



Introduction to Bioinformatics

Professor
Gregory R. Grant

Topic 18 Machine Learning

December 4th-6th, 2023

Gregory R. Grant

Genetics Department

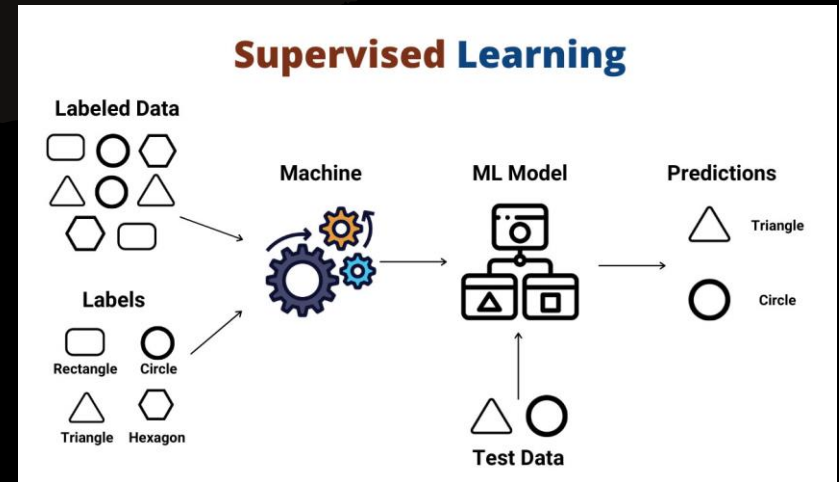
ggrant@pennmedicine.upenn.edu

Teaching Assistants
Chetan Vadali

ITMAT Bioinformatics Laboratory
University of Pennsylvania

Supervised Learning

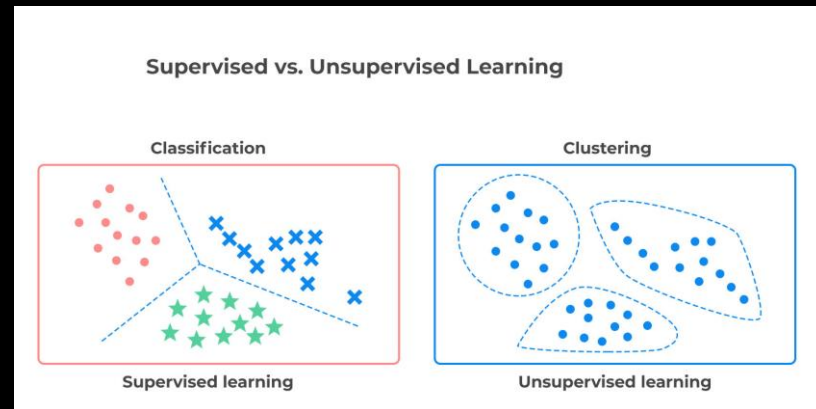
- Machine Learning divides into two main types, supervised and unsupervised.
- Supervised learning is when you have training data where you know the “truth”.



- The truth means you know both the values of the independent *and* the dependent variables.
- The goal is to teach a machine how to infer the value of the dependent variable from independent variables.
- That will be done on future data where we know the values of the independent but do not know the corresponding dependent.

Unsupervised Learning

- Unsupervised learning is when you have data, and you want to search for *unknown* relationships between various subjects.



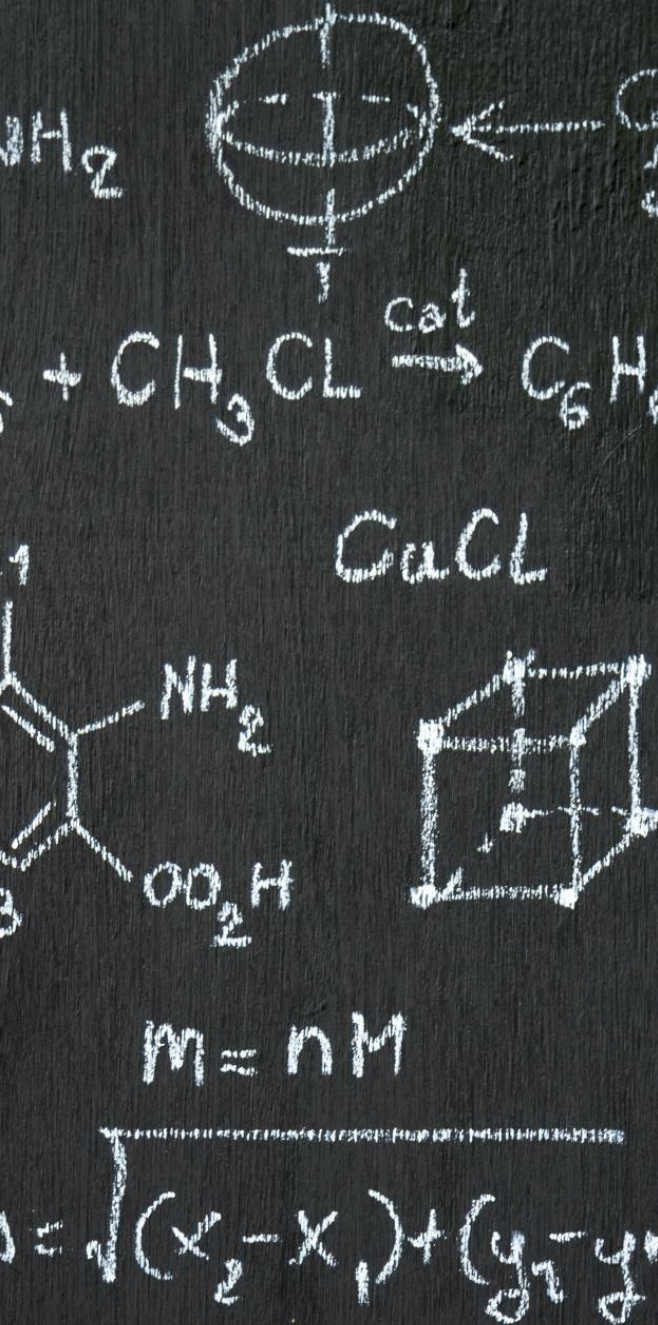
- For example, single cell RNA-Seq reveals novel cell types.
- We're going to talk about supervised learning first.

Unsupervised vs. Supervised

- Unsupervised learning — determine how many disease subtypes are there
- Supervised learning — predict disease subtype using test results as features

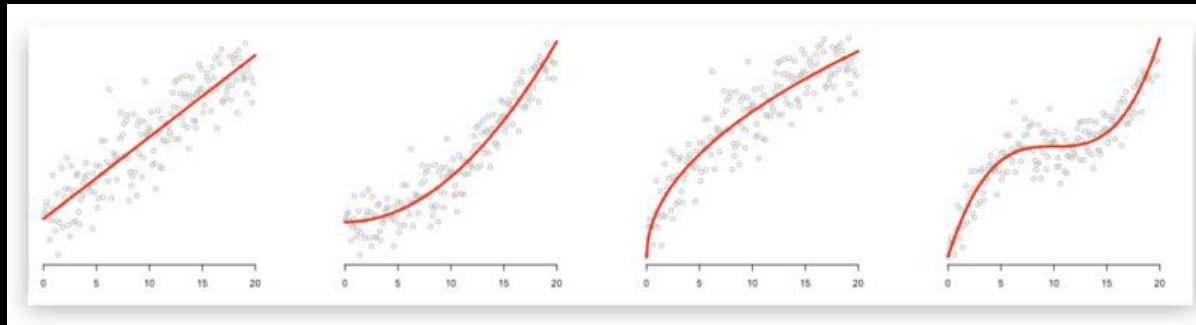
What is a Machine?

- A machine in this context is a mathematical construct.
- Such a construct is usually called a “model” and involves specifying two things.
 1. The form of the model.
 2. The parameters.
- For example:
 - The form might be a straight line in the X - Y plane.
 - The parameters are the slope and Y -intercept.
- This simple example is the model underlying *simple linear regression*.
- Models can be as simple or as complex as the problem at hand requires.



What is Learning?

- The “learning” part is where we use data to specify the model.
- But it’s (usually) not the *form* of the model that is “learned” from the data, it’s the parameters.
- We might plot the data first to help decide on the best model (line, parabola, etc.) but strictly speaking, that’s not the “learning” part, even though it can also involve the data.



The Learning Step

- The “learning” part is where we determine the parameters, once the shape (form) has been established.
- For simple linear regression in the X - Y plane, we plot the data as points and then fit a line to it.
 - Fitting the line is the learning part.
 - Deciding to model the data with a straight line was preliminary to learning.
- If we were modelling with a parabola and not a straight line, then there’d be three parameters:
$$y = a + bx + cx^2$$
 - x and y are the variables.
 - a , b and c are the parameters.

The Form of the Model

- In machine learning, the form of the model is typically determined by a human.
 - The form is based on what a human thinks would work best for the problem at hand.
- Mother Nature does generate data according to some specific distribution.
 - Known only to Mother Nature and God.
 - In theory that one distributional form is the one right model, and all others are wrong.
- But in practice, Mother Nature's distribution may be intractably complex.
 - And we only need an approximation to achieve our goals at hand.
- Which model is best is always debated. Each person will develop preferences.
 - So, it is an art as much as a science.



Two Types of Problems

























- Machine Learning is for two types of problems.
 1. Prediction
 2. Classification
- **Example of prediction:** predict a person's lifespan from three variables: BMI, income and number of cigarette's smoked per day.
 - In this case what is produced is a value from a continuum (age can be any number between 0 and 130).
- **Example of classification:** determine a disease state from several measurements of blood.
 - In this case what is produced is binary (healthy/affected).
 - Or it may just return probabilities of being in each category.
 - But ultimately, it's trying to predict a category, not a number.
- In both cases, something is determined (output) from a set of measurements (input).



Types of Models

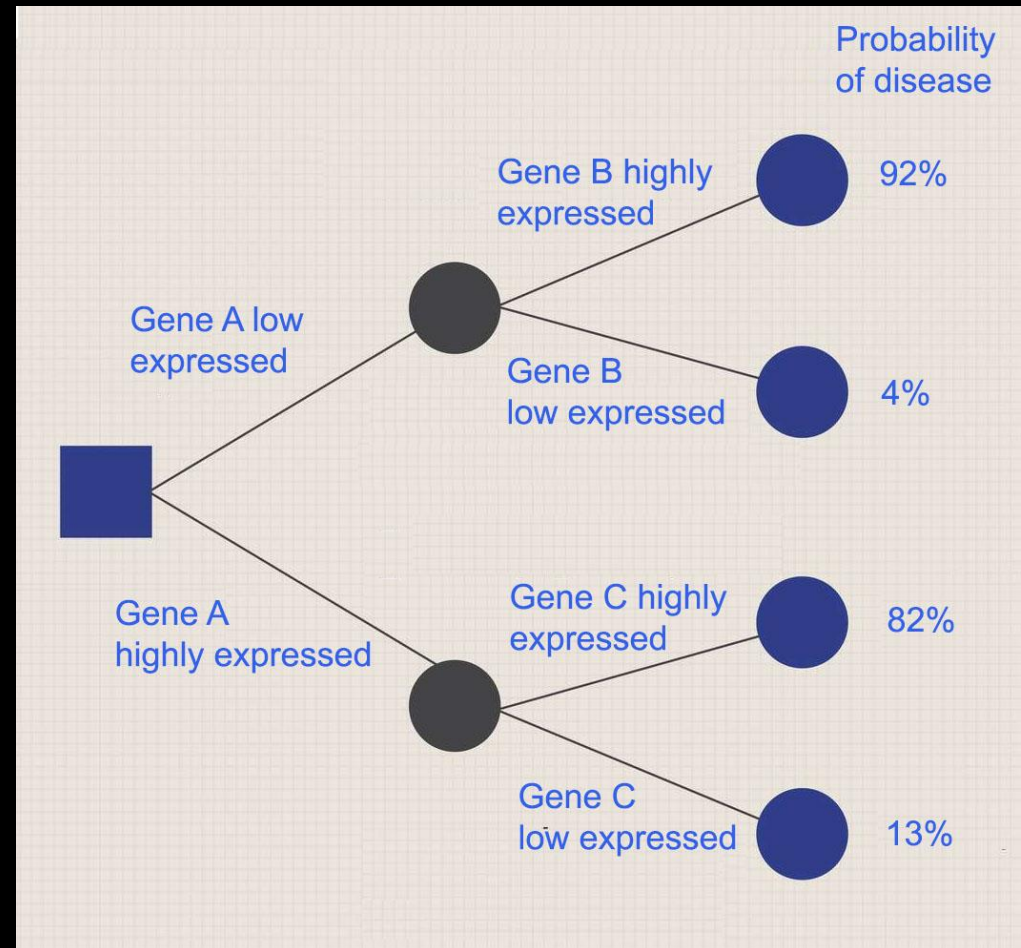
- Much of machine learning is based on regression.
- But there are many other types of models.
- They have little in common except that each has a set of parameters that are “learned” from “training data” where the truth is known.

Types of machine learning models

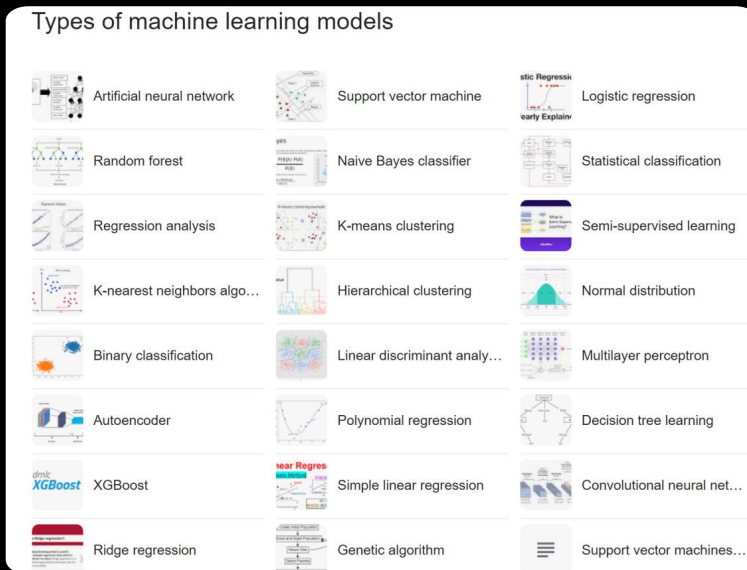
 Artificial neural network	 Support vector machine	 Logistic regression
 Random forest	 Naive Bayes classifier	 Statistical classification
 Regression analysis	 K-means clustering	 Semi-supervised learning
 K-nearest neighbors algo...	 Hierarchical clustering	 Normal distribution
 Binary classification	 Linear discriminant analy...	 Multilayer perceptron
 Autoencoder	 Polynomial regression	 Decision tree learning
 XGBoost	 Simple linear regression	 Convolutional neural net...
 Ridge regression	 Genetic algorithm	 Support vector machines...

