# Statistical Machine Learning

https://cvml.ist.ac.at/courses/SML_W20

**Christoph Lampert**

IST AUSTRIA

*Institute of Science and Technology*

Fall Semester 2020/2021

Lecture 1

**Goals:**

- learn the principles of (mainly supervised) machine learning
- get an overview of some core techniques
- not:hands-on experience in solving real-world tasks using machine learning

**Target Audience:**

- anyone interest in machine learning as a research topic.
- not: anyone who wants to apply machine learning to their own problems

**Prerequisites:**

- mathematics: set notation, linear algebra, multi-dimensional calculus, probabilities
- programming skills: in a language that allows numeric computation, e.g. Python

**Goals:**

- learn the principles of (mainly supervised) machine learning
- get an overview of some core techniques
- not:hands-on experience in solving real-world tasks using machine learning

**Target Audience:**

- anyone interest in machine learning as a research topic.
- not: anyone who wants to apply machine learning to their own problems

**Prerequisites:**

- mathematics: set notation, linear algebra, multi-dimensional calculus, probabilities
- programming skills: in a language that allows numeric computation, e.g. Python

**Caveat:**

- rumor is, the course is quite some work for its 3 ECTS

**Evaluation criteria:**

- 50% homework, 50% final project

**Homework:**

- every Monday there is a new homework sheet
- exercises are mainly theoretical/analytical, but also some practicals
- return your answers via email to the TAs *before the next Monday lecture*
- homeworks must be handed in individually
- group discussion about homeworks is permitted and encouraged
- cut-and-paste solutions are not permitted and will be sanctioned (1st offence: no points for sheet, 2nd offence: fail course)

**Final Team Project:**

- starting two weeks before the end of the course (approx. Nov 1)
- teams should have two participants
- you can use any method and software you want
- teams present their approaches during the "exam week" (Nov 13, Nov 15)

# Overview (tentative)

| Date | | no. | Topic |
|---|---|---|---|
| Oct 05 | Mon | 1 | A Hands-On Introduction |
| Oct 07 | Wed | 2 | Bayesian Decision Theory, Generative Probabilistic Models |
| Oct 12 | Mon | 3 | Discriminative Probabilistic Models |
| Oct 14 | Wed | 4 | Maximum Margin Classifiers, Generalized Linear Models |
| Oct 19 | Mon | 5 | Estimators; Overfitting/Underfitting, Regularization, Model Selection |
| Oct 21 | Wed | 6 | Bias/Fairness, Domain Adaptation |
| Oct 26 | Mon | - | no lecture (public holiday) |
| Oct 28 | Wed | 7 | Learning Theory I |
| Nov 02 | Mon | 8 | Learning Theory II |
| Nov 04 | Wed | 9 | Deep Learning I |
| Nov 09 | Mon | 10 | Deep Learning II |
| Nov 11 | Wed | 11 | Unsupervised Learning |
| Nov 16 | Mon | 12 | project presentations |
| Nov 18 | Wed | 13 | buffer |

**Definition (Mitchell, 1997)**

A computer program is said to *learn* from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$.

**Definition (Mitchell, 1997)**

A computer program is said to *learn* from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$.

**Example: Backgammon**

- T)ask: Play backgammon.
- E)xperience: Games playes against itself
- P)erformance Measure: Games won against human players.

**Definition (Mitchell, 1997)**

A computer program is said to *learn* from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$.

**Example: Spam classification**

- T)ask: determine if emails are Spam or non-Spam.
- E)xperience: Incoming emails with human classification
- P)erformance Measure: percentage of correct decisions

**Definition (Mitchell, 1997)**

A computer program is said to *learn* from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$.

**Example: Stock market predictions**

- T)ask: predict the price of some shares
- E)xperience: past prices
- P)erformance Measure: money you win or lose

## Notation

**Task:**

- $\mathcal{X}$: input set, set of all possible inputs
- $\mathcal{Y}$: output set, set of all possible outputs
- $f : \mathcal{X} \to \mathcal{Y}$: prediction function,
  - e.g. $\mathcal{X} = \{\texttt{all possible emails}\}, \mathcal{Y} = \{\texttt{spam}, \texttt{ham}\}$
    $f$ spam filter: for new email $x \in \mathcal{X}$: $f(x) = \texttt{spam}$ or $f(x) = \texttt{ham}$.

**Performance:**

- $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$: loss function
  - e.g. $\ell(y, y')$ is *cost* of predicting $y'$ if $y$ is correct.
  - $\ell(y, y')$ can be asymmetric: $\texttt{spam} \to \texttt{ham}$ is annoying, but no big deal.
  - $\texttt{ham} \to \texttt{spam}$ can cause serious problems.

**Experience:** task-dependent, many different scenarios

- **Supervised learning**: a labeled **training set** examples from $\mathcal{X}$ with outputs provided by an expert, $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y}$
  - A person goes through his/her $n$ emails and marks each one whether it is spam or not.

Many other variants on how to formalize *experience* exist:

- **Unsupervised Learning:** $\mathcal{D} = \{x^1, \dots, x^n\}$, only observing, no input from an expert/teacher
- **Semi-supervised Learning:** $\mathcal{D} = \{(x^1, y^1), \dots, (x^n, y^n)\} \cup \{x^{n+1}, \dots, x^{n+l}\}$: only a subset of examples has labels
  - spam: quite common, nobody wants to label *every email* in their inbox
- **Reinforcement Learning:** $\mathcal{D} = \{(x^1, r^1), \dots, (x^n, r^n)\}$ with $r^i \in \mathbb{R}$: actions and feedback how good the action was
  - backgammon: nobody tells you the best move, but eventually you observe a win or loss
- **Active Learning:** $\mathcal{D} = \{x^1, \dots, x^n\}$, but the algorithms may *ask* for labels
  - spam: email program can ask the user, if its not too often
- **Zero-Shot Learning:** $\mathcal{D} = \{(x^1, y^1), \dots, (x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y}$ but only a subset of all classes actually shows up in the training data.

In this course, we don't look at these (except a little bit at unsupervised learning).

## Definition

- A *supervised learning system* (or *learner*), $L$, is a (computable) function from the set of (finite) training sets to the set of prediction functions:

$$L : \mathbb{P}^{<\infty}(\mathcal{X} \times \mathcal{Y}) \to \mathcal{Y}^{\mathcal{X}}$$
$$\text{i.e.} \quad L : \mathcal{D} \mapsto f$$

  If presented with a training set $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$, it provides a decision rule/function $f : \mathcal{X} \to \mathcal{Y}$.

## Definition

Let $L$ be a learning system.

