What is Machine Learning?

A rather technical answer by Tom Mitchell

A computer program is said to learn from experience E with respect to some task T and performance measure P, if its performance at task T, as measured by P, improves with experience E.

- In short, we learn a task (T) from input data (E), and we expect the result to improve (P) given more input data.

Machine learning is *data driven* $\Rightarrow$ we make use of available data to make a decision. In contrast to knowledge systems that makes use of domain specific knowledge to make a decision.

Is this a cat?

According to Mitchell (from previous slide)

- **Task** (T): Identify if a picture contains a cat (*Yes* or *No*)
- **Experience** (E): A large number of pictures labelled as cat or non-cat.
- **Perf. measure** (P): Proportion of pictures correctly classified as cats

To learn the task T, we need experience E, and a performance measure P.
We expect P to improve if E is increased.

Recent machine learning achievements

- AI Outperformed Radiologist In Diagnosing Lung Cancer
- AlphaGo beats Go human world champion

- DeepStack beats professional poker players
- AlphaZero beats champion chess program after teaching itself chess in four hours (using 5000 processors working in parallel)
- Deep Net beats human at recognizing traffic signs

- Chat GPT can hold actual conversations, write essays and code, and answer all kind of questions. You are not allowed to use it in this course!

Old achievements

Machine learning has been used in email spam filters since the 1990s.

Why the hype? Why now?

Machine learning has been around for a while

- Machine learning is considered a subtopic of artificial intelligence (AI)
- Many algorithms used today were first presented in the 1990s, ...
- ... and some as early as 1950s (k Nearest Neighbors)

Today (all of a sudden) we have ...

- An introductory course with $>$ 100 students
- Three new ML courses at advanced level
- Spent several million SEK on a High-Performance Computing Center
- Several ML related research projects together with industrial partners

Why the hype? Why now?

- We now have the computational power
- We now have (due to the Internet) access to very large datasets

$\Rightarrow$ We can finally do a lot of fancy stuff that was impossible earlier on

Machine Learning Paradigms

Labeled data $\mathbf{x}_i, y_i \Rightarrow$ we know that input $\mathbf{x}_i$ gives response y_i

- **Supervised Learning** Given *labeled* data consisting of features $\mathbf{x}_i$ and response y_i, find a model $y_i = \hat{f}(\mathbf{x}_i)$.
  - y continuous $\Rightarrow$ *Regression*
  - y discrete $\Rightarrow$ *Classification*
- **Unsupervised Learning** Find structure in a given *unlabeled* dataset $\mathbf{x}_i$. E.g. group customers into different categories based on their purchase patterns. **Lectures 8-10.**
- **Semi-supervised Learning** Improve unsupervised learning by making use of set of labeled data. Or vice versa, improve supervised learning by making use of unlabeled data. **Not part of this course.**
- **Reinforced Learning** No correct input-output pairs available. Instead we provide a method for the machine to quantify its performance in the form of a reward signal. Hot topic ... **but not part of this course**.
  Many recent achievements (previous slide) are based on reinforced learning.

Supervised Learning

- **Task:** Learn function $y = \hat{f}(\mathbf{x})$ from a set of training data $(\mathbf{x}_i, y_i)$
- **Goal:** Identify $\hat{f}(\mathbf{x})$ that can predict $y_\star$ for *unseen* instances $\mathbf{x}_\star$.
- **Expectation:** Precision will improve if more training examples are provided
- It is **regression** if y is continuous, and **classification** if y is discrete

Examples

- Predict the height of a girl based on the height of her parents $\Rightarrow$ learn a function $h_{girl} = f(h_{mom}, h_{dad}) \Rightarrow$ **regression** since height h_{girl} is a continuous variable.
- Predict whether a student should be accepted to a certain university based on his/her parents income and education $\Rightarrow$ **classification** since accepted (yes or no) is binary.

Regression Example: House Prices in Oregon

Problem
Given 200 samples of house area vs price, predict the price for a 3500 square feet house?

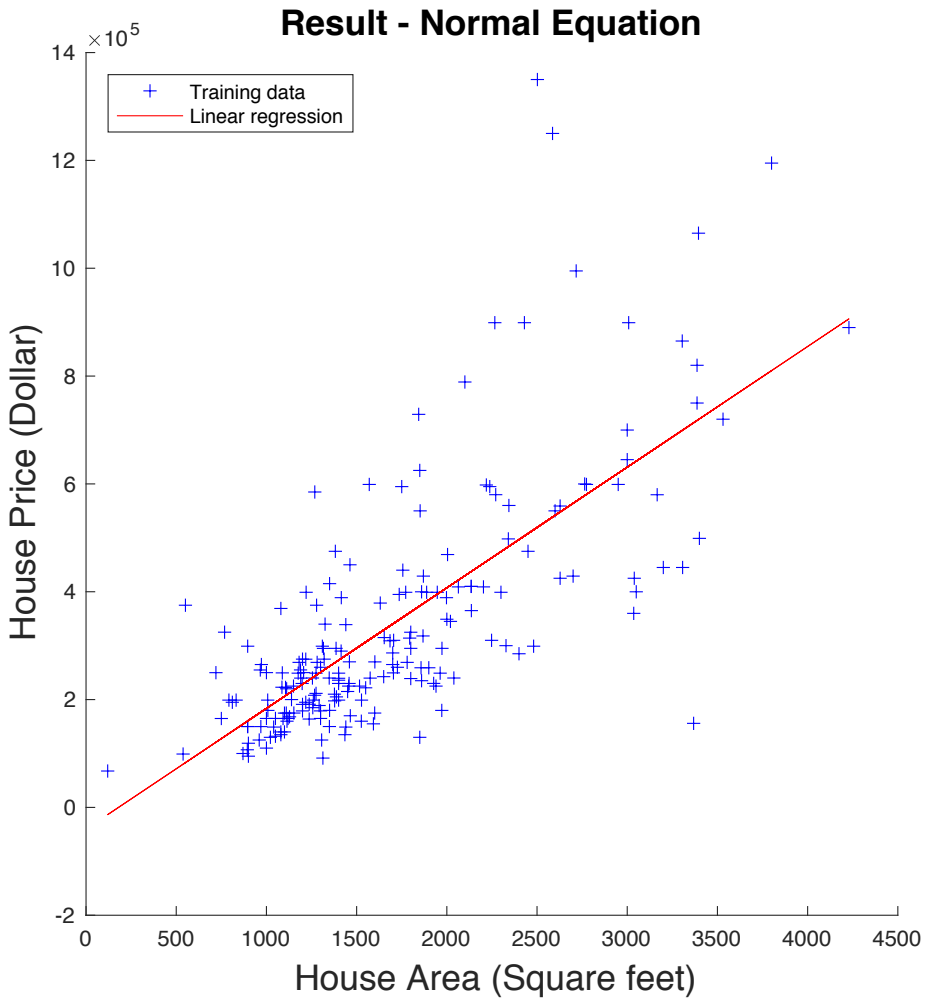
A regression problem since price is continuous