Decision Trees

- As an example of a machine learning model completely different from regression, consider a decision tree.
- Here the parameters are the probabilities at the leaves.
- For a decision tree, the training process is straightforward:
 - Collect a bunch of data where the truth is known (the truth is gene expression measurements from genes A, B and C from a bunch of subjects all of whom have known disease state).
 - Count how many subjects are in each category.
 - Estimate probabilities as ratios.

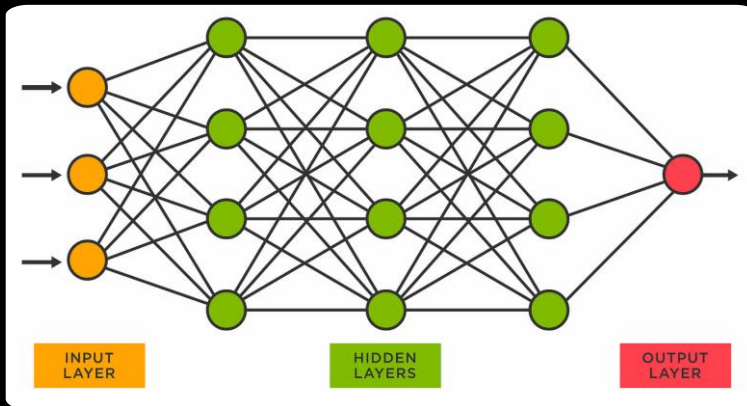


Considerations



- Every model is different.
 - But each has parameters that can be determined from data.
- Many considerations go into which type of model to use.
- Two main considerations are:
 1. How much data is available?
 2. How much do you care about understanding of *why* something is true.

Neural Nets



- Neural Networks are one type of model that has been very successful lately.
- Most of the recent attention in machine learning is due to things now being done with neural nets.
 - ChatGPT
 - Self-driving cars
 - Chess, Go and other games
 - Handwriting and voice recognition.
 - Language translation
 - Etc.

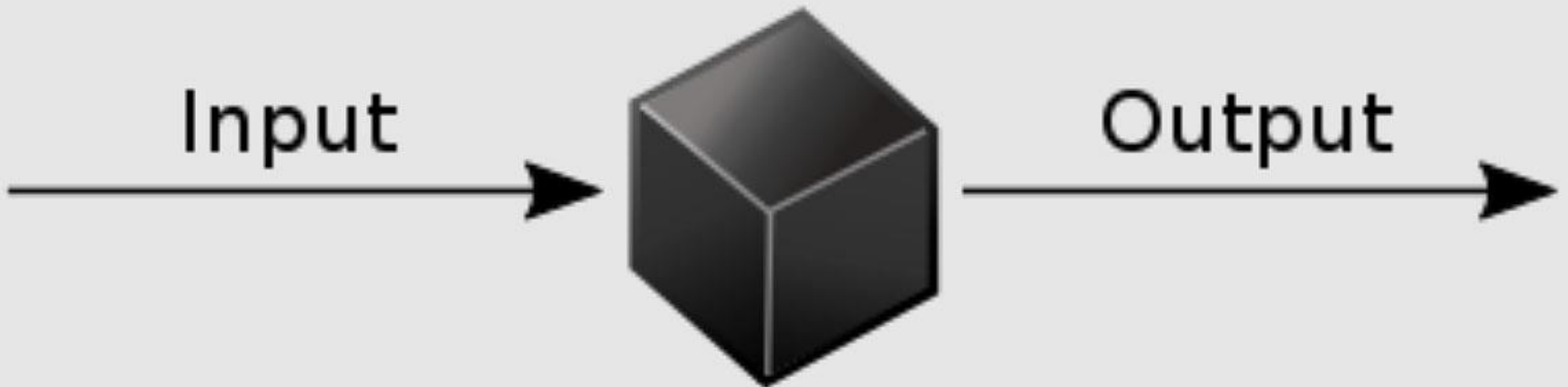


Why Now?

- The remarkable success of neural nets is due to what are called “deep learning” algorithms.
 - Neural Nets can have a *lot* of parameters.
 - Deep Learning Algorithms are algorithms for training the parameters of the model.
 - Neural Nets are complex models, but that’s why they can do complicated things.
- Neural networks have been around for a long time and so have deep learning algorithms. Decades.
- But they needed two things to come available in order to thrive, which only recently became widely available.
 1. Massive amounts of data.
 2. Massive compute power.
- Neural Nets aren’t appropriate for many problems in biology.
 - They’re good for some problems, but often massive amounts of data is hard to come by.

Interpretation

- Another downside of neural nets is that they are black boxes.
- Often, we model things in biology to gain an understanding of why things are true.
- In contrast to neural nets, when we perform regression, we can evaluate how the independent variables determine the dependent variables.
 - That information is encoded in the beta coefficients.



Natural Language

- for example -

- Suppose we want to understand *how* a human produces language.
- A neural network can be taught to speak remarkably similarly to a human.
 - For example, ChatGPT is amazing.
- But even if it passes the Turing test and the output is indistinguishable from a human by a human, it tells us nothing about how a human brain actually produces language.
 - We can't even understand how the neural net does it because it uses deep learning.
 - However it manages to do it, it has nothing to do with how humans do it.



A Time and a Place



If we could use deep learning to identify the causative SNP in a GWAS then that would be justified.



But once we know which SNP is causative, we wouldn't use deep learning to understand *how* the SNP causes the phenotype.

The mechanism of action.



We might however use *regression*, which is still machine learning, but can provide insight into the relations between the relevant variables.

Our Goals



WE WILL NOT BE DISCUSSING
NEURAL NETS OR DEEP LEARNING.



WE WILL BE DISCUSSING REGRESSION
MODELS AND CLUSTERING METHODS.

Types of Regression Models

Regression is used for both prediction and classification.

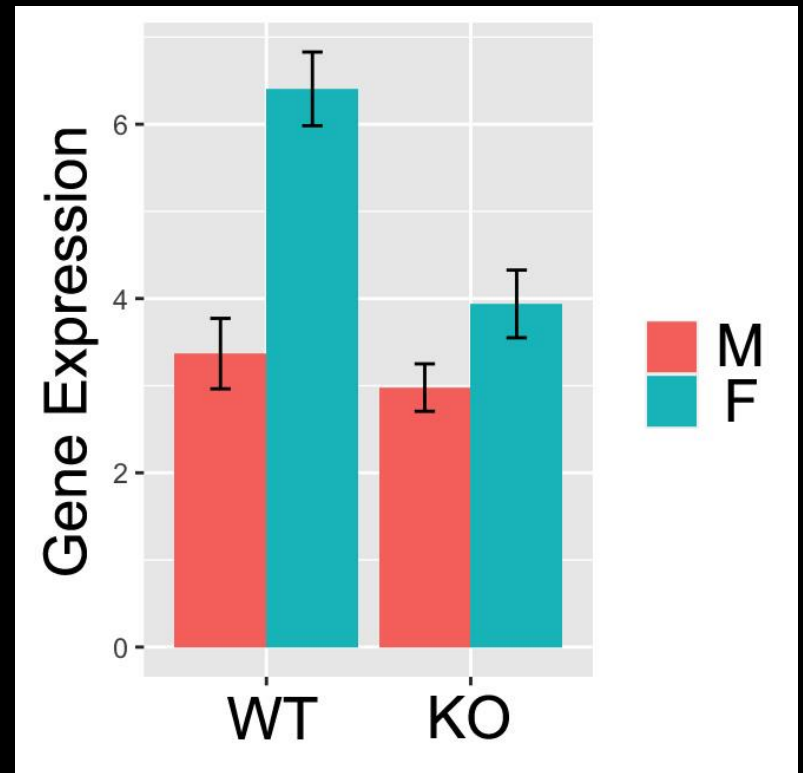
In other words, *dependent* variables can be both continuous or discrete.

Independent variables can also be continuous or discrete.

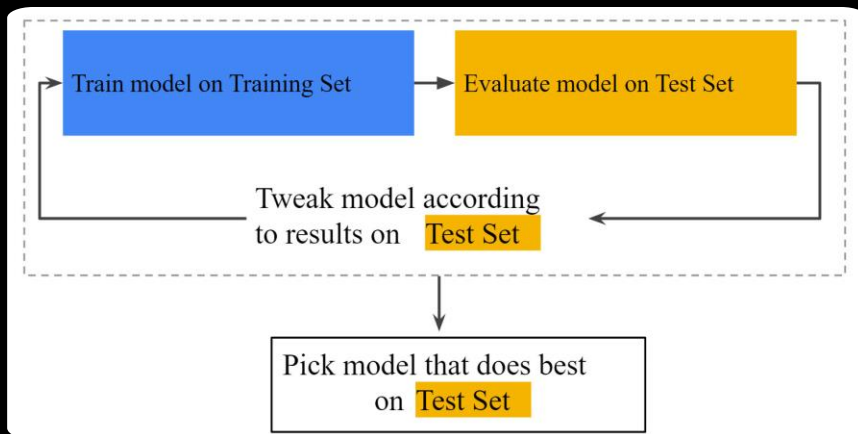
E.g., gene expression (continuous) or gender (discrete).

What is ANOVA?

- ANOVA stands for “Analysis of Variance”.
 - It’s just a flavor of regression.
 - If the dependent variable is continuous and all *independent* variables are categorical, then we’re doing ANOVA.
- For example, we may be interested in the expression of a gene (continuous dependent variable) as a function of gender and genotype (two independent categorical variables).
- Question: Is the gene DE between genotypes, independent of gender.

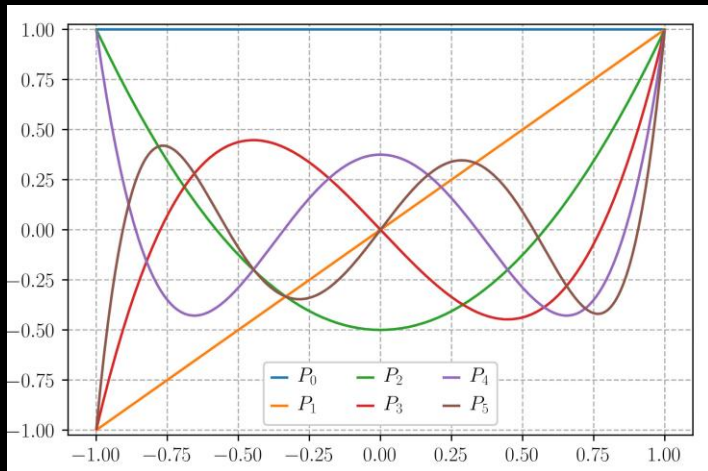


The Machine Learning Process



- The “learning” part involves data in two different ways.
 1. To estimate the parameters
 2. To evaluate performance
- In other words, we don’t just train the model, we also need to know how good it is.
 - There will always be some minimal accuracy below which the “machine” cannot be used for the intended purpose.
 - And if two models are both good enough, we still want to know which is better.
- Therefore, data is divided (at least) into two parts: Training and Test

Regression Choices



- **Decision #1:** to use a regression model
- **Decision #2:** to choose the *form* of the regression function.
 - For one independent variable, the regression function is a curve.
 - For two variables it's a surface.
 - For three variables it's called a "hypersurface".
 - In general, it's just called the "regression function".
- **Decision #3:** to choose the *family* of regression functions.
- Suppose there's one independent variable.
- We know the regression function is a curve.
- But we still don't have "parameters" to learn until we decide on the *family* of curves.
 - Family=Lines: 2 parameters
 - Family=Parabolas: 3 parameters
 - Family=Cubics: 4 parameters
 - N -th degree polynomials: $N+1$ parameters.



Other Families of Curves

- Regression curves need not be polynomials.
 - Could be a cosine wave, which has 4 parameters.
$$\alpha + \beta \cos(\gamma x + \theta)$$
 - 2β is the amplitude, $2\pi\gamma$ is the period, α and θ are vertical and horizontal shifts.
 - Could be exponential, logarithmic, rational, radical, there are *many* families of curves.
 - Could also be any combination of the above, for example a linear function plus a trig function
$$ax + b + \cos(cx^2)$$
-



Degenerate Curves

- Consider the general formula for a 2nd degree polynomial:

$$ax^2 + bx + c$$

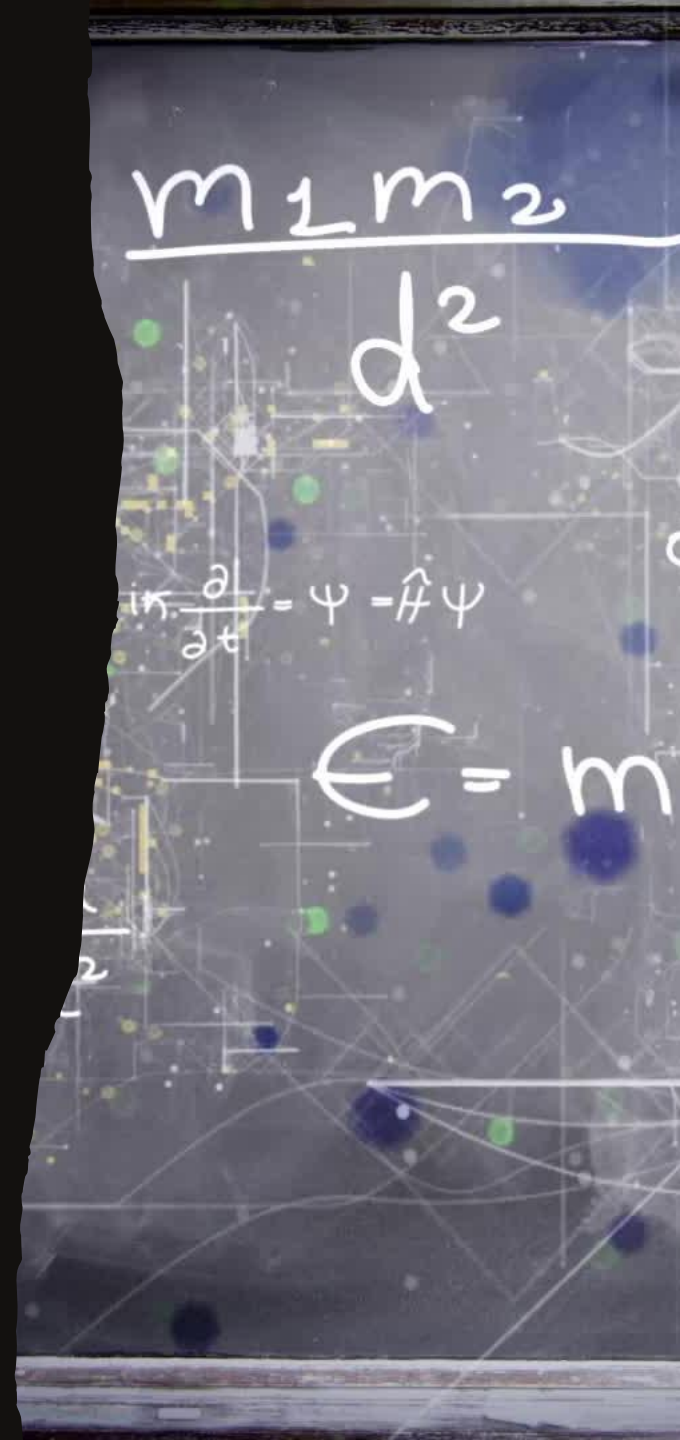
- A straight line is covered by this, by setting $a = 0$.
 - A straight line is called a “degenerate” parabola.