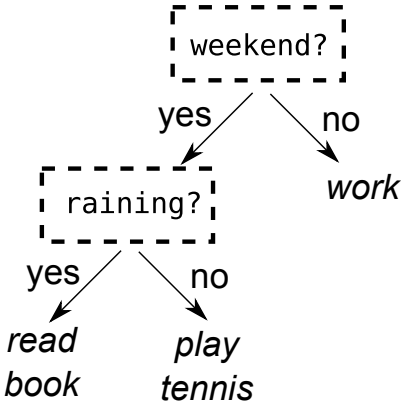
- The process of *computing* $f = L(\mathcal{D})$ is called *training (phase)*.
- Applying $f$ to new data is called *prediction*, or *testing* (phase).

We will look at examples of classical learning algorithms, to get a feeling what *problems* a learning system faces.

- Decision Trees
- Nearest Neighbor Classifiers
- Perceptron
- Boosting
- Artificial Neural Networks

**Caveat:** for each of these are there more advanced, often better, variants. Here, we look at the only as prototypes, not as guideline what to actually use in real life.

**Task:** decide what to do today




Classifier has a **tree structure**:

- each *interior node* makes a decision: it picks an attribute within $x$, branches for each possible value
- each *leaf* assigns an (output) label
- to classify a new example, we
  - put it into the root node,
  - follow the decisions until we reach a leaf.
  - use the leaf value as the prediction

Decisions trees ('expert systems') are popular especially for non-experts:

- **efficient**, **easy to use**, and **interpretable.**

## How to automatically build a decision tree

Given: training set $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\}$.

Convention:

- each node contains a subset of examples,
- its label is the majority label of the examples in this node (any of the majority labels, if there's a tie)

### Decision Tree – Training

initialize: put all examples in root node
mark root as *active*
**repeat**
  pick active node with largest number of misclassified examples
  mark the node as *inactive*
  for each attributes, check error rate of splitting along this attribute
  keep the split with smallest error, if any, and mark children as *active*
**until** no more active nodes.

## Decision Tree – Classification

**input** decision tree, example $x$
  assign $x$ to root node
  **while** $x$ not in leaf node **do**
    move $x$ to child according to the test in node
  **end while**
**output** label of the leaf that $x$ is in

# Decision Trees Example - Training

- We have a personalized dating agency, our only customer is *Zoe*.
- Task: For new customers registering, predict if *Zoe* should date them.
- Performance Measure: If *Zoe* is happy with the decision.
- Experience: We show *Zoe* a catalog of previous custemers and she tells us whether she would have like to date them or not.
- Let $x \in \mathcal{X}$ be a collection of values or properties, $x = (x_1, \ldots, x_d)$.

| property | possible values |
|---|---|
| `eye color` | blue/brown/green |
| `handsome` | yes/no |
| `height` | short/tall |
| `sex` | male (M)/female (F) |
| `soccer fan` | yes/no |

# Decision Trees Example - Training phase

**Preparation:** you give Zoe a set of profiles to see whom she would like to date (none of these people really have to exists...)

Here's her answers, which we'll use as **training data:**

| | $\mathcal{X}$ | | | | | $\mathcal{Y}$ |
|---|---|---|---|---|---|---|
| person | eyes | handsome | height | sex | soccer | date? |
| Apu | blue | yes | tall | M | no | yes |
| Bernice | brown | yes | short | F | no | no |
| Carl | blue | no | tall | M | no | yes |
| Doris | green | yes | short | F | no | no |
| Edna | brown | no | short | F | yes | no |
| Prof. Frink | brown | yes | tall | M | yes | no |
| Gil | blue | no | tall | M | yes | no |
| Homer | green | yes | short | M | no | yes |
| Itchy | brown | no | short | M | yes | yes |

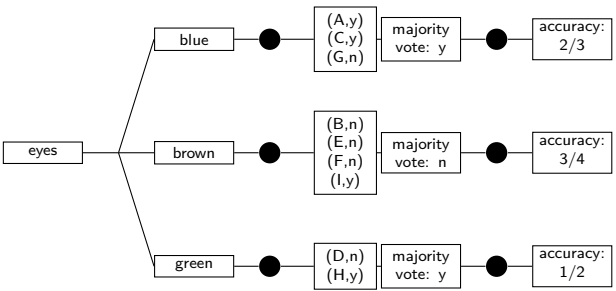
(this table is also on the exercise sheet)

# Decision Trees Example - Training phase

Step 1: put all all training examples into the root node

$$root = \{\ (A,y),(B,n),(C,y),(D,n),(E,n),(F,n),(G,n),(H,y),(I,y)\ \}$$

For each feature, check the classification accuracy of this single feature:



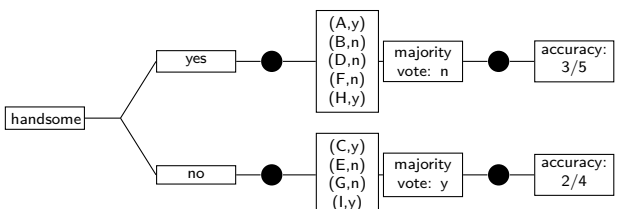

Total accuracy eyes: 6/9

# Decision Trees Example - Training phase

Step 1: put all all training examples into the root node

$$root = \{\ (A,y),(B,n),(C,y),(D,n),(E,n),(F,n),(G,n),(H,y),(I,y)\ \}$$

For each feature, check the classification accuracy of this single feature:




Total accuracy `handsome`: 5/9

# Decision Trees Example - Training phase

Step 1: put all all training examples into the root node

$$root = \{\ (A,y),(B,n),(C,y),(D,n),(E,n),(F,n),(G,n),(H,y),(I,y)\ \}$$

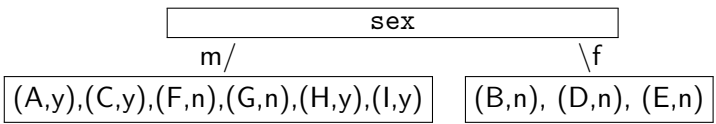
For each feature, check the classification accuracy of this single feature:

| feature | accuracies | | total |
|---|---|---|---|
| `eyes` | blue: (2/3), brown: (3/4), green: (1/2) | → | total: (6/9) |
| `handsome` | yes: (3/5), no: (2/4) | → | total: (5/9) |
| `height` | tall: (2/4), short: (3/5) | → | total: (5/9) |
| `sex` | male: (4/6), female: (3/3) | → | **total: (7/9)** |
| `soccer` | yes: (3/4), no: (3/6) | → | total: (6/9) |

Best feature: `sex`.