One possible solution
Linear fit on training set $\Rightarrow$

$$\text{price} = 40259 + 224 \times \text{area}$$

Result: Price for a 3500 sq-ft house is \$742946

Linear regression will be presented in Lecture 2



Classification Example: Student Admission

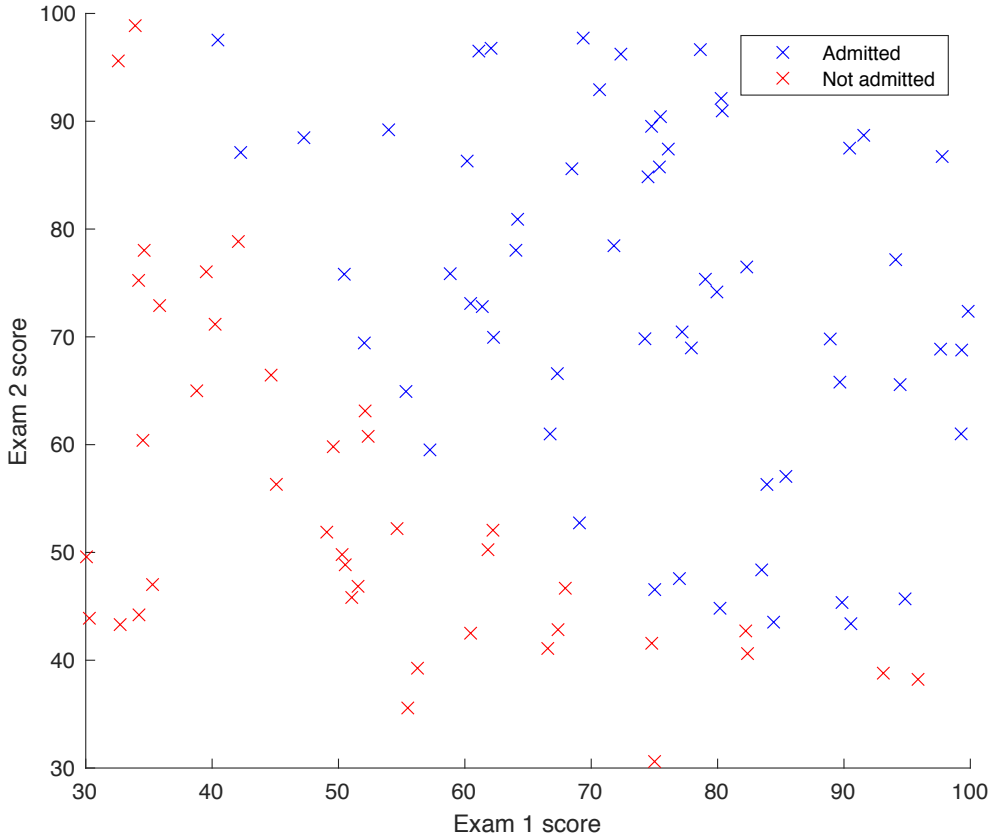
All students are required to take two exams as a part of the admission procedure to a university. We have 100 observations (x_1, x_2) with known admission results *yes* (1) or *no* (0)

Problem
Will a student with scores (45, 85) get admitted? (Yes or No)

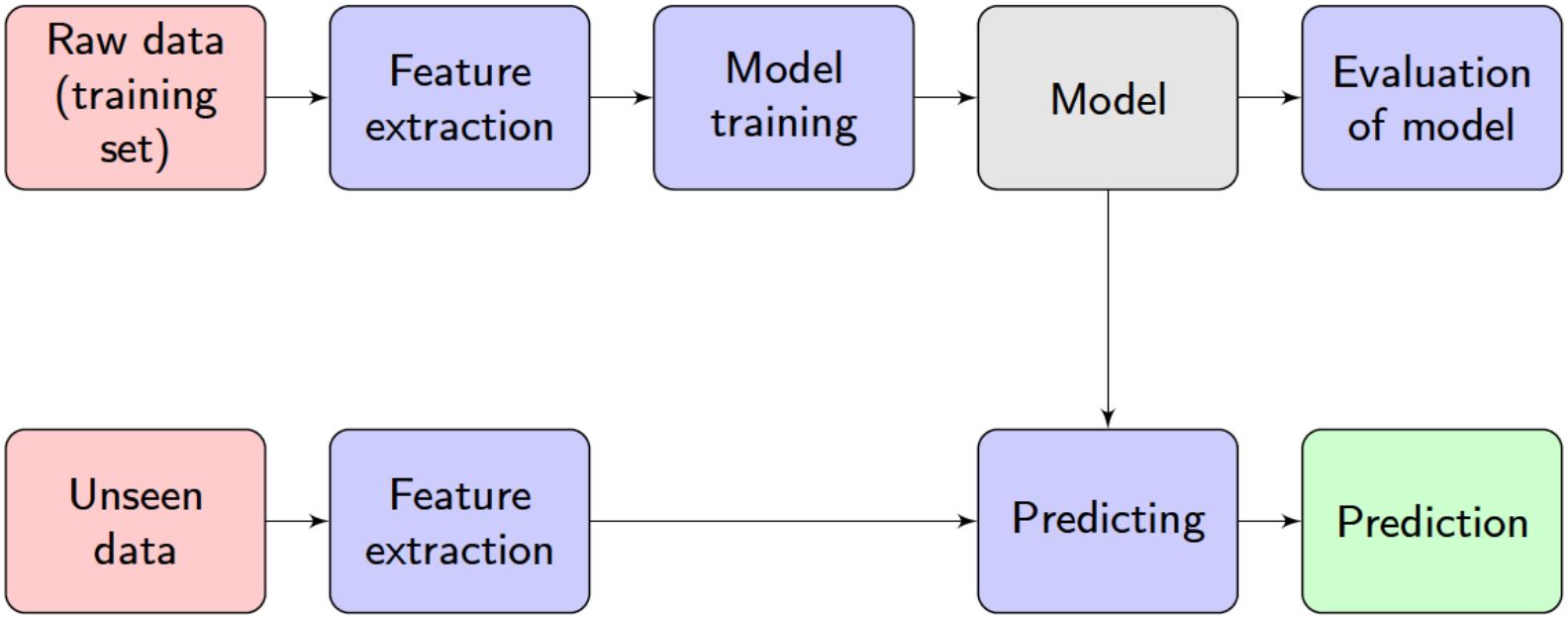
Binary classification since we have only two options.