- Likewise, any linear or quadratic function is covered by the general formula for a cubic:

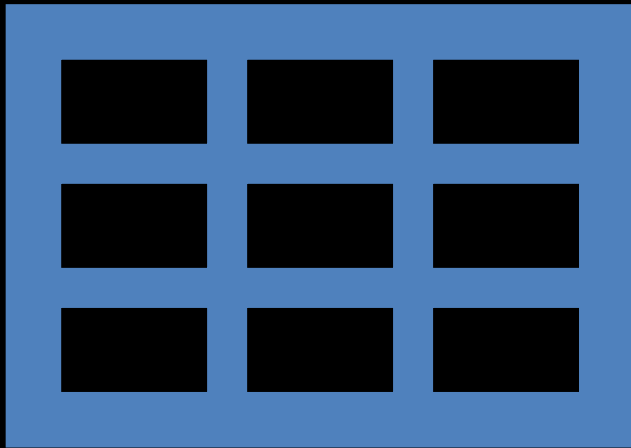
$$ax^3 + bx^2 + cx + d$$

Polynomial Degree

- Based on the previous slide, why do we have to decide whether the form of the model should be linear, or quadratic, or cubic, or etc.?
- Why not make the form of the model some high degree polynomial, since that covers all these cases?
- Several reasons.
- First off, a degree n polynomial requires at least $n+1$ points to fit to it.
 - We cannot fit a line to one point, a parabola to two points, etc.
 - So, the degree will be limited by the number of data points.
- But there's a much bigger reason, something called "overfitting".



Overfitting



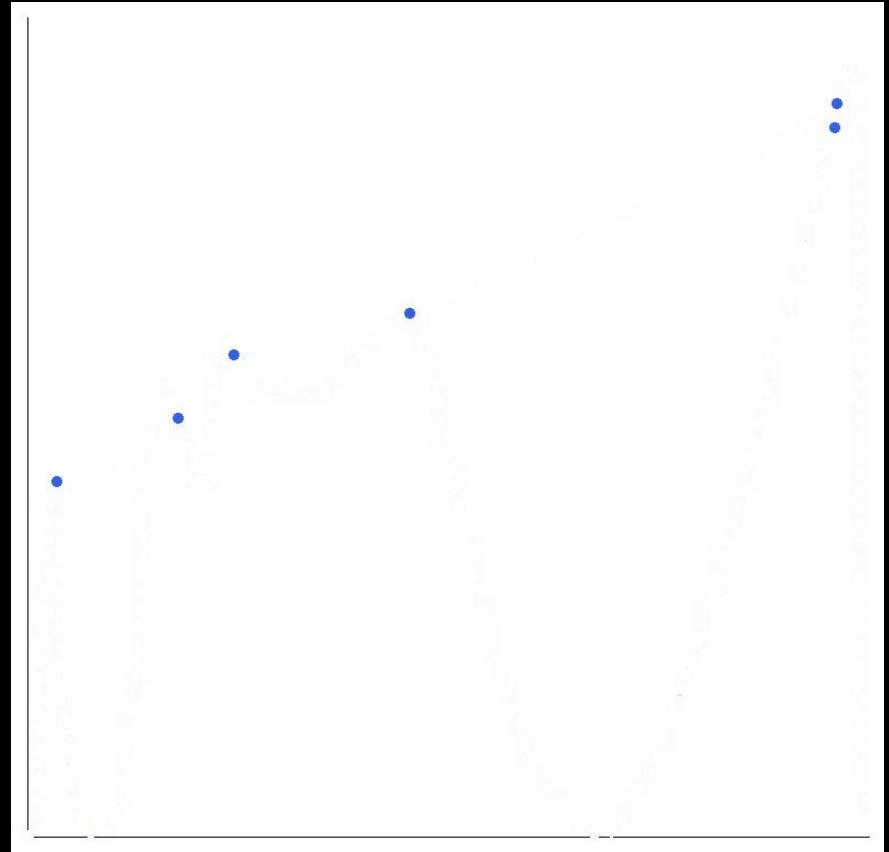
- Suppose there are n data points in the training data.
- Then a degree $n-1$ polynomial can be found that passes through all of them precisely.
- Such a curve would have a “least-squares error” equal to zero, so it would give the best possible fit.
- But it might still be unlikely to explain future data very well.

Overfitting Example

Suppose the data look like this.

The true model of the data is a straight line.

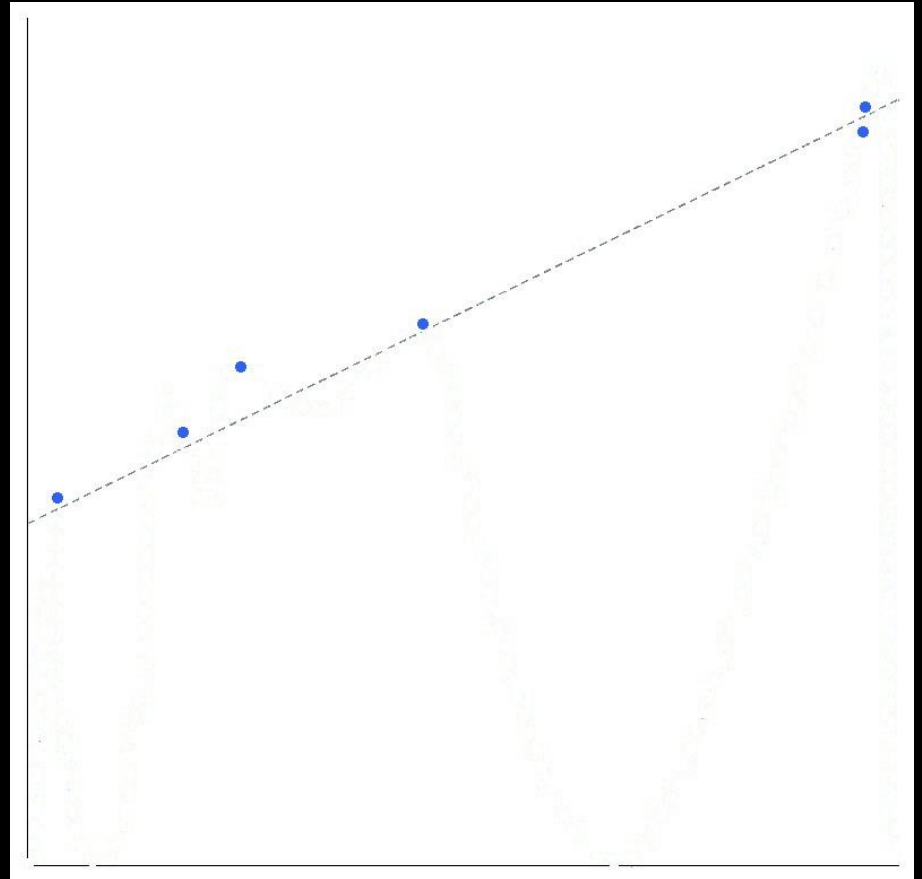
Assume Mother Nature generated the model according to the straight-line ($ax + b$), with some wiggle (\mathcal{E}).



Overfitting Example

A straight line fits the data well.

As it should, given it's the true model.



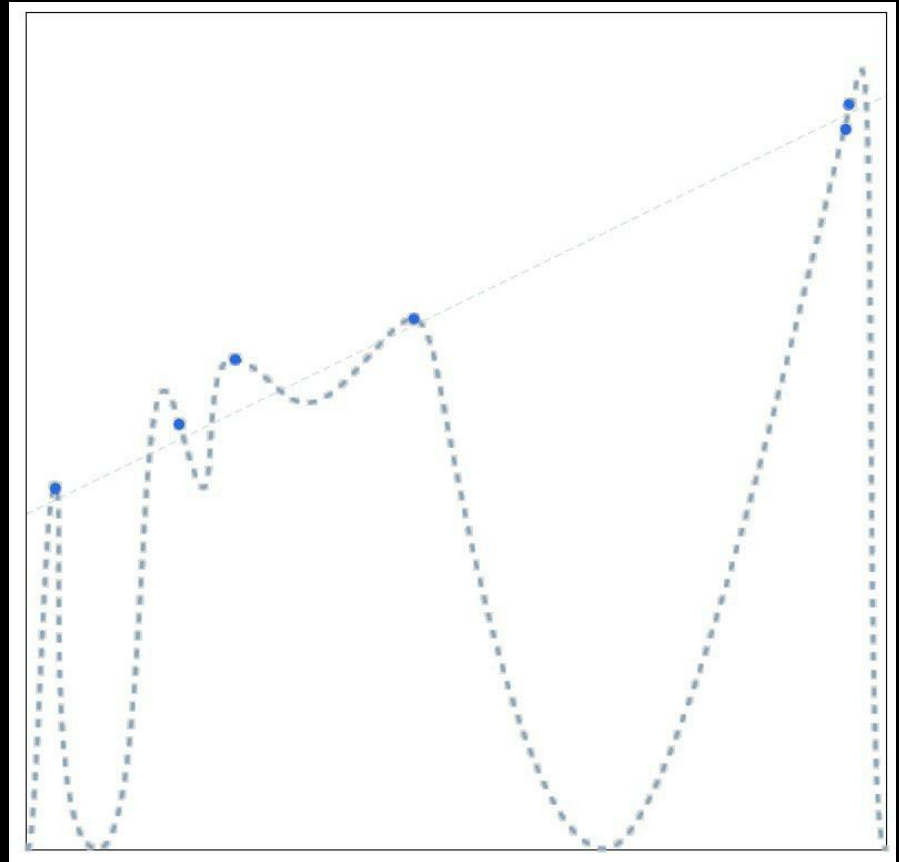
Overfitting Example

Suppose we decide to train a model with 10^{th} degree polynomials.

We assume (erroneously) that if a straight line is the best fit then it will make the higher degree coefficients zero in order to fit a straight line.

But the problem is, a 10^{th} degree polynomial can be made to go through all six points *precisely*.

(even with 5^{th} degree it could be made to go through them all)



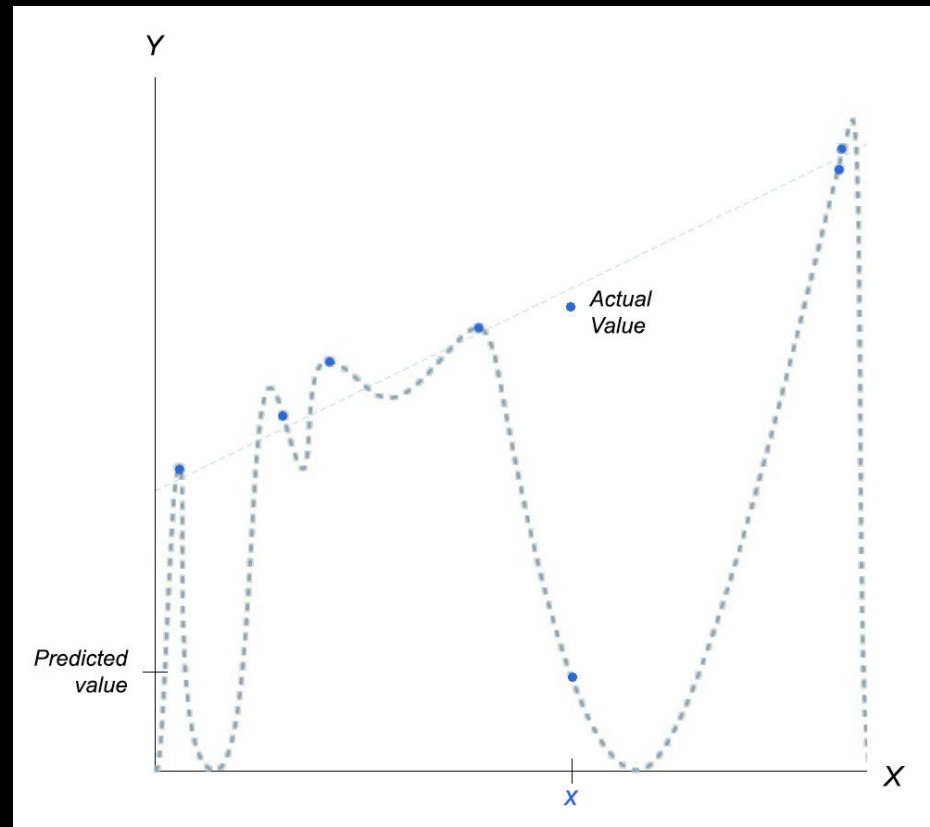
Overfitting Example

Since the true model of the data is linear, the 10th degree curve will give very bad future predictions for many values of x .

In language to be developed soon, we say the 10th degree model has low (in fact zero) “in sample” error but high “out of sample” error.