# Decision Trees Example - Training phase

Step 1 result: first split ist along sex feature




Right node: no mistakes, no more splits
Left node: run checks again for remaining data

Step 2:

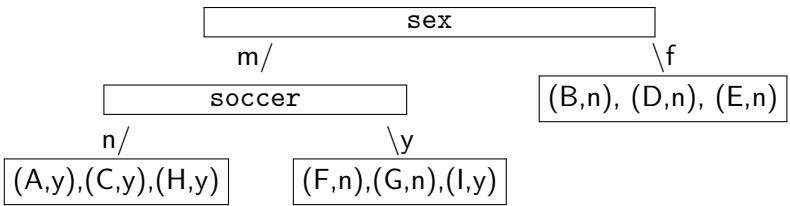
| person | eyes | handsome | height | sex | soccer | date? |
|---|---|---|---|---|---|---|
| Apu | blue | yes | tall | male | no | yes |
| Carl | blue | no | tall | male | no | yes |
| Frink | brown | yes | tall | male | yes | no |
| Gil | blue | no | tall | male | yes | no |
| Homer | green | yes | short | male | no | yes |
| Itchy | brown | no | short | male | yes | yes |

| feature | accuracies | | total |
|---|---|---|---|
| eyes | blue: (2/3), brown: (1/2), green: (1/1) | → | total: (4/6) |
| handsome | yes: (2/3), no: (2/3) | → | total: (4/6) |
| height | tall: (2/4), short: (2/2) | → | total: (4/6) |
| sex | male: (4/6) | → | total: (4/6) |
| soccer | yes: (2/3), no: (3/3) | → | **total: (5/6)** |

Best feature: soccer.

# Decision Trees Example - Training phase

Step 2 result: second split ist along `soccer` feature




Left node: no mistakes, no more splits
Right node: run checks again for remaining data

Step 3:

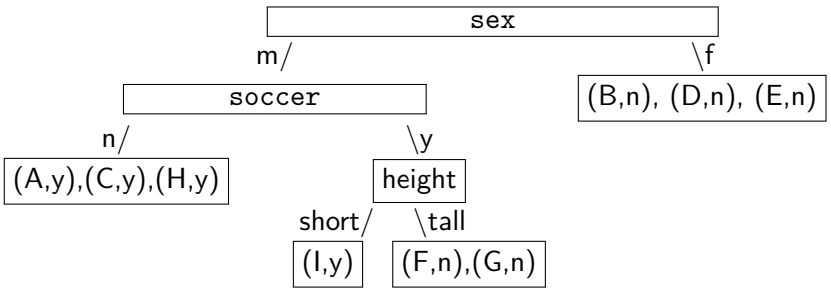
| person | eyes | handsome | height | sex | soccer | date? |
|---|---|---|---|---|---|---|
| Frink | brown | yes | tall | male | yes | no |
| Gil | blue | no | tall | male | yes | no |
| Itchy | brown | no | short | male | yes | yes |

| feature | accuracies | → | total |
|---|---|---|---|
| eyes | blue: (1/1), brown: (1/2), green: (0/0) | → | total: (2/3) |
| handsome | yes: (1/1), no: (1/2) | → | total: (2/3) |
| height | tall: (2/2), short: (1/1) | → | **total: (3/3)** |
| sex | male: (2/3) | → | total: (2/3) |
| soccer | yes: (2/3) | → | total: (2/3) |

Best feature: height.

# Decision Trees Example - Training phase

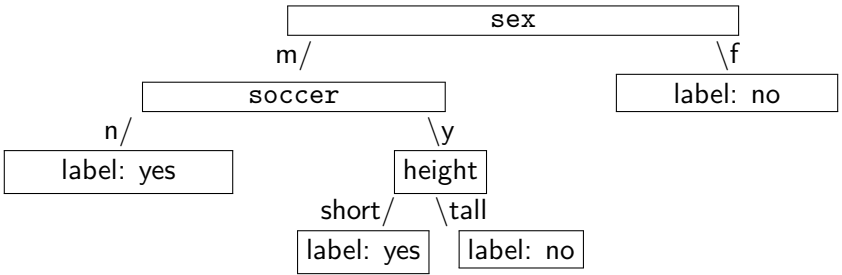
Step 3 result: third split ist along `height` feature




Left node: no mistakes, no more splits
Right node: no mistakes, no more splits

# Decision Trees Example - Training phase

Step 3 result: third split ist along `height` feature



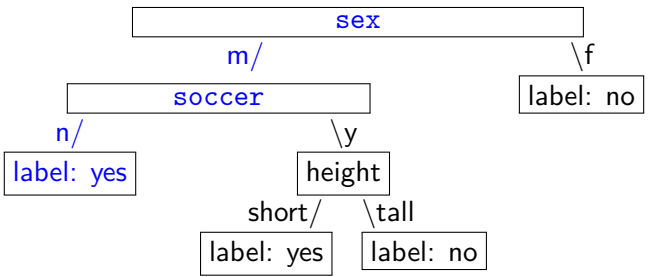

Left node: no mistakes, no more splits
Right node: no mistakes, no more splits

$\rightarrow$ Decision tree learning complete.

# Decision Trees Example - How good is this classifier?

Training example 1: correct

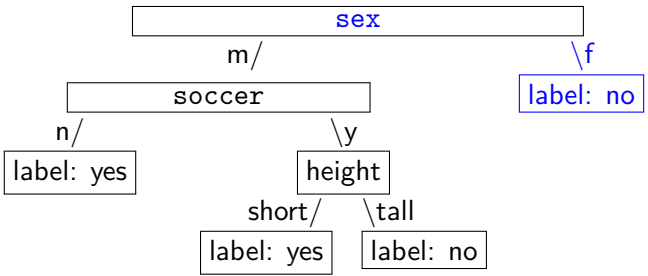
| person | eyes | handsome | height | sex | soccer | date? |
|---|---|---|---|---|---|---|
| Apu | blue | yes | tall | male | no | yes |



# Decision Trees Example - How good is this classifier?

Training example 2: correct

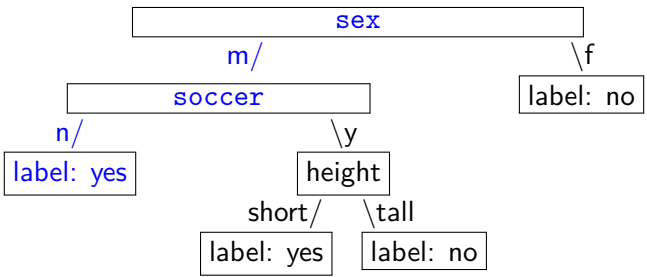
| person | eyes | handsome | height | sex | soccer | date? |
|---|---|---|---|---|---|---|
| Bernice | brown | yes | short | F | no | no |



# Decision Trees Example - How good is this classifier?

Training example 3: correct

| person | eyes | handsome | height | sex | soccer | date? |
|---|---|---|---|---|---|---|
| Carl | blue | no | tall | M | no | yes |



## Decision Trees Example - How good is this classifier?

- All training examples are classified correctly!

## Decision Trees Example - How good is this classifier?

- All training examples are classified correctly!

Not overly surprising... that's how we constructed the tree.

