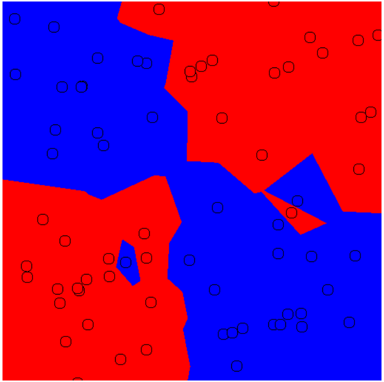


kNN examples

k = 1

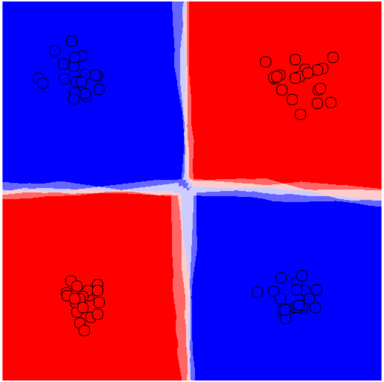


k = 1

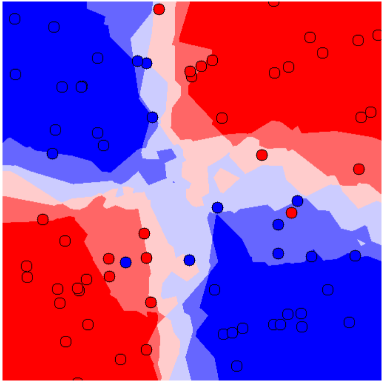


kNN examples

$k = 5$

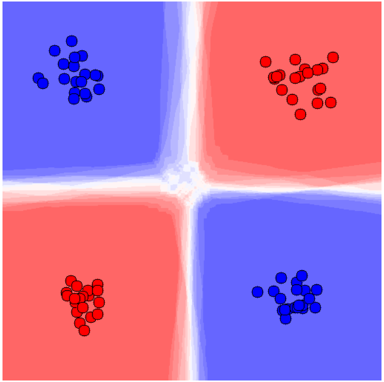


$k = 5$

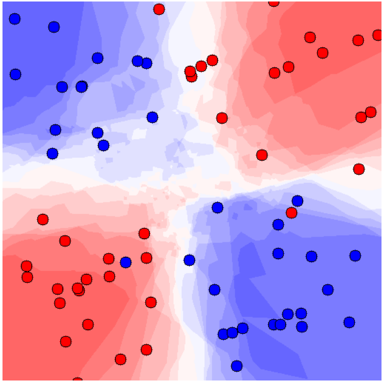


kNN examples

k = 25



k = 25



Feature extraction:

⇒ handcrafting features is nice, but can we do better?

Input

(e.g. satellite image)

↓

Feature extraction

*(e.g. handcrafted, **PCA**)*

↓

Learning Algorithm

(e.g. SVM)

↓

Output (classification)

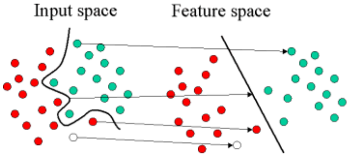
(e.g. land-use)

Feature extraction:

⇒ handcrafting features is nice, but can we do better?

⇒ find a space where samples from different classes are well separable

Input *(e.g. satellite image)*
↓
Feature extraction *(e.g. handcrafted, **PCA**)*
↓
Learning Algorithm *(e.g. SVM)*
↓
Output (classification) *(e.g. land-use)*



Feature extraction:

$\Rightarrow$ **Principal Component Analysis** (PCA) $\rightarrow$ *represent data in a space that best describes the data variation*

Feature extraction:

⇒ **Principal Component Analysis** (PCA) → *represent data in a space that best describes the data variation*

EX: intuitive representation of PCA (video):

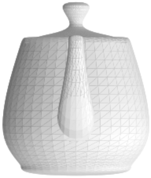
How to take a picture to capture the most information about the teapot?

Feature extraction:

⇒ **Principal Component Analysis** (PCA) → *represent data in a space that best describes the data variation*

EX: intuitive representation of PCA (video):

How to take a picture to capture the most information about the teapot?

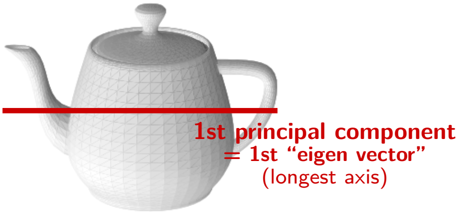


Feature extraction:

⇒ **Principal Component Analysis** (PCA) → *represent data in a space that best describes the data variation*

EX: intuitive representation of PCA (video):

How to take a picture that captures the most information about the teapot?