Solution outline
Try to find a **decision boundary** (a curve) that separates admitted students from not admitted. Decide on what side of this boundary we have (45, 85).

Classification using logistic regression will be handled in Lecture 3.



Supervised learning workflow

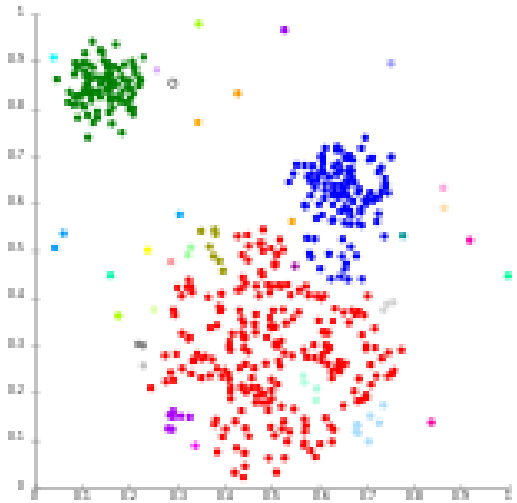


Feature extraction $\Rightarrow$ select data relevant for the problem

Unsupervised learning

Find structure in a given *unlabeled* dataset **X**.
E.g. group customers into different categories based on their purchase patterns.

Example: Clustering "similar" objects



Unsupervised learning is often referred to as a *descriptive task* whereas supervised learning is a *predictive task*.

Most of the course we will focus on supervised learning. Rafael will during the final three lectures talk about unsupervised learning.

Notations and Vocabulary

Chapter 1 and 2 in the textbook by Lindholm *et al.* introduces a number of important concepts that are further elaborated throughout the book.

- Datasets and Matrix Notations
- Different Attribute Types
- Parameters vs Hyper-parameters
- Model Evaluation
- Over- and Under-fitting

In this section we take a brief look at these concepts, they will be further discussed throughout the course.

Note: The mathematical notations used by Lindholm *et al.* is to start with rather heavy/clumpsy. But take your time and get used to it. You will need it!

Machine learning datasets

Problem

Predict the height of a girl with parent heights (60, 70) inches given a dataset of 214 tripples of heights (Mom, Dad, Girl).

	A	B	C	D
1	**Mom**	**Dad**		**Girls**
2	66	71		66
3	62	68		64
4	65	70		64
5	66	76		69
6	63	70		66
7	61	68		63
8	64	69		68
9	62	66		65
10	70	73		64
11	70	75		65
12	63	70		66
13	68	69		68
14	60	77		66
15	61	65		60
16	59	62		60
17	62	63		60
18	60	72		64
19	62	74		70
20	61	70		64
21	61	70		63
22	60	67		61
23	...	...		...

Notations

- Each row is a *sample* or *observation*
- We use the notation $\mathbf{x_i}, y_i$ for a general sample.
- Mom and dad heights (66,71) are *features*, *attributes* or *input*
- The girl height (66) is a *response* , *label* or *output*
- Data used to build the model is the *training set*
- Notation for training sets: $\mathcal{T}$ or $\{\mathbf{x}_i, y_i\}_{i=1}^n$
- Data used to evaluate the model is the *test set*

Matrix Notations

We will often use matrix notations to represent our datasets. Matrix $\mathbf{X}$ are the features, vector $\mathbf{y}$ is the response.

$$\mathbf{X} = \begin{bmatrix} 66 & 71 \\ 62 & 68 \\ 65 & 70 \\ \vdots & \vdots \end{bmatrix}, \ \mathbf{y} = \begin{bmatrix} 66 \\ 64 \\ 64 \\ \vdots \end{bmatrix}$$

The feature matrix $\mathbf{X}$ is often further divided into feature vectors

$$\mathbf{X}_1 = \begin{bmatrix} 66 \\ 62 \\ 65 \\ \vdots \end{bmatrix}, \ \mathbf{X}_2 = \begin{bmatrix} 71 \\ 68 \\ 70 \\ \vdots \end{bmatrix}$$

	A	B	C	D
1	**Mom**	**Dad**		**Girls**
2	66	71		66
3	62	68		64
4	65	70		64
5	66	76		69
6	63	70		66
7	61	68		63
8	64	69		68
9	62	66		65
10	70	73		64
11	70	75		65
12	63	70		66
13	68	69		68
14	60	77		66
15	61	65		60
16	59	62		60
17	62	63		60
18	60	72		64
19	62	74		70
20	61	70		64
21	61	70		63
22	60	67		61
23	...	...		...

Remember: One row (sample) is denoted $\mathbf{x}_i, y_i$, e.g., $(66, 71), 66$.

Feature Types

Discrete versus continuous features

Discrete features has a finite (or countably infinite) set of values, whilst continuous take real numbered values.

- Example: Time measured in hours is discrete, but time measurements in general is continuous.
- Example: The number of *bilar* in a package of Ahlgrens bilar is discrete, but the weight of the bag is continuous.

Numeric, Nominal, or Ordinal

There are several types of data or there are several types of domains of the features, which can be characterized in several different ways.