# Decision Trees Example - How good is this classifier?

What if we check on new data of the same kind?

| person | eyes | handsome | height | sex | soccer | date? |
|---|---|---|---|---|---|---|
| Jimbo | blue | no | tall | M | no | yes |
| Krusty | green | yes | short | M | yes | no |
| Lisa | blue | yes | tall | F | no | no |
| Moe | brown | no | short | M | no | no |
| Ned | brown | yes | short | M | no | yes |
| Quimby | blue | no | tall | M | no | yes |

# Decision Trees Example - How good is this classifier?

What if we check on new data of the same kind?

| person | eyes | handsome | height | sex | soccer | date? | tree |
|---|---|---|---|---|---|---|---|
| Jimbo | blue | no | tall | M | no | yes | **yes** |
| Krusty | green | yes | short | M | yes | no | **yes** |
| Lisa | blue | yes | tall | F | no | no | **no** |
| Moe | brown | no | short | M | no | no | **yes** |
| Ned | brown | yes | short | M | no | yes | **yes** |
| Quimby | blue | no | tall | M | no | yes | **yes** |

2 mistakes in 6, hm...

## Observation

Zoe won't care if our tree classifier worked perfectly on the training data. What really matters is how it works on future data: **ability to generalize**

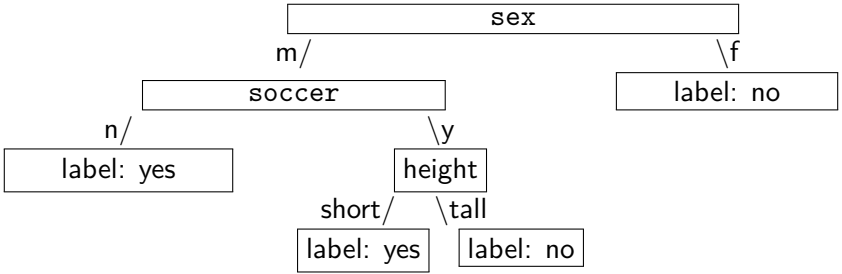
**Observation**

There is a relation between accuracy during training and accuracy at test time, but it isn't a simple one. **Perfect performance on the training set does not guarantee perfect performance on future data!**

Why did the tree make a mistake?
Maybe it took the training data too seriously?
Would Zoe really decide that male soccer fans are only datable, if they are *short*, but not if they are *tall*?
Let's see what happens in we simplify the tree?

# Decision Trees Example - How good is this classifier?

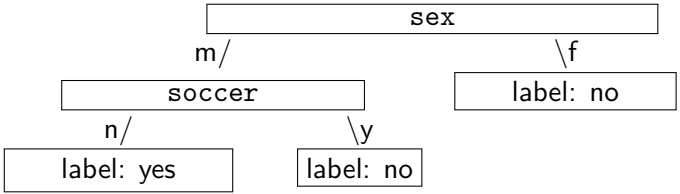
Original four-level tree: 2 mistakes in 6.




| person | eyes | handsome | height | sex | soccer | date? | tree |
|---|---|---|---|---|---|---|---|
| Jimbo | blue | no | tall | M | no | yes | **yes** |
| Krusty | green | yes | short | M | yes | no | **yes** |
| Lisa | blue | yes | tall | F | no | no | **no** |
| Moe | brown | no | short | M | no | no | **yes** |
| Ned | brown | yes | short | M | no | yes | **yes** |
| Quimby | blue | no | tall | M | no | yes | **yes** |

# Decision Trees Example - How good is this classifier?

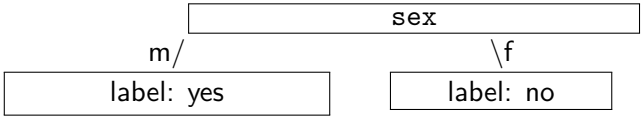
Tree with three levels: 1 mistake in 6.




| person | eyes | handsome | height | sex | soccer | date? | tree |
|---|---|---|---|---|---|---|---|
| Jimbo | blue | no | tall | M | no | yes | **yes** |
| Krusty | green | yes | short | M | yes | no | **no** |
| Lisa | blue | yes | tall | F | no | no | **no** |
| Moe | brown | no | short | M | no | no | **yes** |
| Ned | brown | yes | short | M | no | yes | **yes** |
| Quimby | blue | no | tall | M | no | yes | **yes** |

# Decision Trees Example - How good is this classifier?

Tree with two levels: 2 mistakes in 6.




| person | eyes | handsome | height | sex | soccer | date? | tree |
|---|---|---|---|---|---|---|---|
| Jimbo | blue | no | tall | M | no | yes | **yes** |
| Krusty | green | yes | short | M | yes | no | **yes** |
| Lisa | blue | yes | tall | F | no | no | **no** |
| Moe | brown | no | short | M | no | no | **yes** |
| Ned | brown | yes | short | M | no | yes | **yes** |
| Quimby | blue | no | tall | M | no | yes | **yes** |

# Decision Trees Example - How good is this classifier?

Tree with one level: 3 mistakes in 6.

label: no

| person | eyes | handsome | height | sex | soccer | date? | tree |
|---|---|---|---|---|---|---|---|
| Jimbo | blue | no | tall | M | no | yes | **no** |
| Krusty | green | yes | short | M | yes | no | **no** |
| Lisa | blue | yes | tall | F | no | no | **no** |
| Moe | brown | no | short | M | no | no | **no** |
| Ned | brown | yes | short | M | no | yes | **no** |
| Quimby | blue | no | tall | M | no | yes | **no** |

## Decision Trees Example - How good is this classifier?

Error analysis:

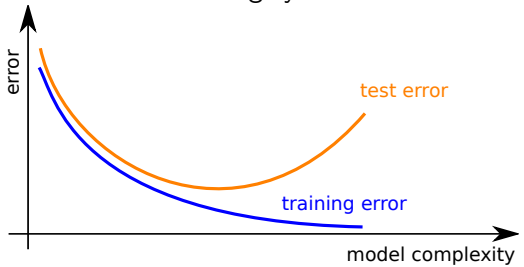
| size | training error | test error |
|---|---|---|
| height 1 | 4/9 | 3/6 |
| height 2 | 2/9 | 2/6 |
| height 3 | 1/9 | 1/6 |
| height 4 (full) | 0/9 | 2/6 |

# Decision Trees Example - How good is this classifier?

Error analysis:

| size | training error | test error |
|---|---|---|
| height 1 | 4/9 | 3/6 |
| height 2 | 2/9 | 2/6 |
| height 3 | 1/9 | 1/6 |
| height 4 (full) | 0/9 | 2/6 |

Typical behaviour of classical machine learning systems:



## Decision Tree Example - Lessons learned

Classifiers can have different **complexity**:

- **Complexity** has impact on both: training error and testing error.
- Training error: usually decreases with increasing complexity
- Test error: first decreases, then might go up again.