The straight line on the other hand has low (but non-zero) “in sample” error but also “low out of sample” error.

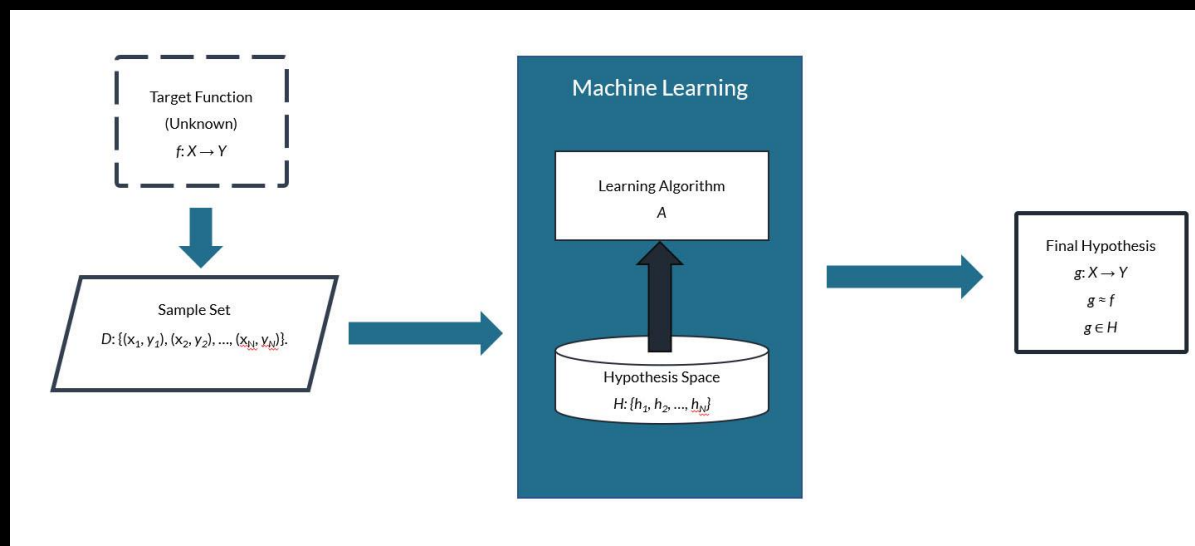
The 10th degree model suffers from overfitting.



- We train models to have low in sample error.
- We evaluate their performance by their out of sample error.
- It is the out of sample error that must be low, or we do not have a good model.

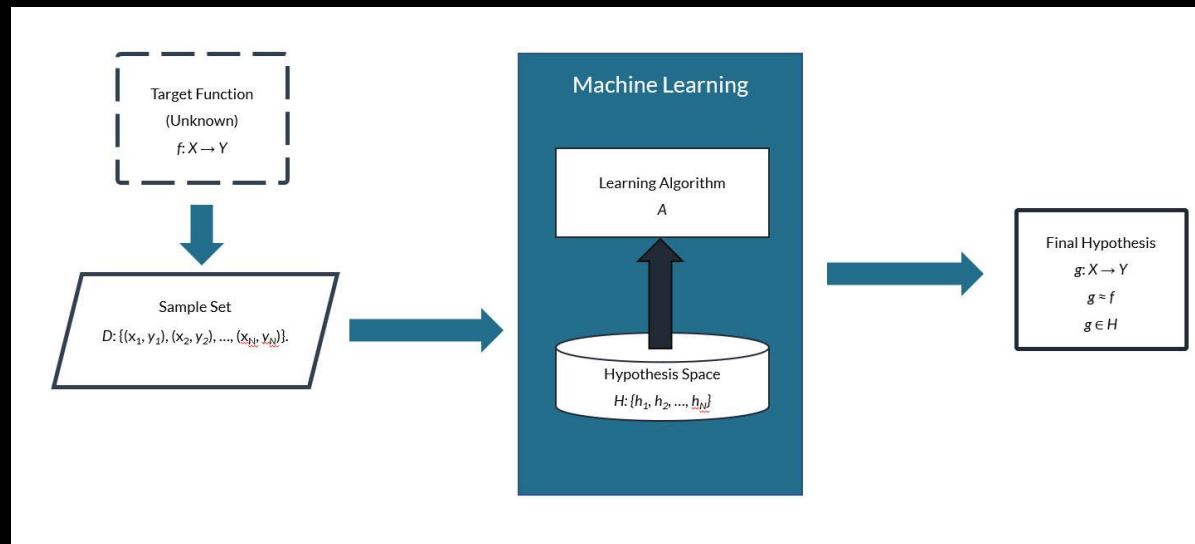
The Hypothesis Set

- From the previous slides it should be clear that we start with a set of “candidate” functions.
 - For example, all linear functions, or all quadratics, or possibly all polynomials or all continuous functions.
- Each candidate is a “machine,” and our goal is to choose one. A best performer.
- These functions are also called “hypotheses” because each is potentially the true relationship between the independent and dependent variables.
- The set of candidate functions (e.g., all linear) is therefore called the “hypothesis set” or the “hypothesis space”.



The Hypothesis Set

- The bigger the hypothesis set, the more likely it is to contain the true function, but the more likely it is to overfit the training data.
 - *Therein lies one of the fundamental tradeoffs in machine learning.*



More Terminology



The independent variables are also called “features” or “input variables”



Values of the dependent variable are also called “labels”.

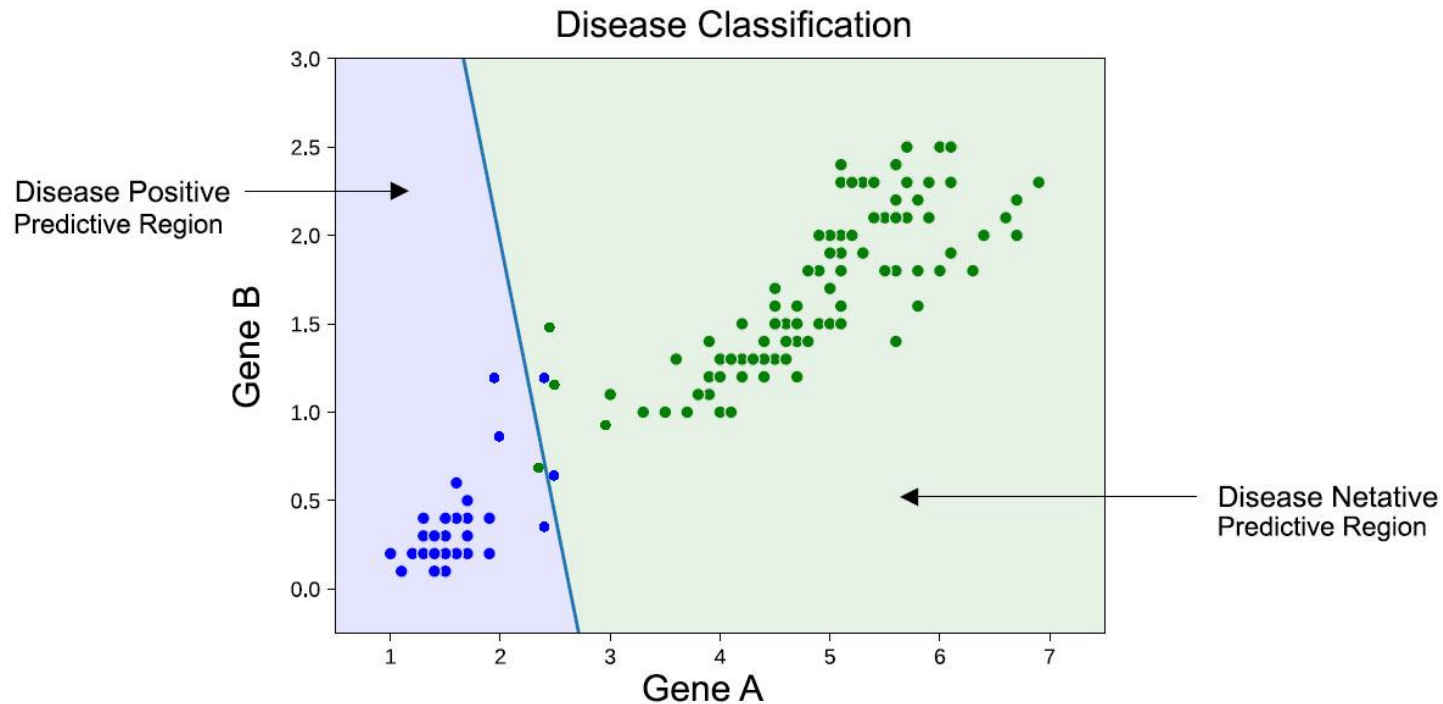


So, in this language, training data is data where the values of the features are given and where the corresponding labels are also known.

Classification

features are continuous variables (Gene A, Gene B)
labels are categorical variables (blue, green)

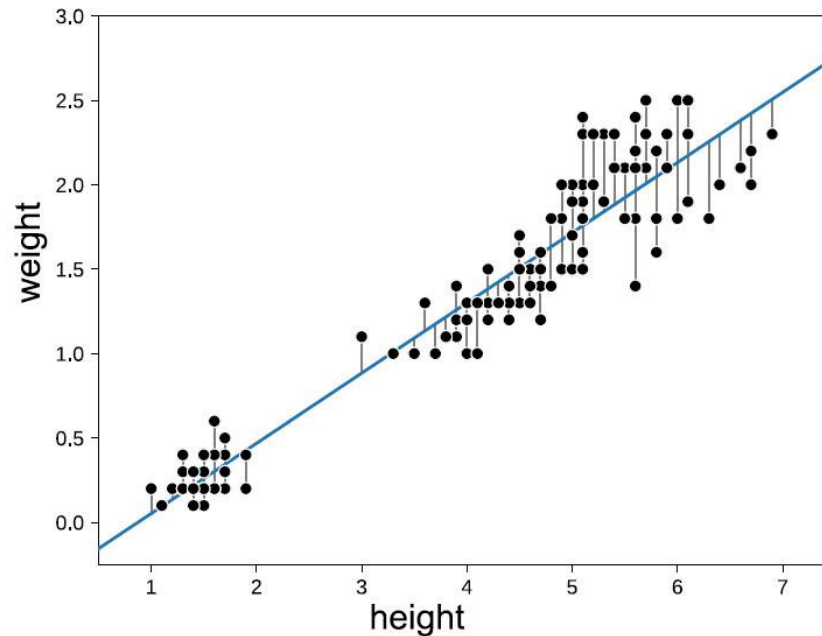
- Classification — labels are categorical



Regression

features is continuous (height)
labels also continuous (weight)