Name	Example	Description
Numeric	Temperature in this room	Numeric value
Ordinal	Grades (e.g A-F)	The objects can be ranked by this feature
Nominal	Color (e.g. blue)	The object can be categorized (but not ranked) by this feature

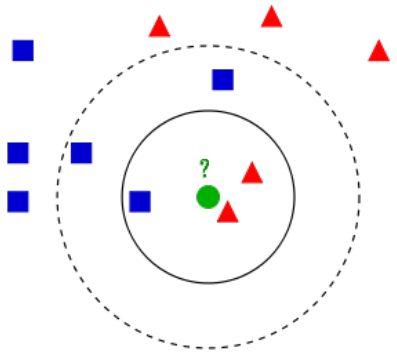
A quick break

...

The k-nearest neighbors (k-NN)

Given an odd positive integer k and a test observation $x_\star$

- identify the k points in your training data closest to $x_\star$
- assign $x_\star$ the class which is most common in the set of neighbors.



- $k = 3$: Three closest samples are red, red, blue $\Rightarrow$ we classify it as red
- $k = 5$: Five closest samples are red, red, blue, blue, blue $\Rightarrow$ we classify it as blue

k-NN is a very simple model. Can be used for both classification and regression.

- Classification $\Rightarrow$ Majority decision for the labels among k closest samples
- Regression $\Rightarrow$ Mean y_i value among k closest samples

k-NN Classification in Pseudo Code

We can implement k-NN classification by following the steps below:

1. Load the data
2. Initialise the value of k
3. Assume we want to classify a data point z
4. Find the k nearest neighbors to z
   - 4.1 Calculate the distance between point z and each row in the training data.
   - 4.2 Sort the calculated distances in ascending order based on distance values
   - 4.3 Get top k rows from the sorted list
   - 4.4 Get the most frequent class (called mode) of these rows
   - 4.5 Return the predicted class

Imagine it as operations on spreadsheet (e.g. Excel).

Algorithm taken from easy-to-read tutorial

`www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/`

that also provides simple Python code.

k-NN: Closeness

What do we mean by two object *being close* to each other?
Many options are available. But we will mainly use two well known *distance measures* (or *norms*).

Assume $\mathbf{x}$ and $\mathbf{y}$ are two vectors in $\mathbb{R}^n$, then

1. The *Euclidean distance* between x and y is defined as:

$$||\mathbf{x} - \mathbf{y}||_2 = \sqrt{(x_1 - y_1)^2 + \ldots (x_n - y_n)^2}.$$

2. The *Manhattan distance* (or *taxicab norm*) between x and y is defined as:

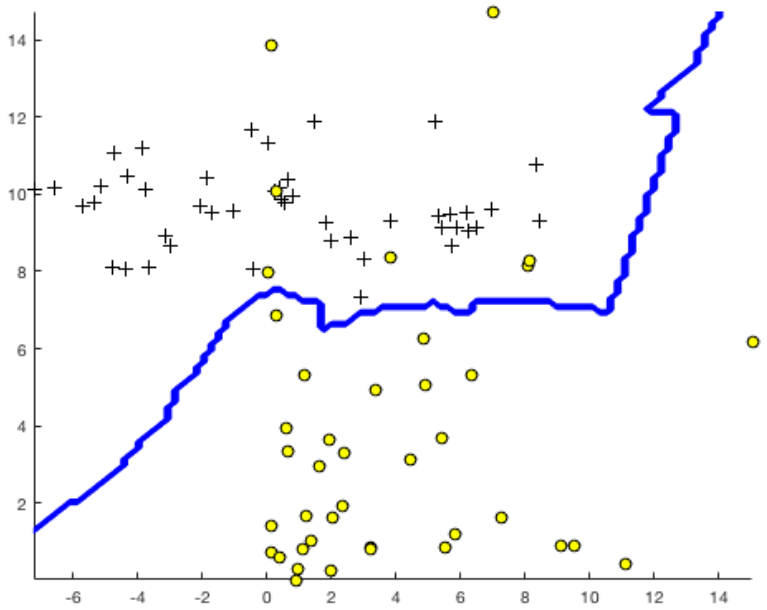
$$||\mathbf{x} - \mathbf{y}||_1 = |x_1 - y_1| + \ldots |x_n - y_n|.$$

We here use the same notations as Lindholm *et al.* that refer to them as L_2 norm ($||\mathbf{x} - \mathbf{y}||_2$) and L_1 norm ($||\mathbf{x} - \mathbf{y}||_1$).

- The choice of distance measure often depends on the task at hand.
- The Euclidian distance is intuitive but (due to the square) sensitive to outliers.
- k-NN is sometimes very sensitive to the choice of distance measure.