Test error behavior is so common that it has its own name:

- too simple models: high test error due to **underfitting**
  - the model cannot absorb the information from the training data
- too complex models: high test error due to **overfitting**
  - the model tries to reproduce idiosyncracies of the training data that future data will not have

Optimal classifier has a complexity somewhere inbetween, but:

- we cannot tell from either training error or test error alone if we underfit, overfit or neither
- seeing the complete *curve* will tell us!

## Decision Trees - Conclusion

- Categorial data can often be handled nicely by a tree.
- For continuous data, $\mathcal{X} = \mathbb{R}^d$, one typically uses splits by comparing any coordinate by a threshold: $[\![x_i \geq \theta]\!]$?
- Finding a split consists of checking all $i = 1, \ldots, d$ and all (reasonable) thresholds, e.g. all $x_i^1, \ldots, x_i^n$
- If $d$ is large, and all dimension are roughly of equal importance (e.g. time series), this is tedious, and the resulting tree might not be good.

Caveat: single decision trees are rarely used for real-world systems these days. But related techniques are, in particular *random forests*.

## Nearest Neighbor – Training

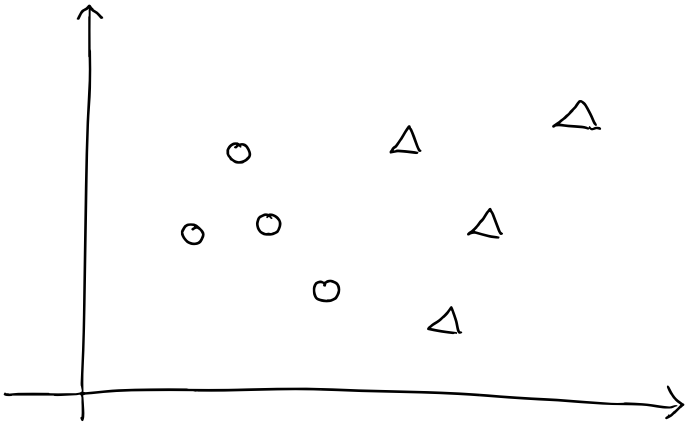
**input** dataset $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\} \subset \mathbb{R}^d \times \mathcal{Y}$
store all examples $(x^1, y^1), \ldots, (x^n, y^n)$.

## Nearest Neighbor – Prediction

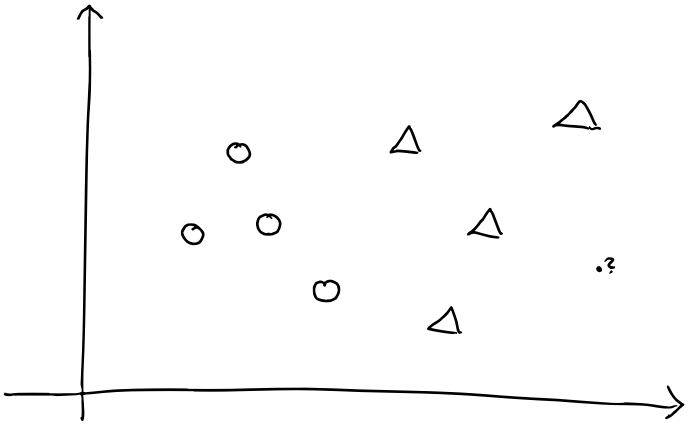
**input** new example $x \in \mathbb{R}^d$
for each training example $(x^i, y^i)$
compute $dist_i(x) = \|x - x^i\|$ (Euclidean distance)
**output** $y^j$ for $j = \mathbf{argmin}_j \, dist_i(x)$

(if $\mathbf{argmin}$ is not unique, pick between possible examples)