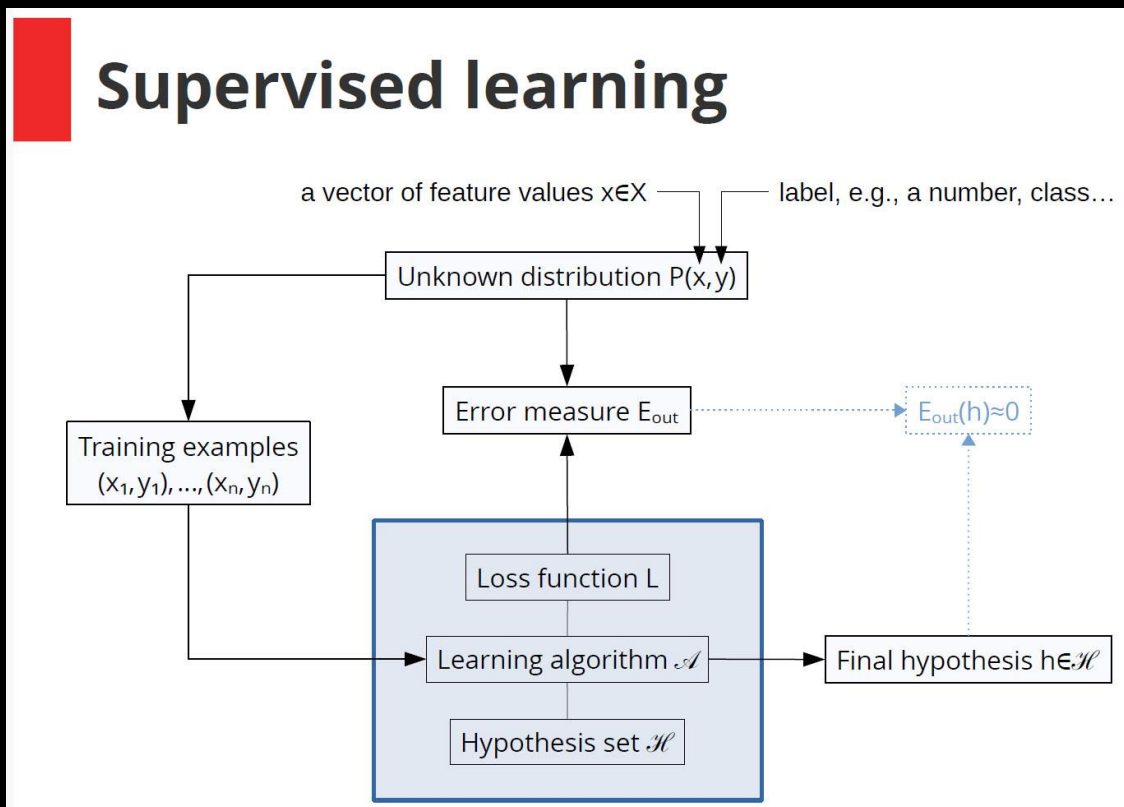
- Regression — labels are numerical



Model dependence
of weight on height

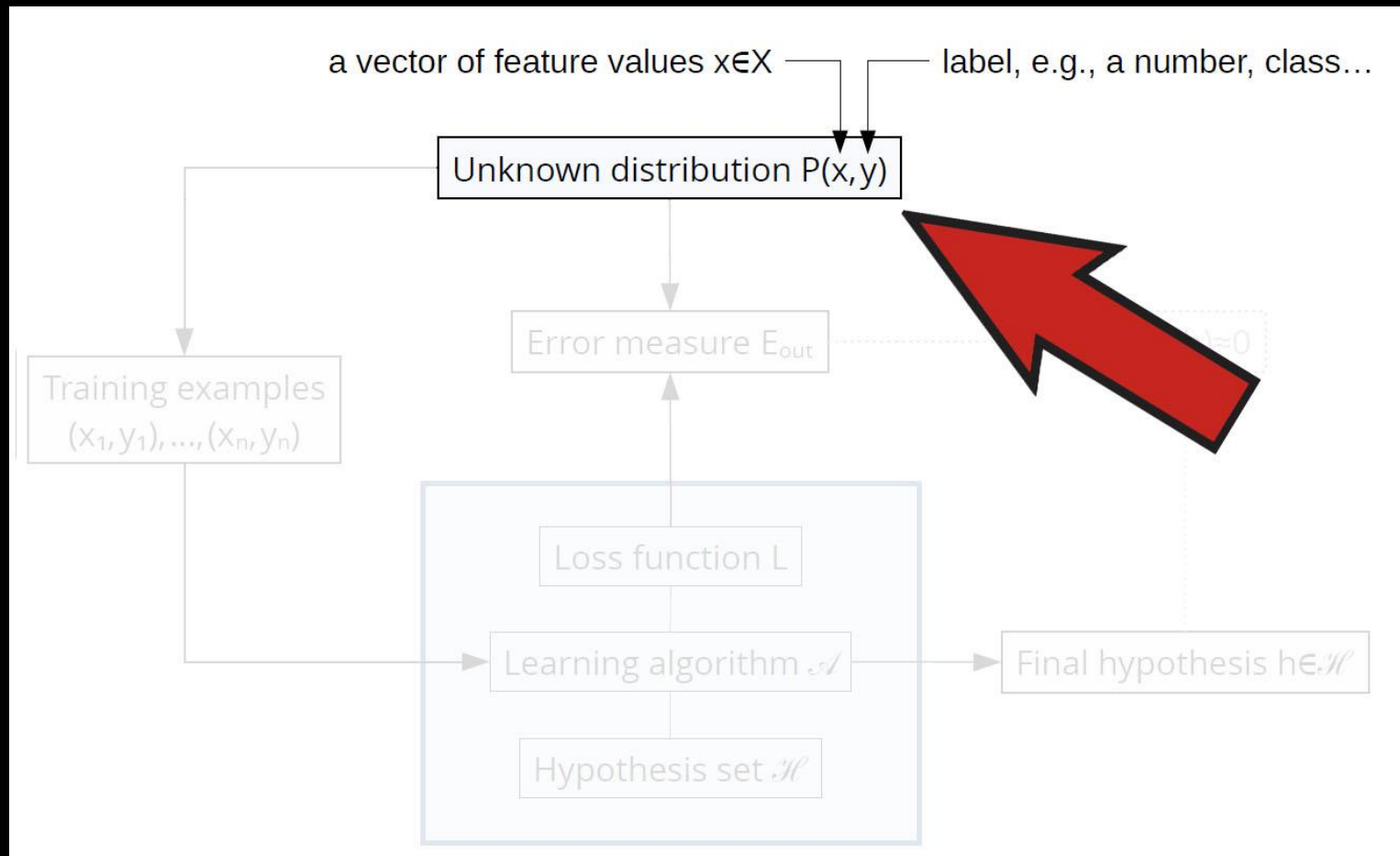
Anatomy of a Machine Learning Procedure

This diagram describes the components of a machine learning procedure.



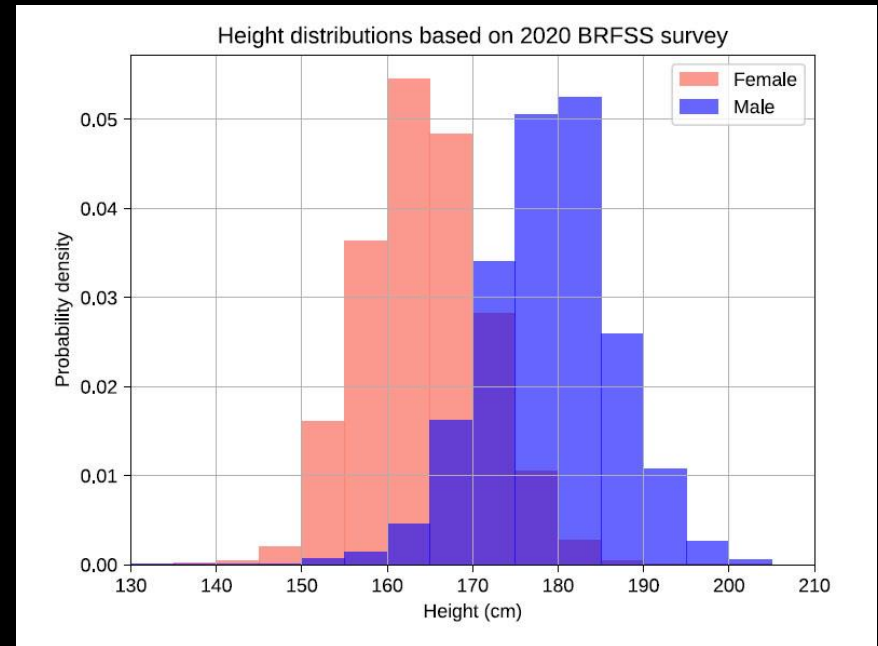
We will go through this piece-by-piece.

The Unknown Distribution



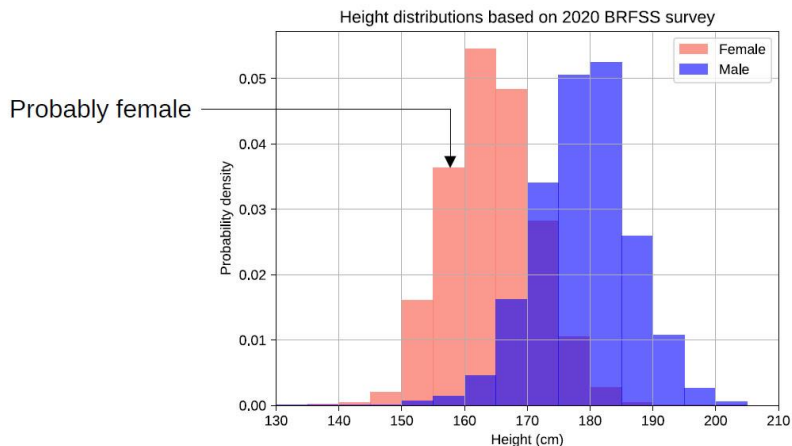
Unknown Distribution

- The relationship between the features and labels is rarely deterministic.
- For example, correctly deriving the gender of a person using only their height is not possible.
- However, there is a probability that a given person has a certain gender given their height.



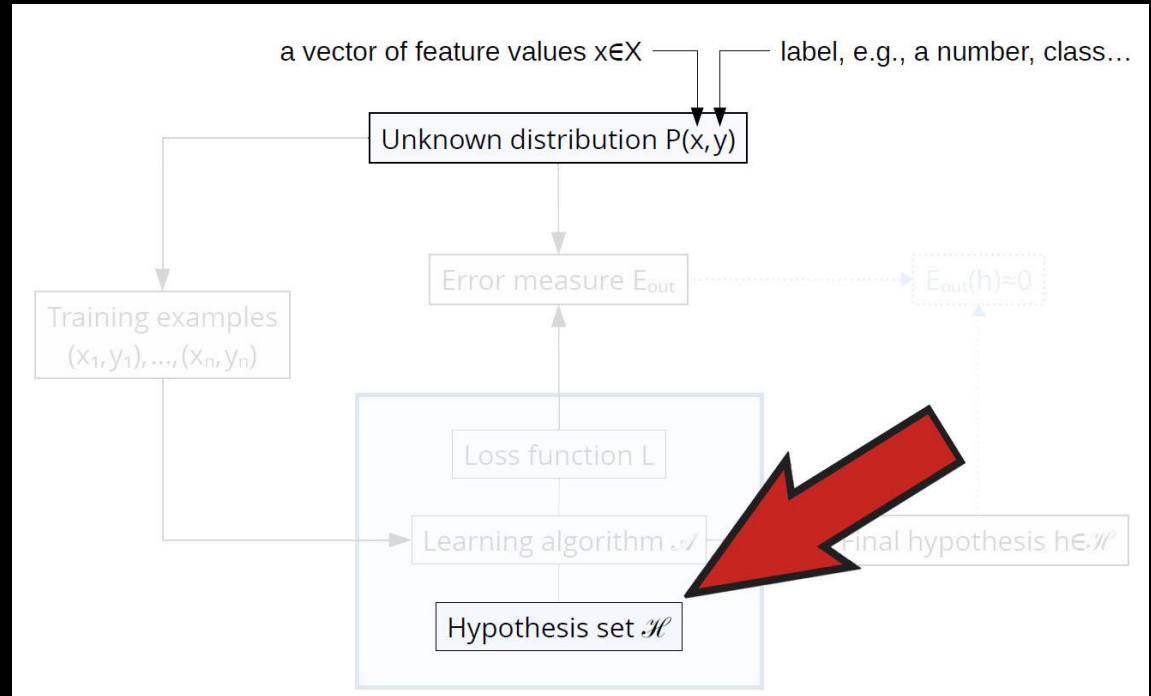
Unknown Distribution

- Distribution of heights



- If x are the feature values and y are the labels, then the unknown (joint) distribution $P(x, y)$ captures:
 - The conditional probability distribution of the feature values given the labels
 - The conditional probability distribution of the labels given feature values
- The unknown distribution is part of reality and usually, we have no control over it

The Hypothesis Set



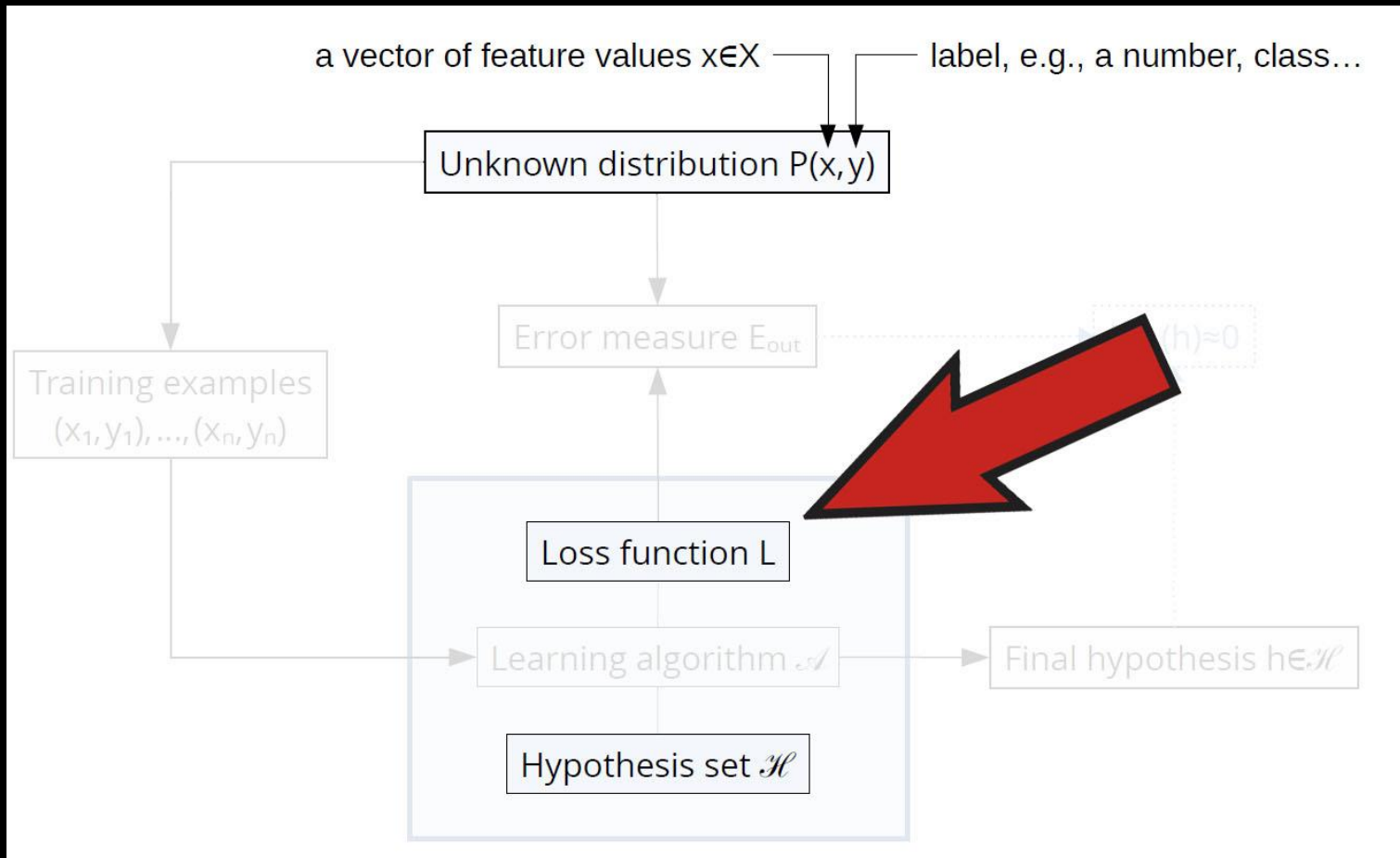
The Hypothesis Set

- We are searching for a function that assigns labels to feature values

$$f: X \rightarrow Y$$

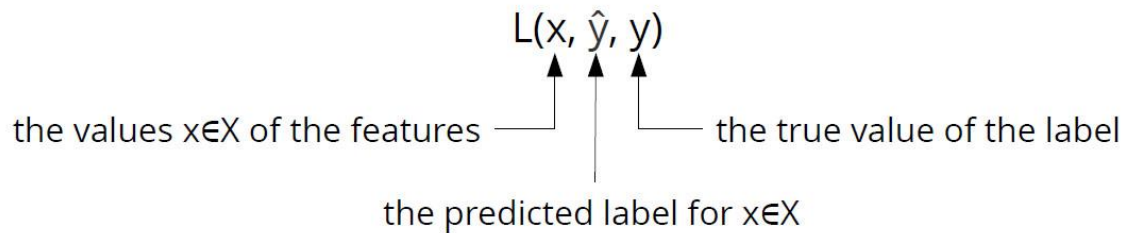
- The Hypothesis set \mathcal{H} is the set of all candidate functions which we consider in our search
 - We are generally free to choose our hypothesis set
 - For example, all linear functions, or all quadratics, etc.