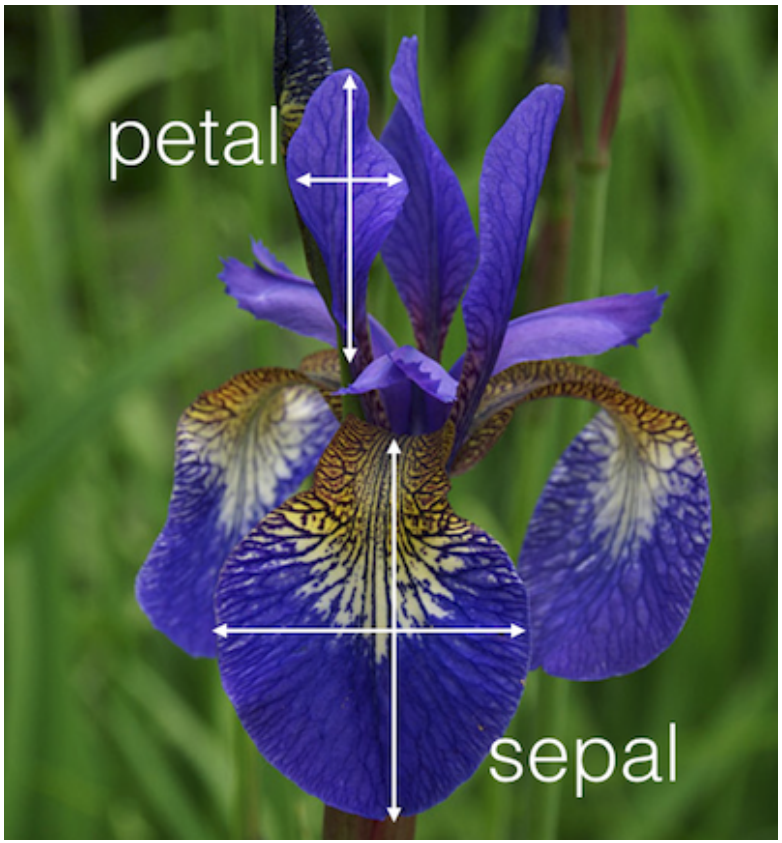
Decision Boundary for k-NN

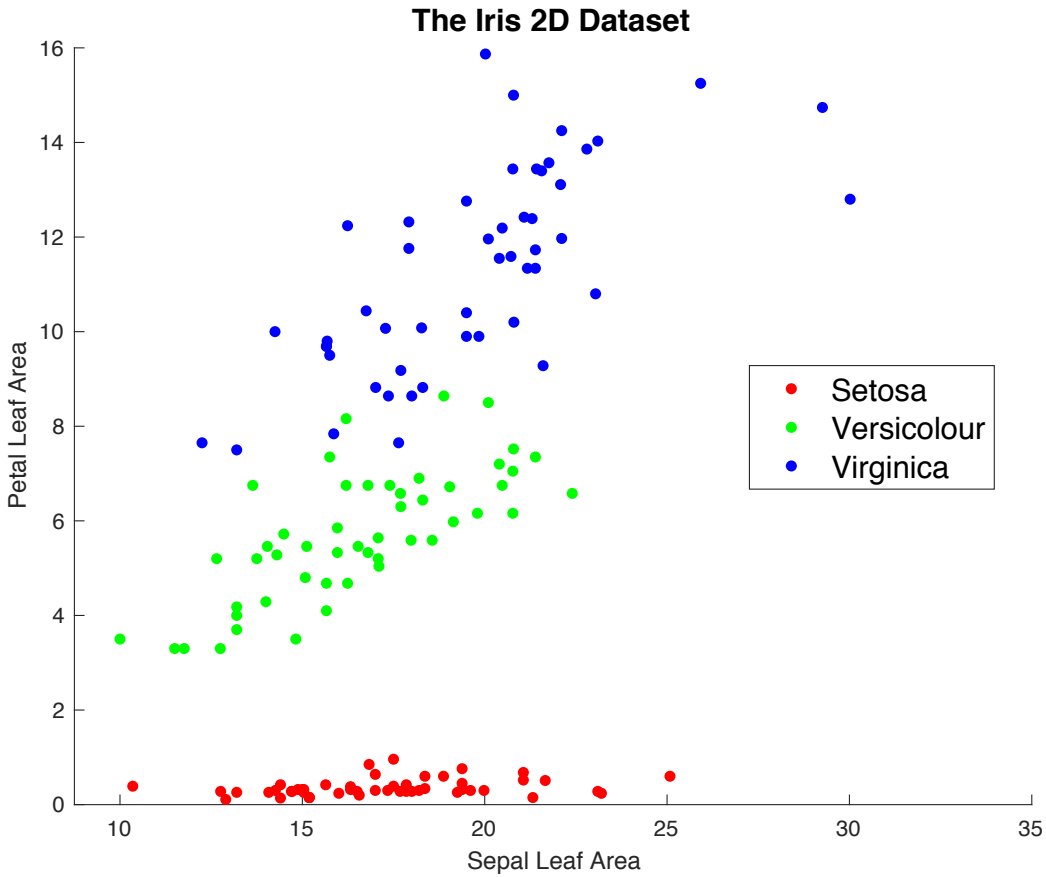
Below is an example of binary classification using 5-NN. The top-most section in the picture is the part where at least three of the five closest points are positive (+), and vice versa for the lower part of negative examples (o). The boundary between these two sections is the *decision boundary*.



A typical choice of k is 3, 5, or 7. Setting $k = 1$ makes it very susceptible for *noise*, and if k is large the local information becomes less relevant.

Example: Iris2D

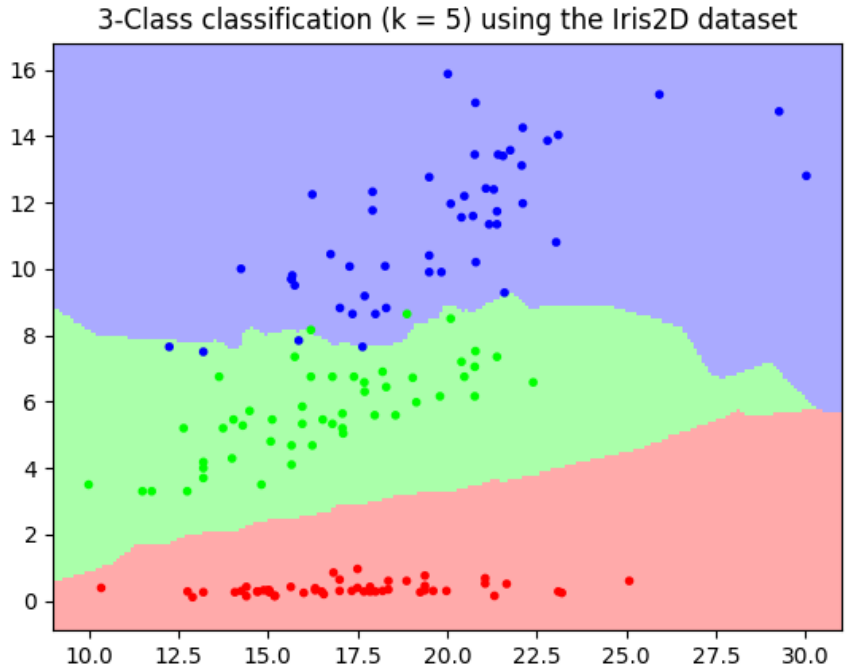




Problem: Identify three subspecies of Iris based on petal and sepal leaf areas.