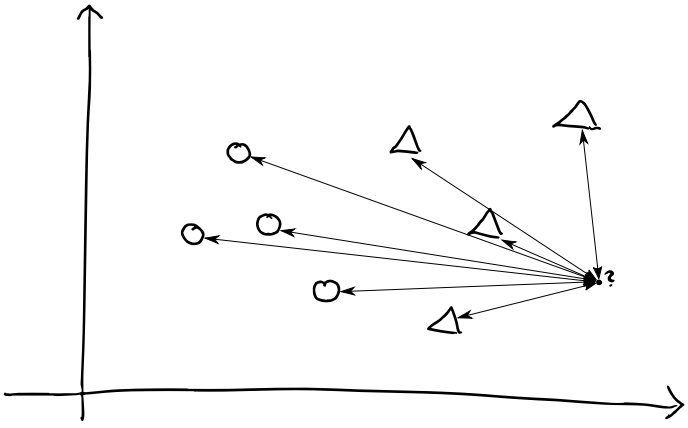
# Nearest Neighbor – Illustration



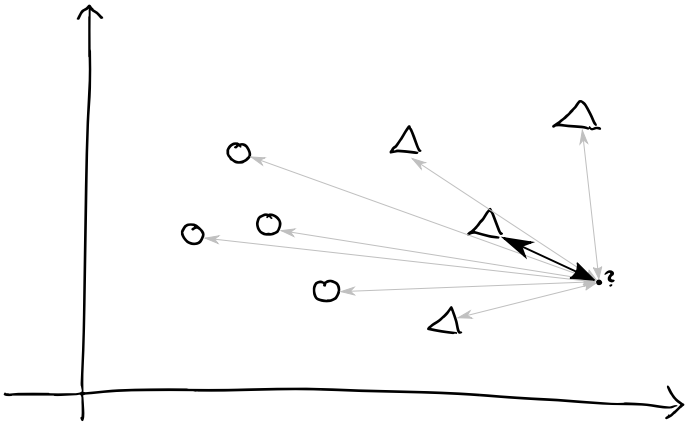
# Nearest Neighbor – Illustration



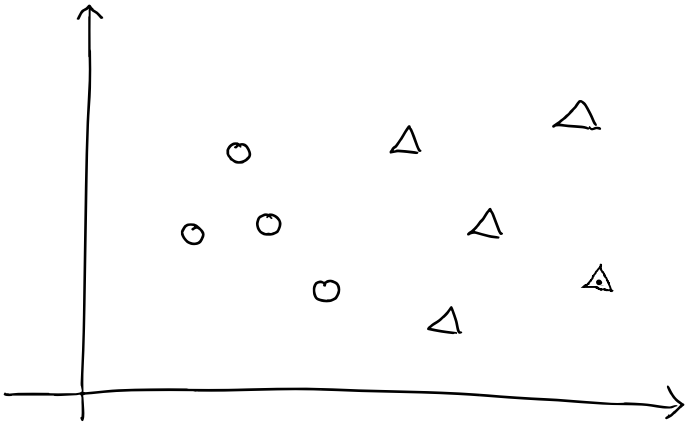
# Nearest Neighbor – Illustration



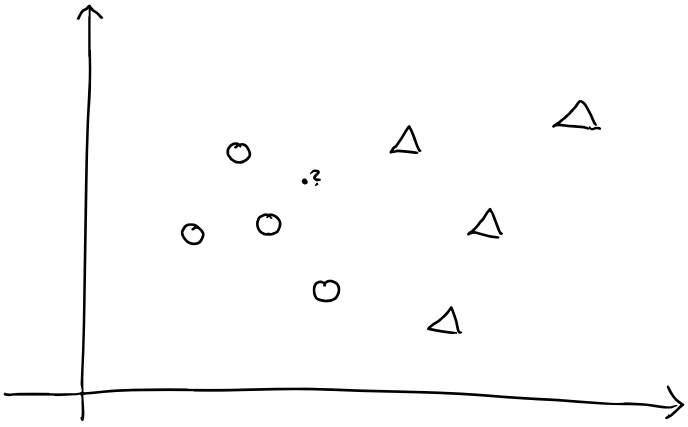
# Nearest Neighbor – Illustration



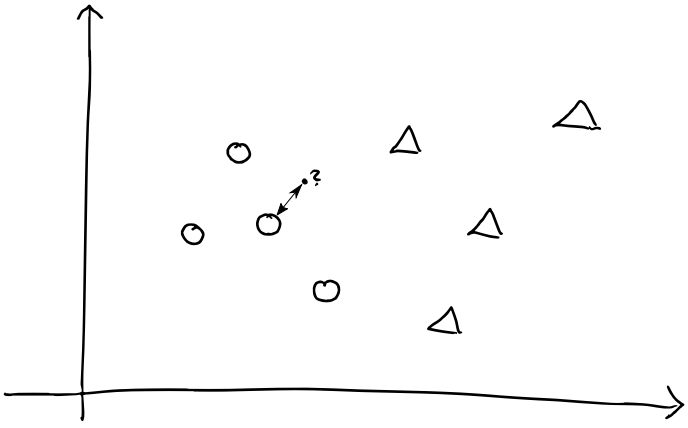
# Nearest Neighbor – Illustration



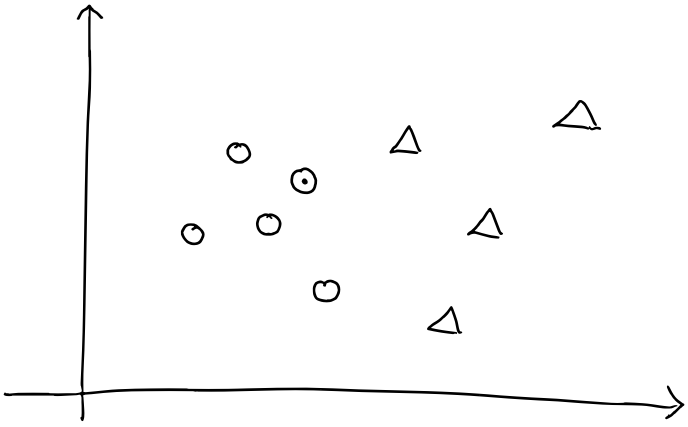
# Nearest Neighbor – Illustration



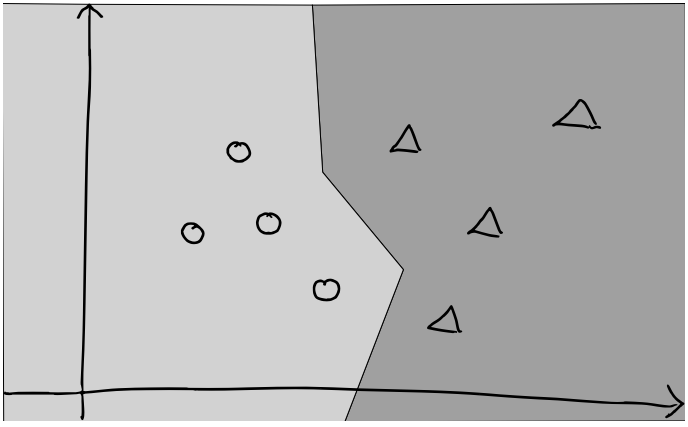
# Nearest Neighbor – Illustration

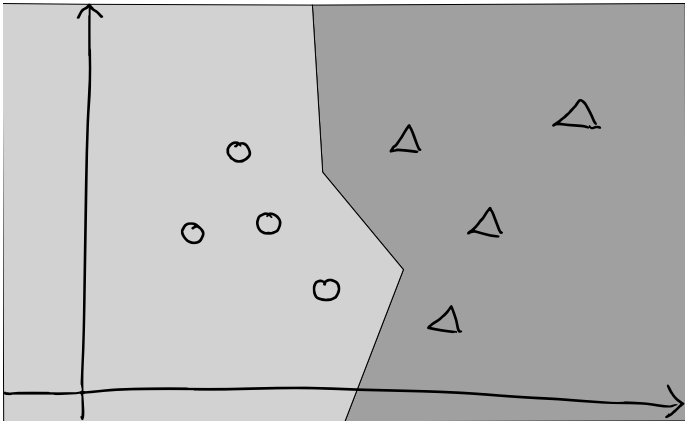


# Nearest Neighbor – Illustration



# Nearest Neighbor – Illustration





**Definition (Decision Boundary)**

Let $f : \mathcal{X} \to \mathcal{Y}$ be a classifier with discrete $\mathcal{Y} = \{1, \ldots, M\}$. The points where $f$ is *discontinuous* are called *decision boundary*.

## Nearest Neighbor

Nearest Neighbor prediction in the real world:

- very natural and intuitive
- we apply it without even considering it "learning" or "prediction"
- very popular in industry under a variety of names, e.g. 'case based reasoning', Example: helpdesk **"Similar problems have similar solutions".**

From a machine learning point of view:

- consider data as points in a (potentially high-dim.) vector space
- distance between two points tells us their *similarity*
- **Similar points tend to have the same label**.

We can also use NN for categorical labels: embed values into $\mathbb{R}^d$, e.g.

$$x_{Apu} = (\underbrace{1}_{\text{blue}}, \underbrace{0}_{\text{brown}}, \underbrace{0}_{\text{green}}, \underbrace{1}_{\text{handsome}}, \underbrace{0}_{\text{not handsome}}, \underbrace{1}_{\text{tall}}, \underbrace{0}_{\text{short}}, \underbrace{1}_{\text{male}}, \underbrace{0}_{\text{female}}, \underbrace{1}_{\text{soccer}}, \underbrace{0}_{\text{not soccer}})$$

More general than nearest neighbor:

## $k$-Nearest Neighbor – Training

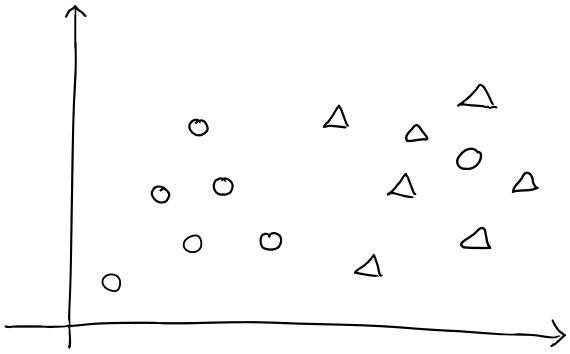
**input** dataset $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\} \subset \mathbb{R}^d \times \mathcal{Y}$
store all examples $(x^1, y^1), \ldots, (x^n, y^n)$.