The Loss Function



The Loss Function

- How can we compare two hypotheses to decide which one is better?
- What does *better* or *best* even mean?
 - For this we need an error measure to measure how well a hypothesis performs.
 - An error measure is almost always derived from a loss function
- A loss function measures how close a prediction gets to the truth for a single subject.
- A loss function takes three values as inputs:



- The loss value $L(x, \hat{y}, y)$ quantifies how far the true value of the label is from the predicted value \hat{y} for a single subject.



Examples of Loss Functions

- Examples:
 - 0-1 loss in classification:

$$L(x, \hat{y}, y) = \begin{cases} 1 & \text{if } \hat{y} \neq y \\ 0 & \text{if } \hat{y} = y \end{cases}$$

- quadratic loss in regression:

$$L(x, \hat{y}, y) = (\hat{y} - y)^2$$

The Loss Function

- The loss function is one of the choices we must make.
- The choice of a loss function affects the ranking of hypotheses and hence the outcome of the learning process.
- Loss functions are often chosen because they have nice mathematical properties such as continuity or differentiability.
- This choice can be difficult, so in many cases people stick with defaults such as 0–1 or quadratic loss.

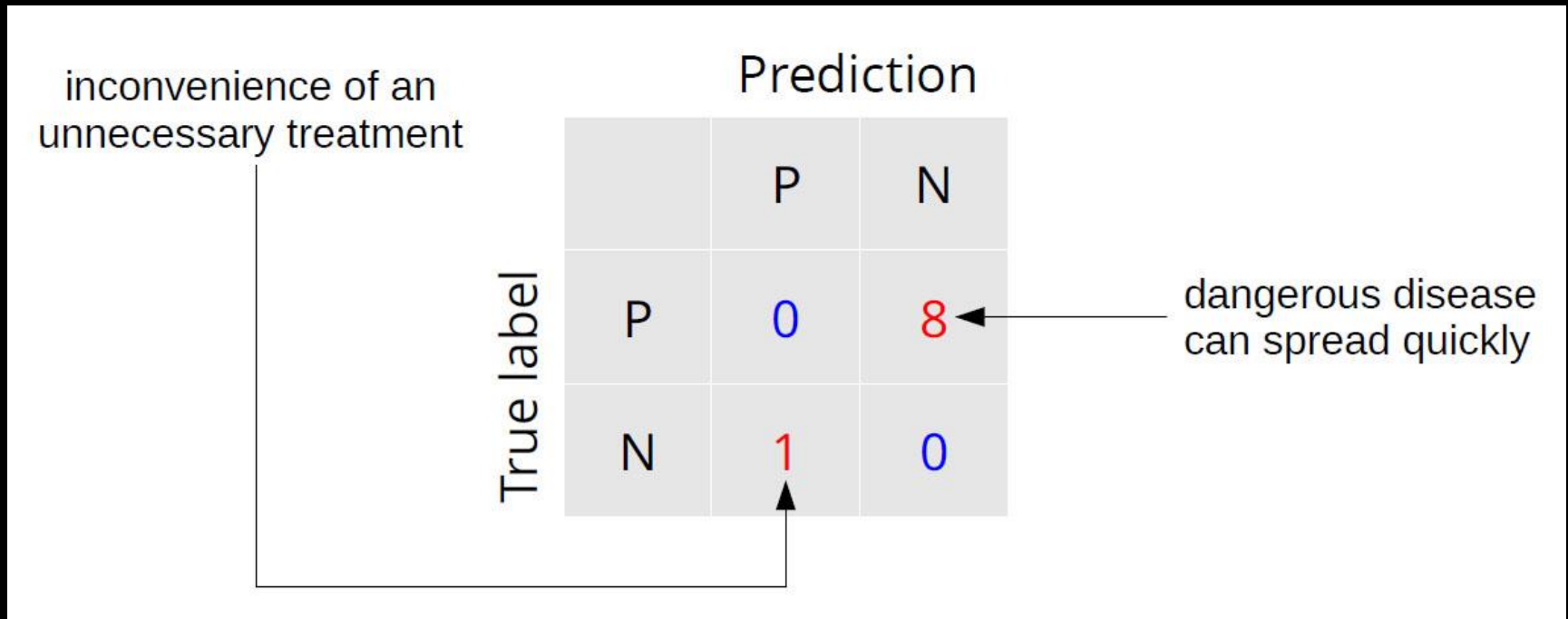


Symmetry of the Loss Function

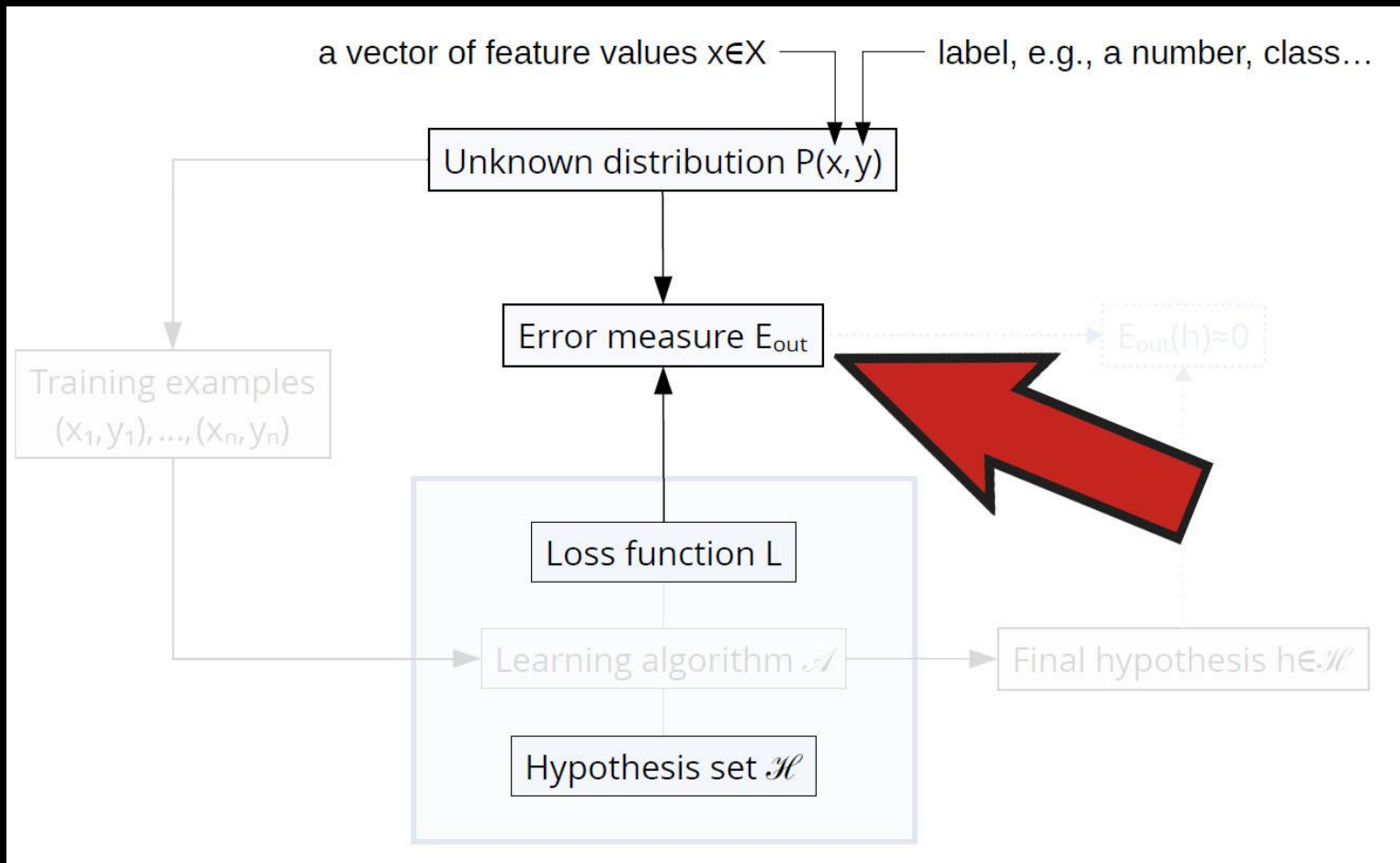
- The 0-1 loss function is symmetric in the sense that false positives are penalized by the same amount as false negatives.
- This might not be realistic, for example if we are trying to test for a dangerous highly contagious disease.
 - In this case, false negatives have much more dire consequences than false positives.

Alternative to the 0-1 Loss Function

This loss function accounts for the asymmetry between false-positives and false-negatives.



The Error Measure



The out-of-sample Error Measure

$$E_{\text{out}}(h) = \mathbb{E}_{x \in X, y \in Y} [L(x, h(x), y)]$$

↑
expectation with respect to the
unknown probability distribution $P(x, y)$

- Once the choice of the loss function has been made, we define the (out-of-sample) error measure as the expected loss, with respect to the unknown true distribution.
- It depends on the hypothesis.
- And the lower it is, the better the hypothesis.
- Therefore, it is used to compare the performance of different hypotheses, to choose a good one.
- Since the true distribution is unknown, we cannot calculate it exactly, but we estimate it using the “test” data.

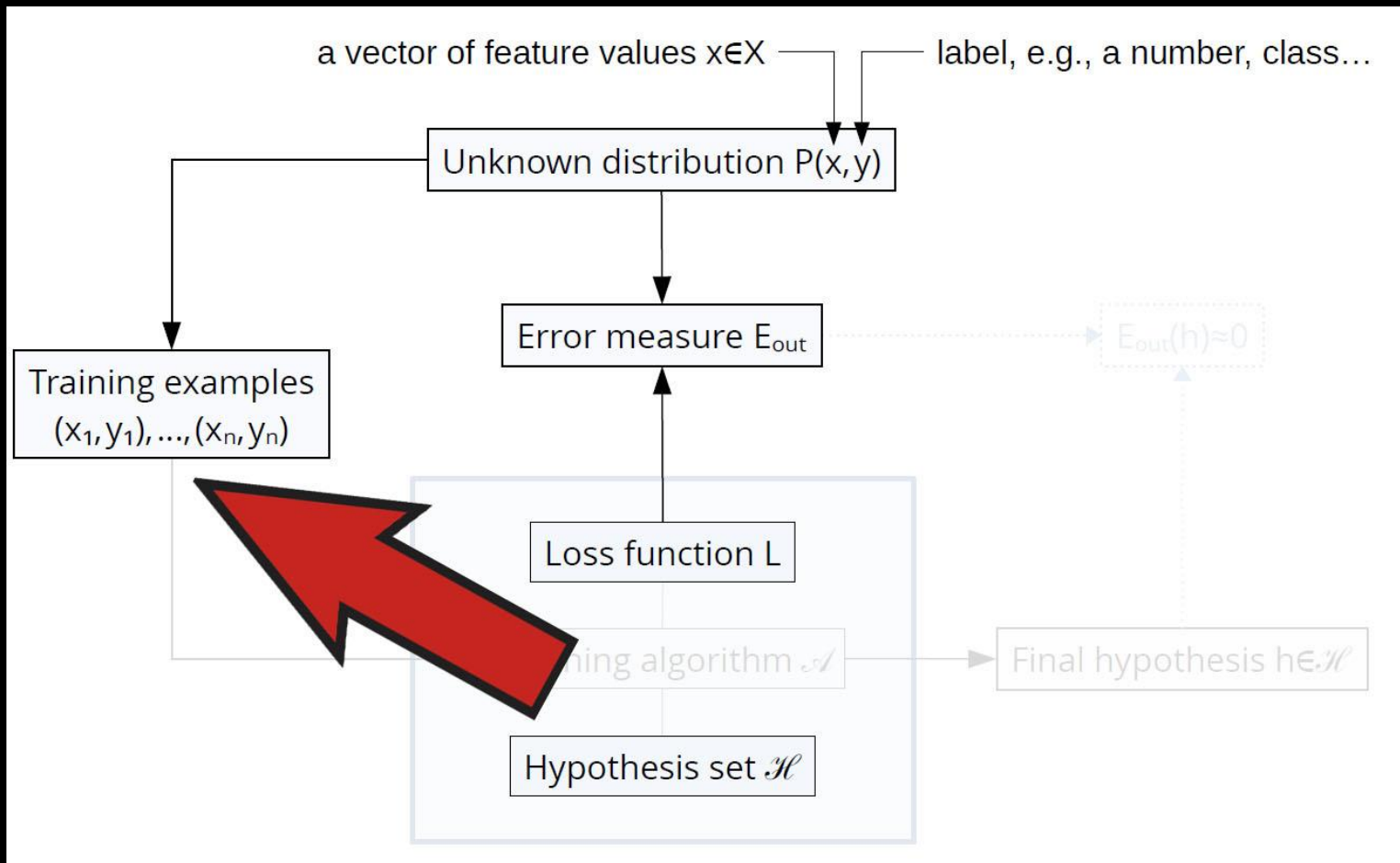
The Error Measure Example

Example of the error measure for the 0-1 loss function.