Example: Iris2D - Decision Boundary

Mesh decision boundary for the Iris2D dataset using 5-NN

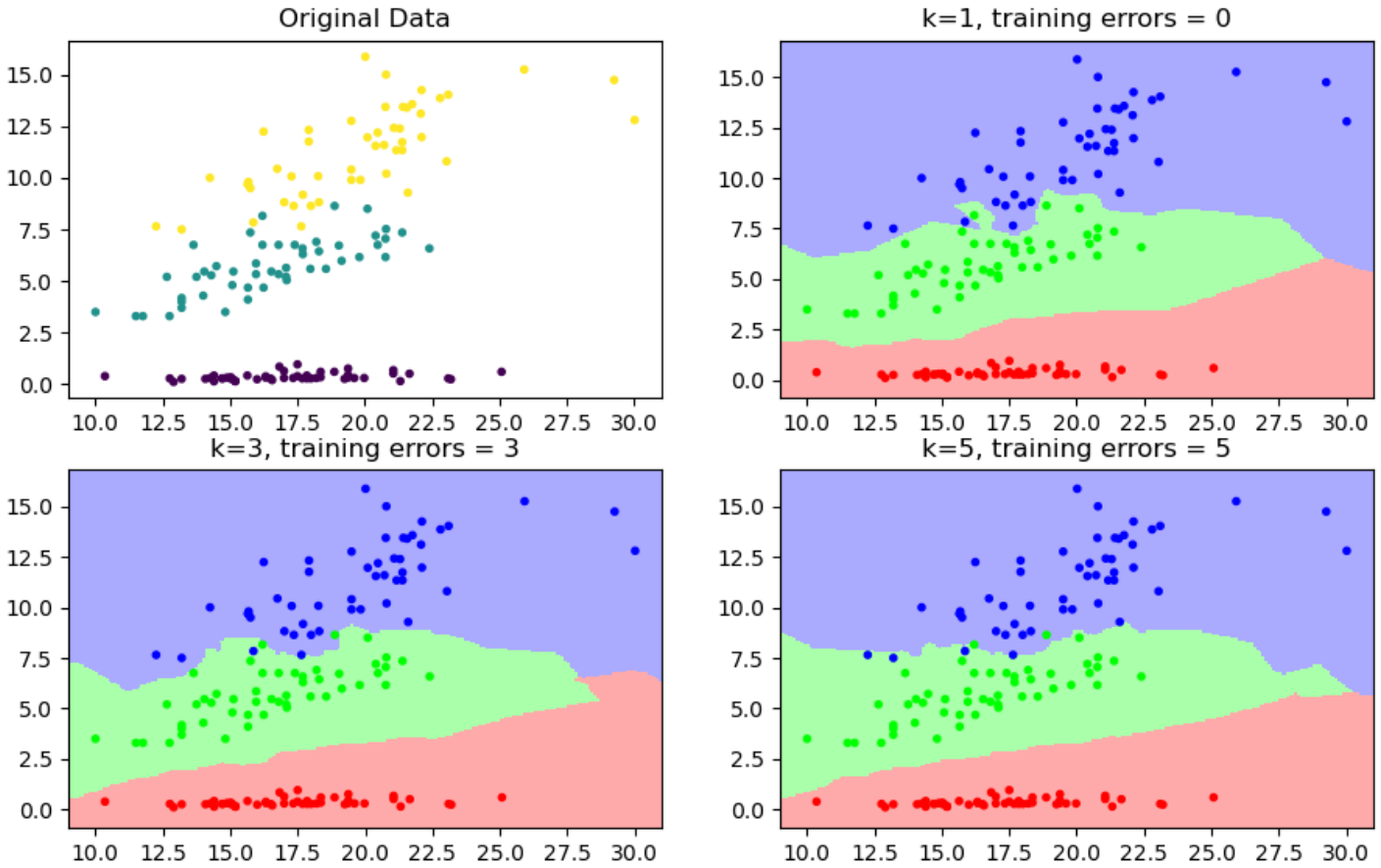


Mesh $\Rightarrow$ classify each point in a 200$\times$200 grid.

Notice also that we have five training errors. Four blue dots in the green area and one green dot in the blue area.

Training error: Errors found when classifying the training set

Example: Iris2D $k = 1, 3, 5$ with errors



Which k gives the best result?

k-NN Regression in Pseudo Code

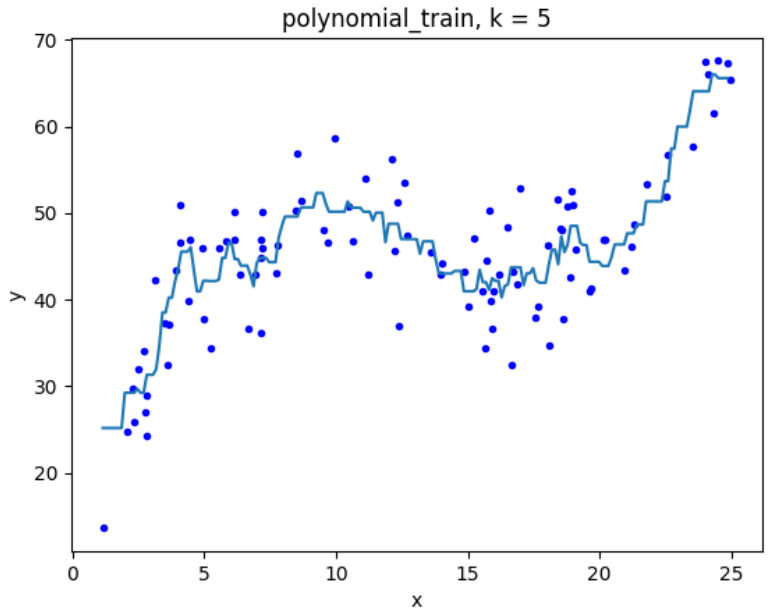
We can implement k-NN regression by following the steps below:

1. Initialise the value of k
2. Assume we want to predict the value of a data point z
   For example, predict height of girl with parent heights 60, 70.
3. Find the k nearest neighbors to z
   - 3.1 Calculate the distance between point z and each row in the training data.
   - 3.2 Sort the calculated distances in ascending order based on distance values
   - 3.3 Get top k rows from the sorted list
   - 3.4 Return the mean value of the labels of these rows

Example: Use $k = 3$ to predict height of girl with parent heights 60, 70.