## $k$-Nearest Neighbor – Classification

**input** new example $x$
for each training example $(x^i, y^i)$ compute $d_i(x) = \|x - x^i\|$ (Euclidean distance)
sort $d_i$ in increasing order
**output** majority vote among $y^i$s within the $k$ smallest $d^i$

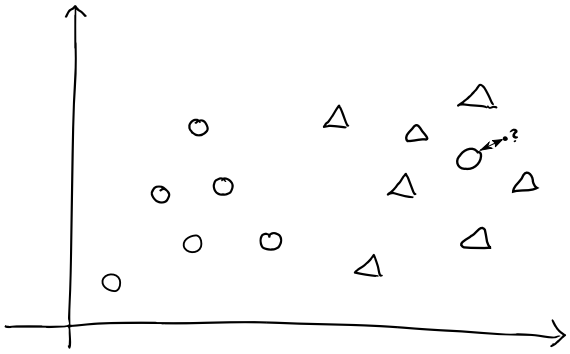
$k = 1$:

- original nearest neighbor rule
- any "outlier" influences the decision boundary



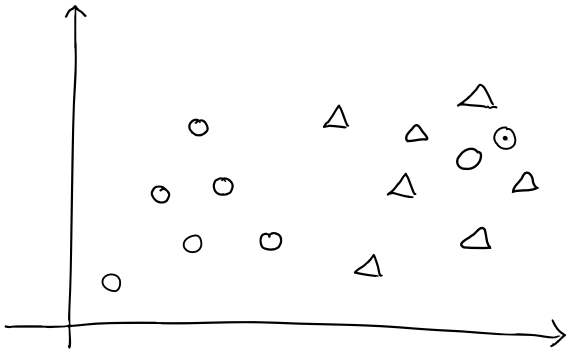
$k = 1$:

- original nearest neighbor rule
- any "outlier" influences the decision boundary



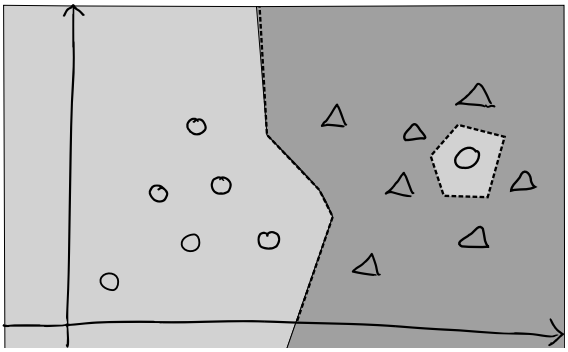
$k = 1$:

- original nearest neighbor rule
- any "outlier" influences the decision boundary



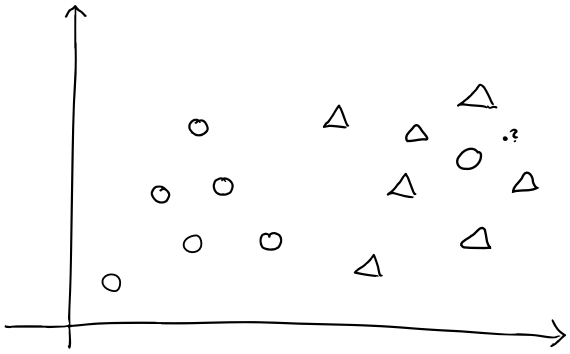
$k = 1$:

- original nearest neighbor rule
- any "outlier" influences the decision boundary



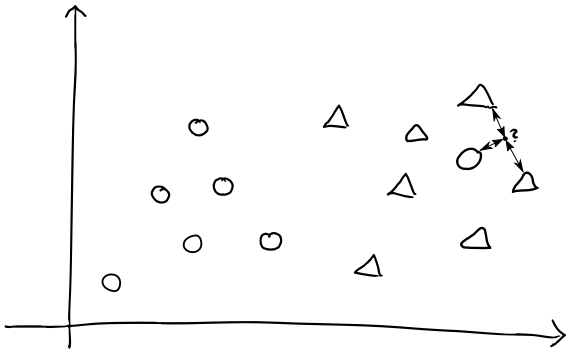
$k = 3$:

- isolated outliers are ignored
- decision boundary becomes smoother



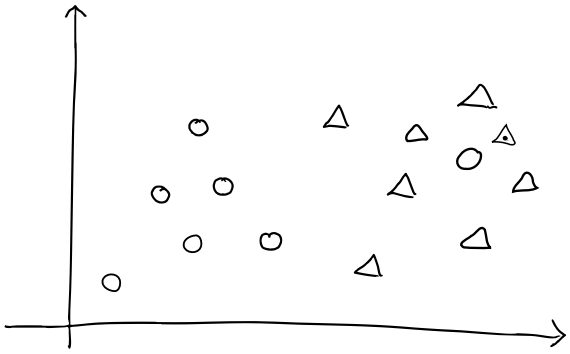
$k = 3$:

- isolated outliers are ignored
- decision boundary becomes smoother



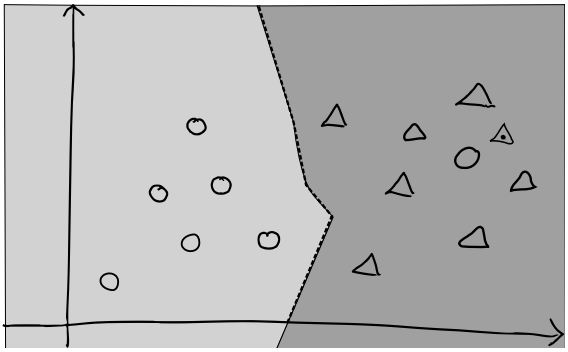
$k = 3$:

- isolated outliers are ignored
- decision boundary becomes smoother



$k = 3$:

- isolated outliers are ignored
- decision boundary becomes smoother



**Observation:** $k$ controls the complexity of the model:

- $k = 1$, decisions based on a single (most similar) example at a time, this might have an unreliable label (overfitting).
- $k = n$, all new points are classified with the label (underfitting).
- as before: there's a sweet spot inbetween.

## Perceptron – Rosenblatt, 1957

So far we've seen two classifiers:

- decision tree: picks a few important features to base decision on
- $k$-NN: all features contribute equally (to Euclidean distance)

Often, neither is optimal:

- we have many features, we want to make use of them.
- but some features are more useful or reliable than others.