- The error measure for the 0-1 loss function is the misclassification probability of the classifier:

$$E_{\text{out}}(h) = \mathbb{E}_{x \in X, y \in Y} \left[\begin{cases} 1 & \text{if } h(x) \neq y \\ 0 & \text{if } h(x) = y \end{cases} \right] = P(h(x) \neq y)$$

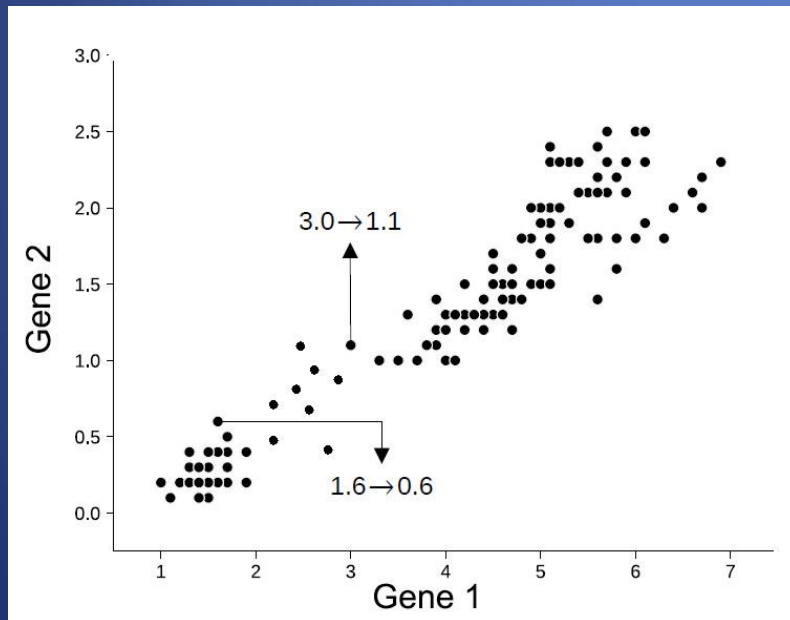
The Training Data



The Training Data

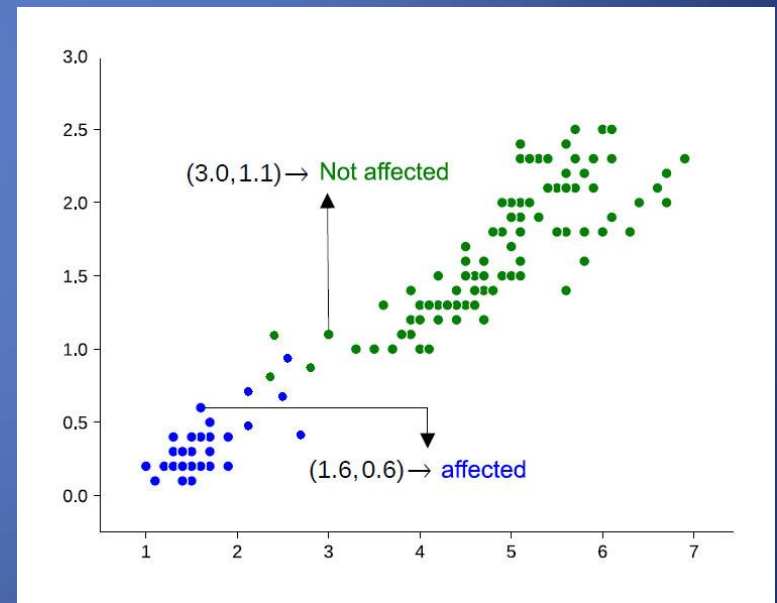
- To do supervised learning, we need data which contain input values *and the corresponding labels*.
- Like a bunch of question and answer pairs

REGRESSION



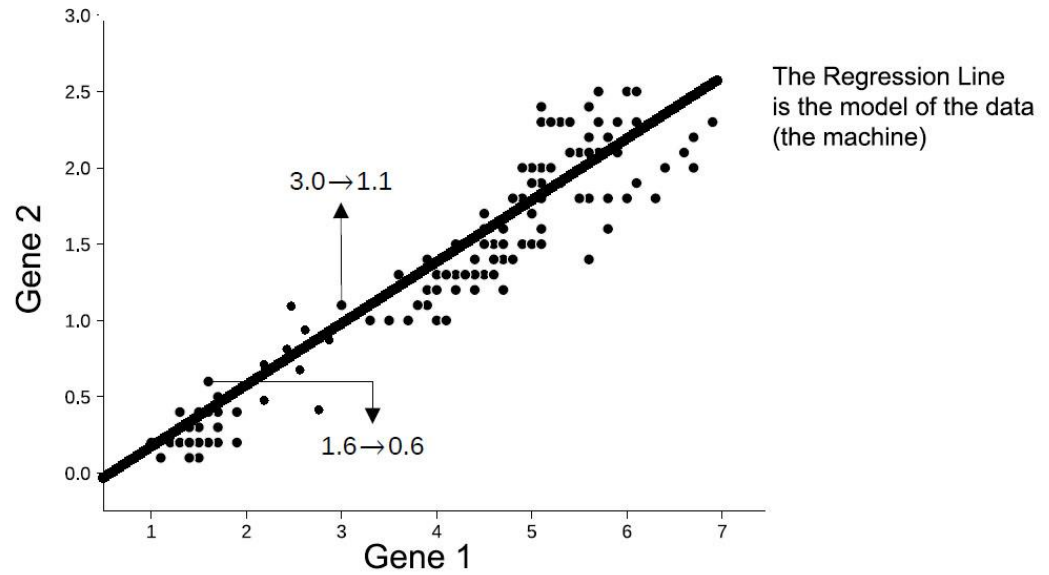
The inputs are Gene 1 and the labels are Gene 2

CLASSIFICATION

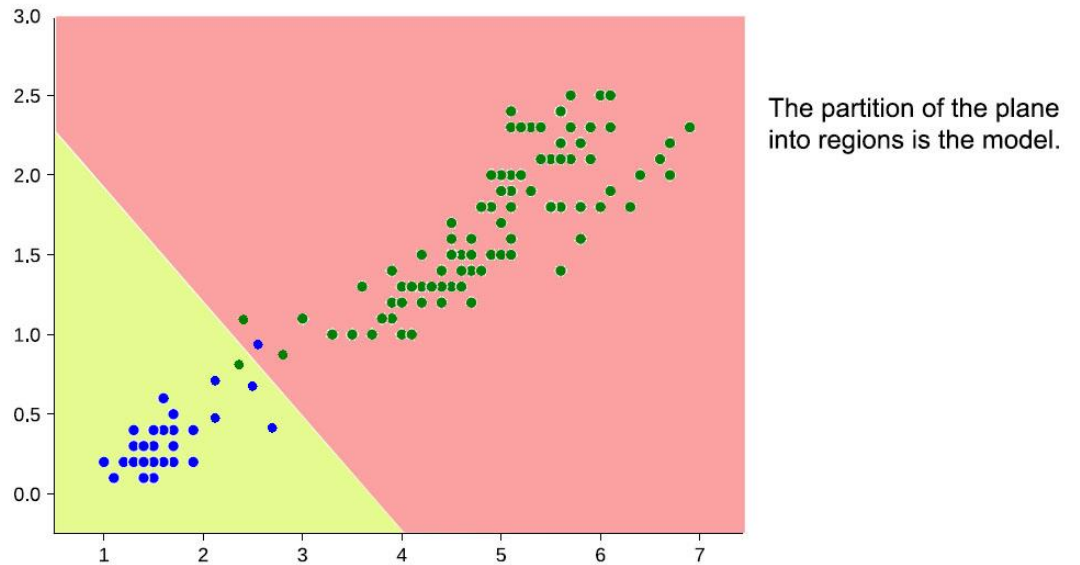


The inputs are Gene 1 and Gene 2 and the labels are the categories: affected/not affected

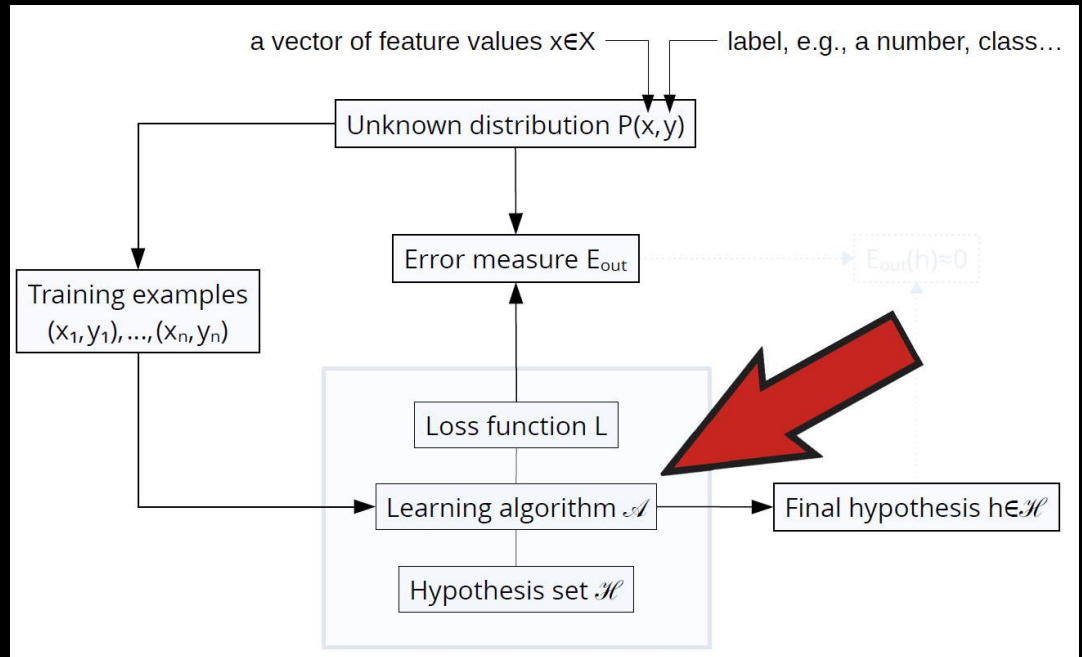
- Regression



- Classification



The Learning Algorithm



The Learning Algorithm

- $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ is training data.
 - Also called a “sample”.

- An algorithm \mathcal{A} is a procedure that takes training examples in S as input and searches the hypothesis set \mathcal{H} for a good hypothesis $h \in \mathcal{H}$ with low error



Example of an Algorithm

- least squares -

- The Least Squares procedure of fitting a line to points in simple linear regression is the learning algorithm.
- Any method used to fit the regression function to training data is a learning algorithm.
- For classification, any method that partitions the plane into two parts corresponding to the two categories is an algorithm.
- Some algorithms are better than others.



In a Nutshell

Learning model

- Learning model consists of:
 - hypothesis set \mathcal{H}
 - loss function L
 - algorithm \mathcal{A}
- These three (usually not independent) choices need to be made in order to specify a learning model

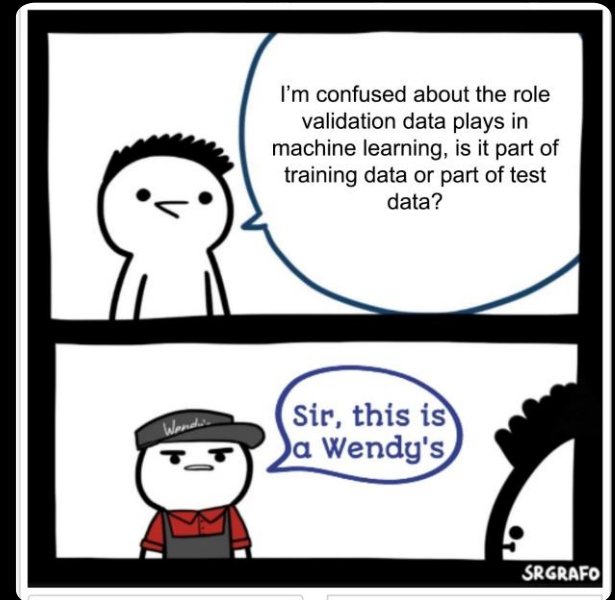
Training vs. Testing

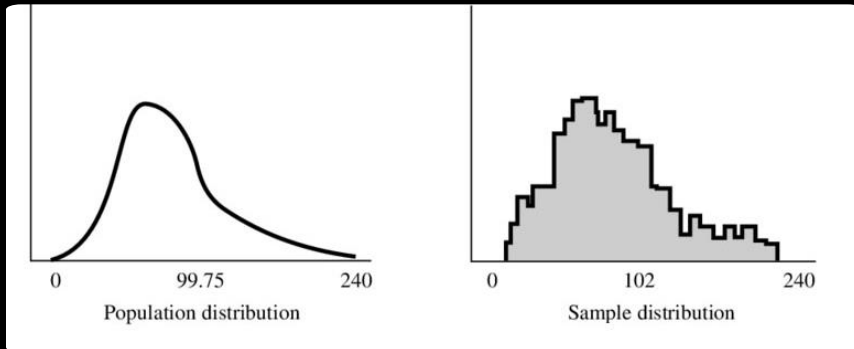
- What was described above is about training the model.
- But once trained, we need to evaluate how well it performs.
 - And perhaps modify things if performance is less than required.
- Therefore, data (where the truth is known) must be divided into at least two parts:
 - Training data – to train the model
 - Test data – to evaluate the model



Training vs. Validation vs. Testing

- If the test data indicates performance is less than desired, then the model may be modified further.
- At this point, yet more test data is required to evaluate performance, because of the following:
- **MAXIM: Final performance cannot be evaluated on data which was used to inform the model in any way.**
- Once the test data has been used to further modify the model, it is no longer test data, it has become training data.
 - Which is often rebranded as “Validation” data.
- Validation data is not data which was used for initial training, nor data which was used for final evaluation of performance.
 - It is data which is used for further refinement after initial training.



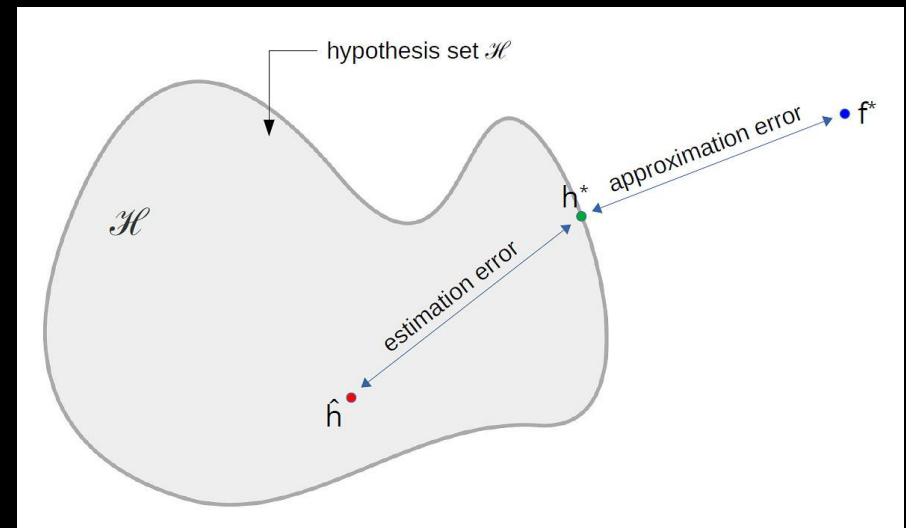


In-Sample and Out-of-Sample Error

- We now turn to matters of evaluating the performance of the model.
- With training or test data, since we know the truth, we can put a metric on how close the model gets to the truth.
 - Which could be a distance (for prediction), or a 0-1 loss (for classification).
- **IN-SAMPLE:** If we apply the error metric to training data, then we call it the “In-Sample Error”
- **OUT-OF-SAMPLE:** If we apply the error metric to the test data, then we are *approximating* the “Out-of-Sample Error” which is defined to be the true error based on the true (unknown) distributions.
 - Since the distributions are unknown, we can only approximate the true (out-of-sample) error, by using test data.