- The three closest samples are (60, 72; 64), (61, 70; 64), (61, 70; 63)
- The mean value of girl heights among 3 closest is 63.7.
- We therefore predict 63.7.

k-NN regression - Plotting the Result



- Above: A $k = 5$ fit to a given dataset (x_i, y_i)
- Plot:
  - Divide x-axis interval $[1, 25]$ into (say) 200 equidistant points X_j
  - For each such X_j: find the 5 data points in dataset closest to X_j
  - Compute average corresponding y-value for the 5 selected data points $\Rightarrow Y_j$
  - Plot X_j, Y_j

Parametric and vs Non-Parametric Learning

Two approaches in estimating $\hat{f}$: *Parametric* and *Non-Parametric Learning*

Parametric regression example: Assume $\hat{f}(x)$ of the form

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3.$$

We then use the training set to find θ_i that makes the best possible fit. And later use $\hat{f}(x)$ to make predictions

k-**NN is Non-Parametric.** We do not build a model. We use the entire dataset to make a prediction.

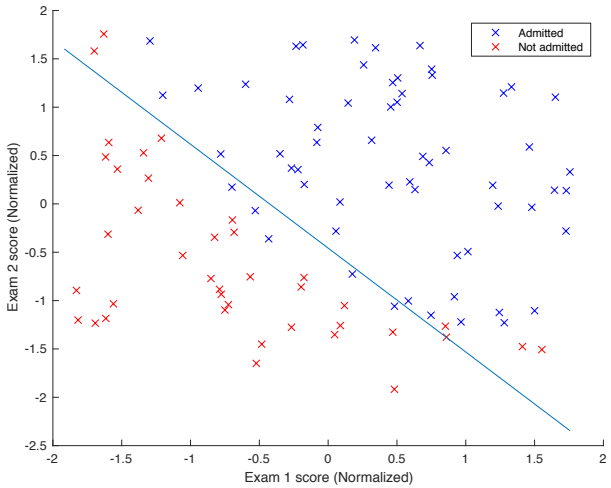
	Parametric Learning	Non-Parametric Learning
What?	Assume some fixed set of parameters θ which describes future predictions independent of the observed data.	No model, we use entire training set to make a prediction
Pros	Easier to model, fast	No assumptions on f, slow
Cons	Not flexible $\Rightarrow$ Chosen model might not fit the data perfectly	Overfitting $\Rightarrow$ bad at adapting to new unseen data

k in k-NN is a so-called **hyper-parameter** $\Rightarrow$ not part of model, set before learning process begins. For example, number of iterations used to find best fitting θ_i in regression example above.

k-NN is an exception, we will mainly focus on parametric models in this course.

Flexibility and interpretability

Some models are less flexible than others, for example a straight line decision boundary.



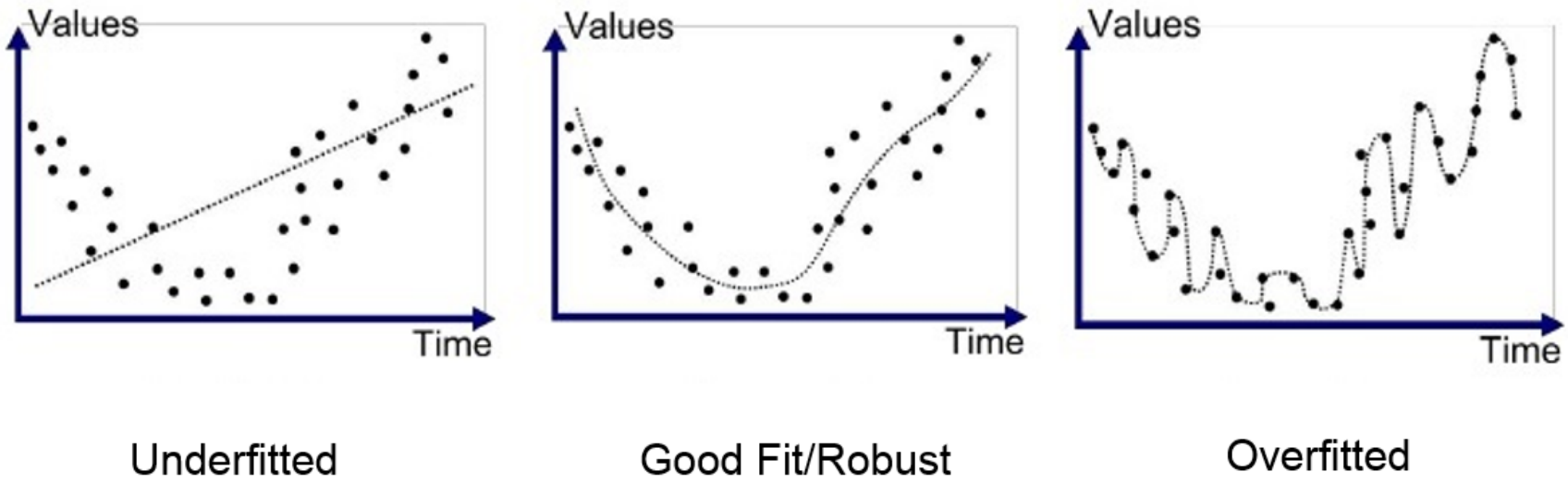
Not a very flexible model, but it's very interpretable, the response Y is a linear function of the variable X.

k-NN in contrast is very flexible $\Rightarrow$ adapts to any type of data.

Other models are not as interpretable, such as *support vector machines*, but very flexible in the sense that it can capture many different forms of data.

Interpretability is a hot topic. Would you like to get a job application rejected by a machine without any motivation?

Over- and Underfitting



- **Underfitting**: Model not flexible enough. Large test and training errors no matter what size of dataset
- **Overfitting**: Model much too flexible $\Rightarrow$ makes unrealistic fitting to given training set $\Rightarrow$ Very small training error and large test errors. k-NN (for small k) often suffers from overfitting.