**Idea:** learn how important each feature, $x_j$, is by a weight, $w_j$

**Perceptron algorithm:** inspired by (early) neuroscience:

- neurons form a weighted sum of their inputs $x = (x_1, \ldots, x_d)$
- they output a spike if the result exceeds a threshold, $\theta$

$$h(x) = \begin{cases} +1 & \text{if } \sum_j w_j x_j \geq \theta \\ -1 & \text{otherwise} \end{cases} \qquad = \text{sign } (\langle w, x \rangle - \theta).$$

## Perceptron – Training (for $\theta = 0$)

**input** training set $\mathcal{D} \subset \mathbb{R}^d \times \{-1, +1\}$
  initialize $w = (0, \ldots, 0) \in \mathbb{R}^d$.
  **repeat**
    **for all** $(x, y) \in \mathcal{D}$: **do**
      compute $a := \langle w, x \rangle$  ('activation')
      **if** $ya \leq 0$ **then**
        $w \leftarrow w + yx$
      **end if**
    **end for**
  **until** $w$ wasn't updated for a complete pass over $\mathcal{D}$

## Perceptron – Classification (for $\theta = 0$)

**input** new example $x$
**output** $f(x) = \text{sign}\langle w, x \rangle$  by convention, $\text{sign}(0) = -1$

## Perceptron – Example

$\mathcal{D}$: $(x^1, y^1) = (\begin{pmatrix}3\\1\end{pmatrix}, +1)$, $(x^2, y^2) = (\begin{pmatrix}1\\1\end{pmatrix}, +1)$, $(x^3, y^3) = (\begin{pmatrix}1\\4\end{pmatrix}, -1)$.

Round 1:

- $w = \begin{pmatrix}0\\0\end{pmatrix}$, $\quad i = 1$: $\langle w, x^1 \rangle = 0$, $\quad 1 \cdot 0 = 0 \leq 0 \quad \rightarrow \quad$ update

  $w_{new} = w_{old} + 1 \cdot \begin{pmatrix}3\\1\end{pmatrix} = \begin{pmatrix}3\\1\end{pmatrix}$

- $w = \begin{pmatrix}3\\1\end{pmatrix}$, $\quad i = 2$: $\langle w, x^2 \rangle = 4 \quad 1 \cdot 4 = 4 \not\leq 0 \quad \rightarrow$ no change

- $w = \begin{pmatrix}3\\1\end{pmatrix}$, $\quad i = 3$: $\langle w, x^3 \rangle = 7$, $\quad (-1) \cdot 7 = -7 \leq 0 \quad \rightarrow \quad$ update

  $w_{new} = w_{old} + (-1) \begin{pmatrix}1\\4\end{pmatrix} = \begin{pmatrix}2\\-3\end{pmatrix}$

## Perceptron – Example

$\mathcal{D}$: $(x^1, y^1) = (\begin{pmatrix}3\\1\end{pmatrix}, +1)$, $(x^2, y^2) = (\begin{pmatrix}1\\1\end{pmatrix}, +1)$, $(x^3, y^3) = (\begin{pmatrix}1\\4\end{pmatrix}, -1)$.

Round 2:

- $w = \begin{pmatrix}2\\-3\end{pmatrix}$, $\quad i = 1$: $\langle w, x^1\rangle = 3$, $\quad 1 \cdot 3 = 3 \not\leq 0 \quad \rightarrow$ no change
- $w = \begin{pmatrix}2\\-3\end{pmatrix}$, $\quad i = 2$: $\langle w, x^2\rangle = -1$, $\quad 1 \cdot (-1) = -1 \leq 0$

  $\rightarrow \quad w_{new} = w_{old} + 1 \begin{pmatrix}1\\1\end{pmatrix} = \begin{pmatrix}3\\-2\end{pmatrix}$
- $w = \begin{pmatrix}3\\-2\end{pmatrix}$, $\quad i = 3$: $\langle w, x^3\rangle = -5$, $\quad (-1) \cdot (-5) = 5 \not\leq 0$

  $\rightarrow \quad$ no change

## Perceptron – Example

$\mathcal{D}$: $(x^1, y^1) = (\begin{pmatrix}3\\1\end{pmatrix}, +1)$, $(x^2, y^2) = (\begin{pmatrix}1\\1\end{pmatrix}, +1)$, $(x^3, y^3) = (\begin{pmatrix}1\\4\end{pmatrix}, -1)$.

Round 3:

- $w = \begin{pmatrix}3\\-2\end{pmatrix}$, $\quad i = 1$: $\langle w, x^1\rangle = 7$, $\quad 1 \cdot 7 = 7 \not\leq 0$
- $w = \begin{pmatrix}3\\-2\end{pmatrix}$, $\quad i = 2$: $\langle w, x^2\rangle = 1$, $\quad 1 \cdot 1 = 1 \not\leq 0$
- $w = \begin{pmatrix}3\\-2\end{pmatrix}$, $\quad i = 3$: $\langle w, x^3\rangle = -5$, $\quad (-5) \cdot (-1) = 5 \not\leq 0$

nothing changed for 1 complete round: converged

## Perceptron – Example

$\mathcal{D}$: $(x^1, y^1) = (\begin{pmatrix}3\\1\end{pmatrix}, +1)$, $(x^2, y^2) = (\begin{pmatrix}1\\1\end{pmatrix}, +1)$, $(x^3, y^3) = (\begin{pmatrix}1\\4\end{pmatrix}, -1)$.

Round 3:

- $w = \begin{pmatrix}3\\-2\end{pmatrix}$, $\quad i = 1$: $\langle w, x^1 \rangle = 7$, $\quad 1 \cdot 7 = 7 \not\leq 0$
- $w = \begin{pmatrix}3\\-2\end{pmatrix}$, $\quad i = 2$: $\langle w, x^2 \rangle = 1$, $\quad 1 \cdot 1 = 1 \not\leq 0$
- $w = \begin{pmatrix}3\\-2\end{pmatrix}$, $\quad i = 3$: $\langle w, x^3 \rangle = -5$, $\quad (-5) \cdot (-1) = 5 \not\leq 0$

nothing changed for 1 complete round: converged

Final classifier: $f(x) = \operatorname{sign}(3 \cdot x_1 - 2 \cdot x_2)$

Limitation: always has a *linear* decision boundary, might not converge

## Summary

**Learning algorithms come in all kind of forms and flavors:**

- tree structured, "expert systems"
- similarity-based, geometric
- linear thresholding function
- weighted combinations of simpler units ($\rightarrow$ next lecture)
- iterated/stacked combinations of simpler units ($\rightarrow$ next lecture)

# Summary

**Learning algorithms come in all kind of forms and flavors:**

- tree structured, "expert systems"
- similarity-based, geometric
- linear thresholding function
- weighted combinations of simpler units ($\rightarrow$ next lecture)
- iterated/stacked combinations of simpler units ($\rightarrow$ next lecture)

**Machine learning research**

- studies their properties
- provides tools for choosing between different methods
- allows constructing new ones (with better properties)