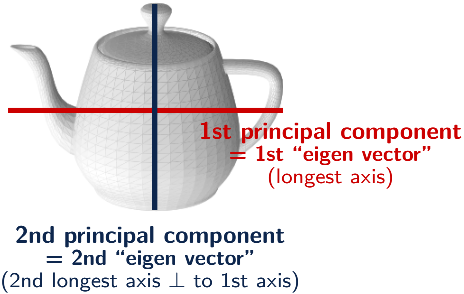


Feature extraction:

⇒ **Principal Component Analysis** (PCA) → *represent data in a space that best describes the data variation*

EX: intuitive representation of PCA (video):

How to take a picture that captures the most information about the teapot?



Feature extraction:

$\Rightarrow$ **Principal Component Analysis** (PCA) $\rightarrow$ *represent data in a space that best describes the data variation*

$\Rightarrow$ PCA can be used to reduce data dimensions $\rightarrow$ *will reduce computational load of the classifier*

PCA toy example (inspired by this post)

*We have several wine bottles in our cellar, described by different **features**: alcohol, color, etc. However many features will measure related properties, and so will be redundant.*

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04
...	...	...	...	...	...	...	...	...	...	...	...

PCA toy example (inspired by this post)

We have several wine bottles in our cellar, described by different ***features****: alcohol, color, etc. However many features will measure related properties, and so will be redundant.*

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04
...	...	...	...	...	...	...	...	...	...	...	...

$\Rightarrow$ PCA allows to summarize each wine with fewer characteristics
$\Rightarrow$ **reduce data dimensions**

PCA toy example (inspired by this post)

We have several wine bottles in our cellar, described by different ***features****: alcohol, color, etc. However many features will measure related properties, and so will be redundant.*

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04
...	...	...	...	...	...	...	...	...	...	...	...

$\Rightarrow$ PCA allows to summarize each wine with fewer characteristics
$\Rightarrow$ **reduce data dimensions**

$\Rightarrow$ PCA does *not* select some features and discards others,
instead it **defines new features** (using linear combinations of available features)
which will best represent wine variability

PCA toy example (inspired by this post)

We have several wine bottles in our cellar, described by different ***features****: alcohol, color, etc. However many features will measure related properties, and so will be redundant.*

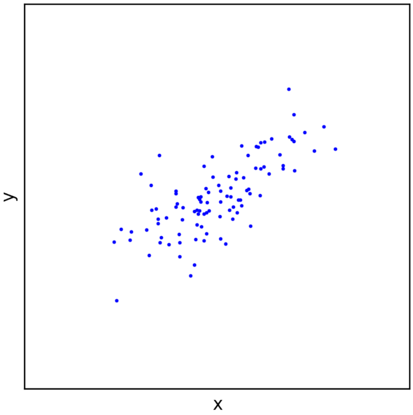
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04
...	...	...	...	...	...	...	...	...	...	...	...

⇒ PCA allows to summarize each wine with fewer characteristics
⇒ **reduce data dimensions**

⇒ PCA does *not* select some features and discards others,
instead it **defines new features** (using linear combinations of available features)
which will best represent wine variability

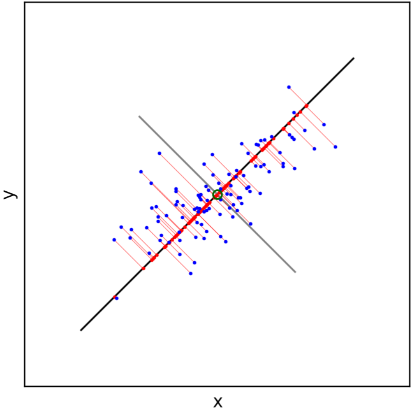
How?

Consider 2 correlated features x and y:



Consider 2 correlated features x and y:

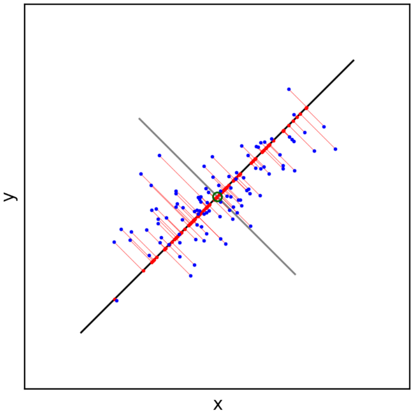
$\Rightarrow$ a **new "feature"** (red dots •) can be constructed by drawing a line through the cloud and projecting all points onto it



Consider 2 correlated features x and y:

$\Rightarrow$ a **new "feature"** (red dots •) can be constructed by drawing a line through the cloud and projecting all points onto it

$\Rightarrow$ **linear combination** $w_1 x + w_2 y$



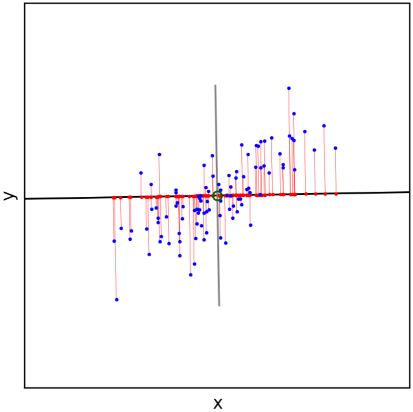
Consider 2 correlated features x and y:

⇒ a **new "feature"** (red dots •) can be constructed by drawing a line through the cloud and projecting all points onto it

⇒ **linear combination** $w_1 x + w_2 y$

⇒ PCA will find the "best" line according to 2 criteria:

- maximum **variance** of the red dots (i.e., spread along black line)
- minimum **distance** to black line (i.e., length of red lines)



NB: the above animation will only run with PDF readers having built-in JavaScript engine (ex: Adobe Reader, recent versions of Okular, etc.)

Consider 2 correlated features x and y:

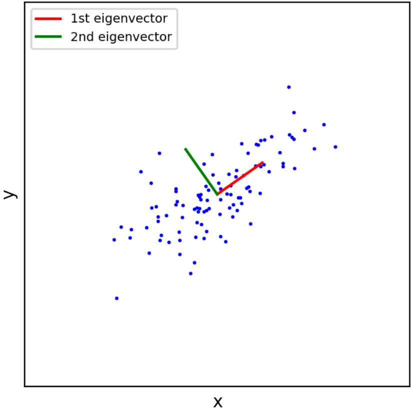
⇒ a **new "feature"** (red dots •) can be constructed by drawing a line through the cloud and projecting all points onto it

⇒ **linear combination** $w_1x + w_2y$

⇒ PCA will find the "best" line according to 2 criteria:

- maximum **variance** of the red dots (i.e., spread along black line)
- minimum **distance** to black line (i.e., length of red lines)

⇒ "best" line = **1st eigenvector** = **1st principal component**



Consider 2 correlated features x and y:

⇒ a **new "feature"** (red dots ●) can be constructed by drawing a line through the cloud and projecting all points onto it

⇒ **linear combination** $w_1 x + w_2 y$

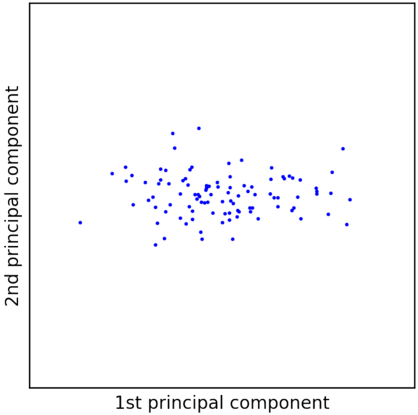
⇒ PCA will find the "best" line according to 2 criteria:

- maximum **variance** of the red dots (i.e., spread along black line)
- minimum **distance** to black line (i.e., length of red lines)

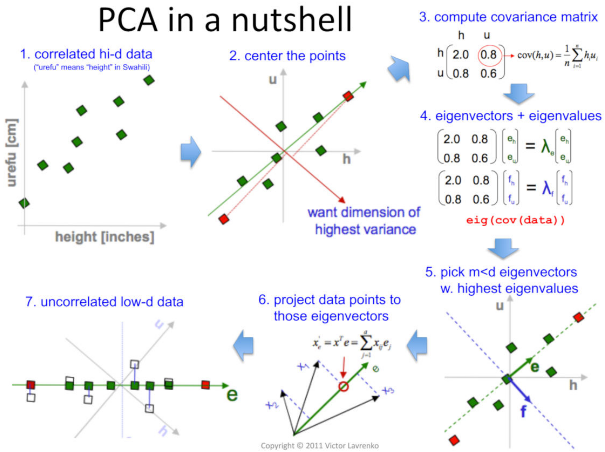
⇒ "best" line = **1st eigenvector** = **1st principal component**

⇒ we can project the data on the principal components, and thereby reduce dimensionality

NB: if only one eigenvector was kept, the transformed data would have only one dimension



⇒ **PCA implementation steps** (video link):



EXERCICE:

PCA analysis on satellite image crops

Math reminders

$$
\begin{aligned}
\textbf{variance } \sigma^2 &= \text{measure of the ``spread'' or ``extent'' of the data about some particular axis} \\
&= \text{average of the squared differences from the mean} \\
&= \text{square of standard deviation } (\sigma) \\
var_x &= \frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N} \\
var_y &= \frac{\sum_{i=1}^{N}(y_i - \bar{y})^2}{N}
\end{aligned}
$$

$$
\begin{aligned}
\textbf{covariance} &= \text{measure the level to which two variables vary together.'' of the joint variability of two random variables} \\
cov_{x,y} &= \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{N-1} \\
\textbf{covariance matrix} &= \begin{bmatrix} var_x & cov_{x,y} \\ cov_{x,y} & var_y \end{bmatrix}
\end{aligned}
$$