We aim for good robust fit that makes a reasonable fit to the given dataset

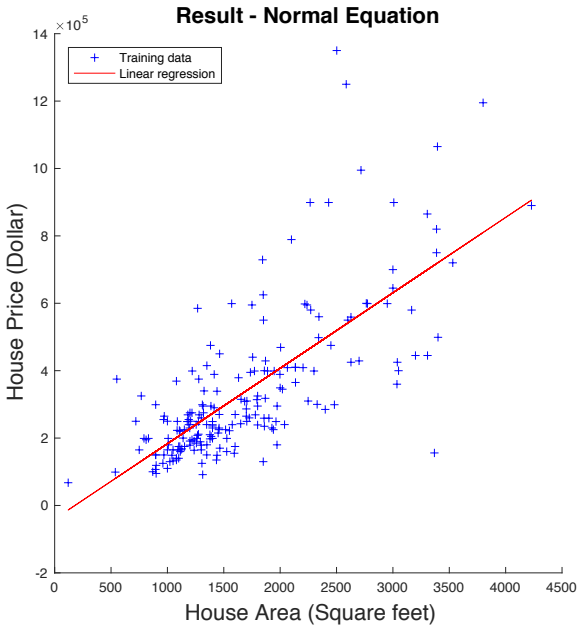
Quality of Regression Models - MSE

Assume that we have found a model

$$y = \hat{f}(x)$$

using some regression method.

Q: Is the model good? How do we compare it with other models?



- $(y_i - \hat{f}(x_i))^2 \Rightarrow$ squared y-distance between estimate $\hat{f}(x_i)$ and actual value y_i
- The most common regression error measure is *mean squared error* (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2.$$

- The MSE will be small if the discrepancy between the estimated model $\hat{f}$ and the actual value y_i is small over all training examples.

Test set and Test error

We typically use 80% of the labeled data as *training set*,
and 20% of the labeled data as *test set*.

Training Set

- We build the model using the training set
- The training set is also used to select which type of model to use ...
- ... and to fine-tune the hyper-parameters
  $\Rightarrow$ We get a model suitable for the training set
- The error measured by the training set is called the *training error*

Test Set

- We evaluate the model (built using the training set) using the test set
- The test set is also known as the *gold standard* set and represents unseen data
- Important: The test set must not be used for training
- The error measured by the test set is called the *test error*
- Hence, we determine whether a model is 'good' by it's test error

Feature Normalization

- A common problem for many ML algorithms is if the values in different features are vastly different. E.g. in range $[0, 1]$ in one feature, and in range $[10000, 50000]$ in another.
- For example, the euclidian distance measure in k-NN would basically just take into account the "large" feature.
- Solution: Normalize feature data $\Rightarrow$ all features of similar size
- **Note**: I do this all the time just to be sure. I start by normalizing all features.

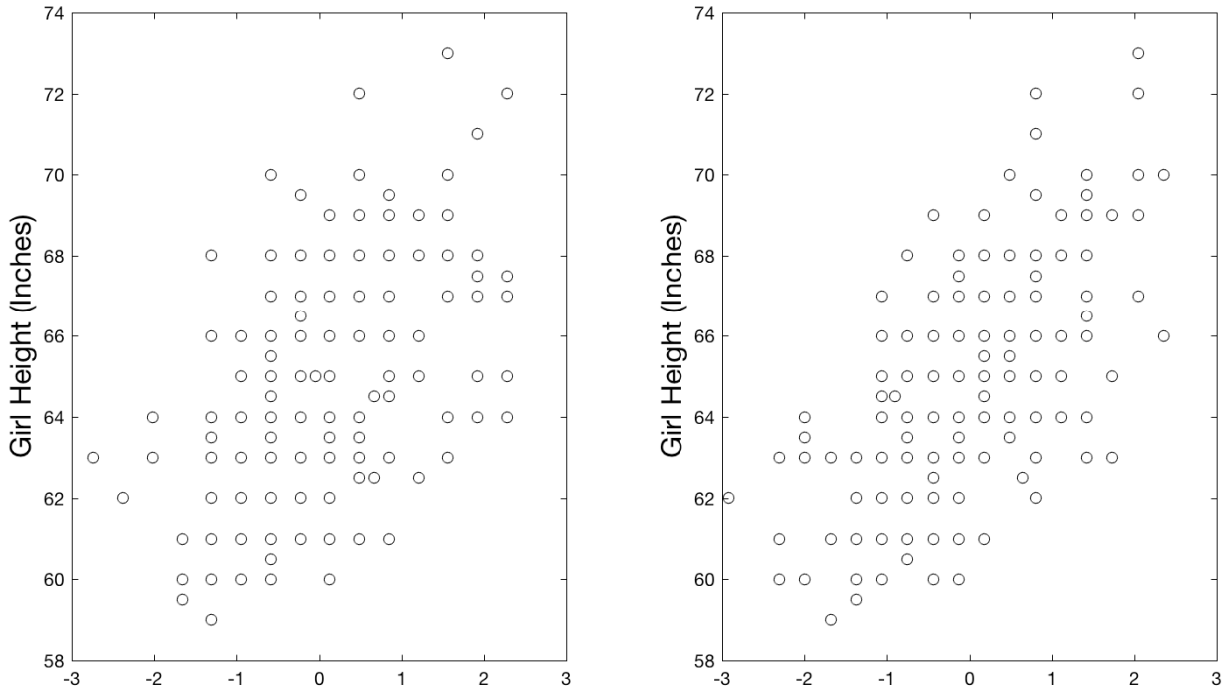
Feature Normalization For each feature X^i in $\mathbf{X}$

1. Compute mean μ_i
2. Compute standard deviation σ_i
3. Compute normalized X_n^i as $X_n^i = (X^i - \mu_i)/\sigma_i$
4. Build normalized matrix $\mathbf{X}_n = [X_n^1, X_n^2, ..., X_n^p]$ and continue

After normalization each feature X_n^i will have a mean value of 0, and a standard deviation of 1.

Girls Height – Feature Normalization

After normalization with $\mu = [63.63, 69.41]$ and $\sigma = [2.79, 3.22]$



Notice

- Mom vs Girl (left) and Dad vs Girl (right) after normalization
- All feature values (mom and dad heights) now centered around 0
- All features have the same spread (standard deviation 1)