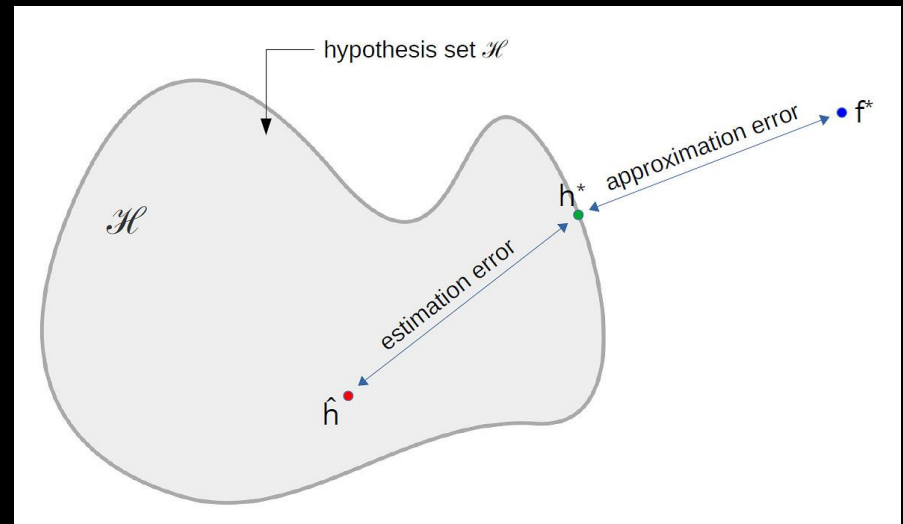
Empirical Risk Minimization

- The hypothesis set must be big enough to either contain the true function or at least one close to it.
 - But we must keep \mathcal{H} from being too big (because of overfitting).
- Here \mathcal{H} is the hypotheses set.
- f^* is the true (unknown) hypothesis by which Mother Nature has generated the data.
- h^* is the hypothesis in \mathcal{H} that is closest to f^* .
 - With \mathcal{H} given, we can't do better than h^* .
- \hat{h} is the hypothesis in \mathcal{H} found by the (machine) learning algorithm.
 - \hat{h} is not necessarily h^* , because there's no guarantee the algorithm finds the best hypothesis in \mathcal{H} .



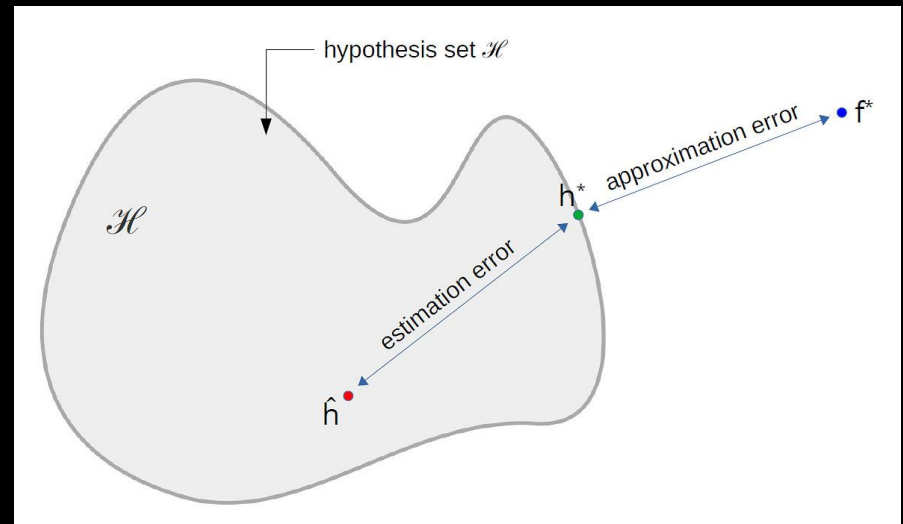
Empirical Risk Minimization

- The distance from \hat{h} to f^* is the *total error*.
- The total error decomposes into two parts:
 1. The distance from \hat{h} to h^* (called the *estimation error*)
 2. And the distance from h^* to f^* called the *approximation error*.



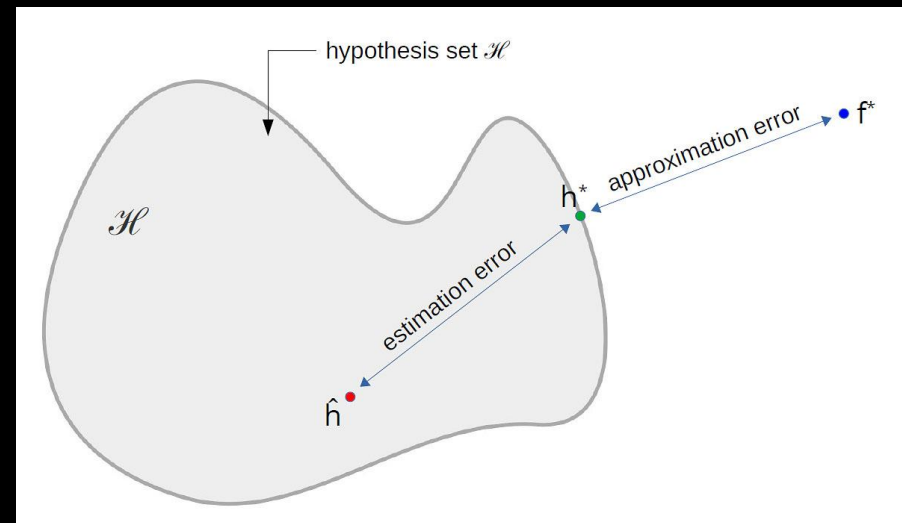
Trade-Off

- We can get the approximation error down to zero by expanding the hypothesis set.
- At the extreme it could contain all possible functions.
- That would guarantee f^* is in the set and therefore $f^* = h^*$
- But the estimation error increases because of overfitting.
 - In other words, we can get the in-sample error low, while the out-of-sample error remains high.



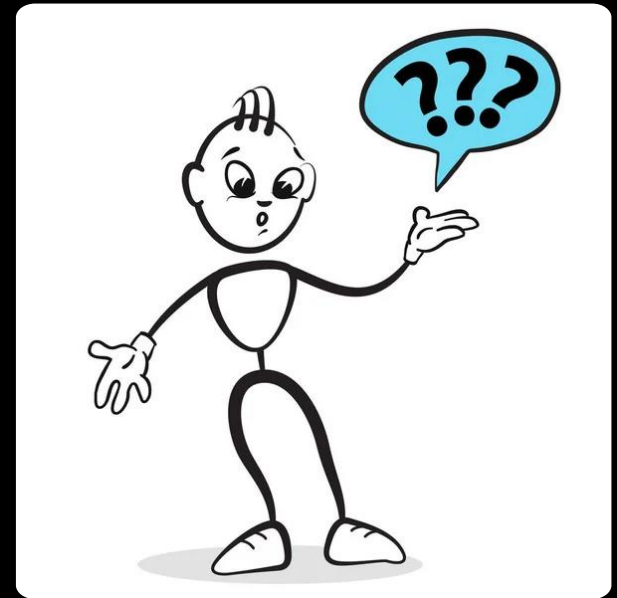
Bayes Risk

- Even if we could find f^* , using it for classification only reduces the error to be as low as possible.
- But it cannot lower it to zero, unless there happens to be a deterministic relationship between the independent and dependent variables.
- For example, height can only predict weight so well.
- In other words, $E_{\text{out}}(f^*) > 0$.
- $E_{\text{out}}(f^*)$ is the inherent error that cannot be removed.
- It is called the “*Bayes risk*” and is denoted E_{out}^* .



The True (Total) Error

- The true total error of a model is composed of three parts.
 1. The Bayes risk, which is the minimal possible error.
 2. The approximation error, which comes from \mathcal{H} being too small to contain the true function f^* .
 3. The estimation error, which comes from the learning algorithm not finding the best possible function in \mathcal{H} .



The Learning Inequality – first thing to notice

The diagram shows the Learning Inequality: $P(|E_{in}(h) - E_{out}(h)| > \epsilon) \leq 2|\mathcal{H}| \cdot e^{-2n\epsilon^2}$. The entire inequality is enclosed in a blue rectangular box. An arrow points from the text 'probability that $E_{out}(h) \neq E_{in}(h)$ ' to the left side of the inequality. Another arrow points from the text 'tries to make the probability small' to the right side of the inequality.

probability that $E_{out}(h) \neq E_{in}(h)$

$$P(|E_{in}(h) - E_{out}(h)| > \epsilon) \leq 2|\mathcal{H}| \cdot e^{-2n\epsilon^2}$$

tries to make the probability small

- This inequality is explained on the next slides.
- But the first thing to notice is the right-hand side has $|\mathcal{H}|$, which is the size of the hypothesis set.
- For the examples we've been looking at, $|\mathcal{H}| = \infty$ (there are infinitely many lines in the plane, for example) and this makes the inequality uninteresting.
 - Everything is less or equal to infinity.
- There is a version of the Learning Inequality that works for infinite hypothesis set, but it involves some mathematical technicalities.
- We're going to work with this finite version to avoid that math, because the main concept is the same, a trade-off between model complexity and size of the training data set.

The Learning Inequality – second thing to notice

The diagram shows the Learning Inequality: $P(|E_{in}(h) - E_{out}(h)| > \epsilon) \leq 2|\mathcal{H}| \cdot e^{-2n\epsilon^2}$. The entire equation is enclosed in a blue rectangular box. An arrow points from the text 'probability that $E_{out}(h) \neq E_{in}(h)$ ' to the left side of the equation. Another arrow points from the text 'tries to make the probability small' to the right side of the equation.

probability that $E_{out}(h) \neq E_{in}(h)$

tries to make the probability small

$$P(|E_{in}(h) - E_{out}(h)| > \epsilon) \leq 2|\mathcal{H}| \cdot e^{-2n\epsilon^2}$$

- The second thing to realize is that $E_{in}(h)$ is a random variable.
 - The in-sample error is a function of the training data, and training data is a random sample of data from a population.
- $E_{out}(h)$ on the other hand is not random, it's the true error from using h which is an expected value with respect to the true (unknown) distributions. So $E_{out}(h)$ is just a fixed number.
- Because $E_{in}(h)$ is random, that's why the left-hand side of the Learning Inequality is a probability. It's a bound on a probability. Read it like this:
 - The probability that $|E_{in}(h) - E_{out}(h)|$ is larger than ϵ is less than $2|\mathcal{H}|e^{-2n\epsilon^2}$

The Learning Inequality - explained

The diagram shows the Learning Inequality equation: $P(|E_{in}(h) - E_{out}(h)| > \epsilon) \leq 2|\mathcal{H}| \cdot e^{-2n\epsilon^2}$. The equation is enclosed in a blue rectangular box. An arrow points from the text 'probability that $E_{out}(h) \neq E_{in}(h)$ ' to the left side of the equation. Another arrow points from the text 'tries to make the probability small' to the right side of the equation.

probability that $E_{out}(h) \neq E_{in}(h)$

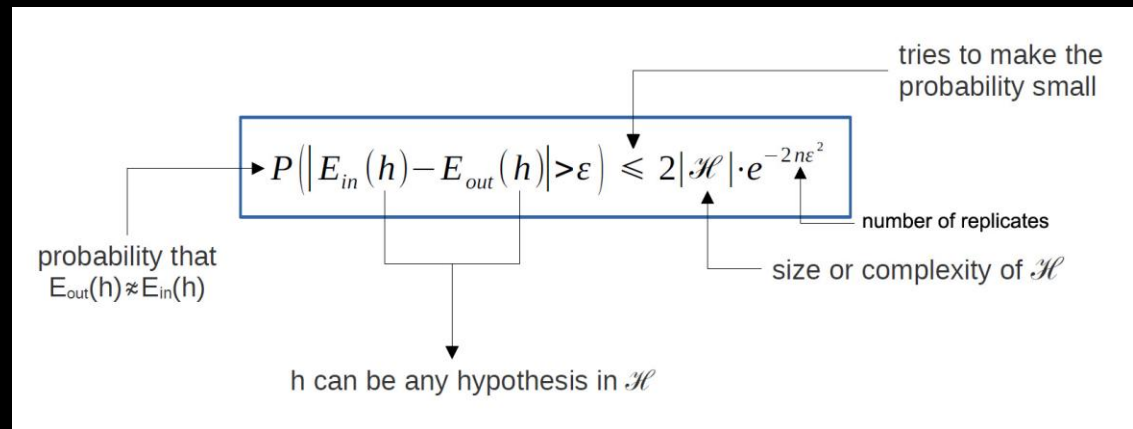
$$P(|E_{in}(h) - E_{out}(h)| > \epsilon) \leq 2|\mathcal{H}| \cdot e^{-2n\epsilon^2}$$

tries to make the probability small

- We're searching for $h \in \mathcal{H}$ which has $E_{out}(h)$ as close as possible to the best possible $E_{out}(f^*)$.
 - $E_{out}(f^*)$ is what we're also calling "the Bayes Risk", the inherent minimal possible error.
 - Once we find such an h , we denote it \hat{h} .
- The main things that causes $E_{out}(h)$ and $E_{out}(f^*)$ to be different is:
- \mathcal{H} is too small – so large approximation error (model too simple).
- \mathcal{H} is too big – overfitting (model too complex).
 - Overfitting causes $E_{out}(h)$ to be larger than $E_{in}(h)$.
 - So, we want $|E_{in}(h) - E_{out}(h)|$ to be small.
- To deal with this, we have a powerful theorem, called "The Learning Inequality" which puts a bound on $|E_{in}(h) - E_{out}(h)|$.

The Learning Inequality - interpretation

- If the left-hand side is small, then overfitting is under control.
- And this inequality tells us we can get the left-hand side small by making the right-hand side small.
- And the right-hand side depends on only two things:



1. The right-hand side gets smaller as n gets bigger.
 - More replicates, less chance of overfitting.
2. And it gets bigger as \mathcal{H} gets bigger.
 - Bigger \mathcal{H} , more chance of overfitting

Empirical Risk Minimization

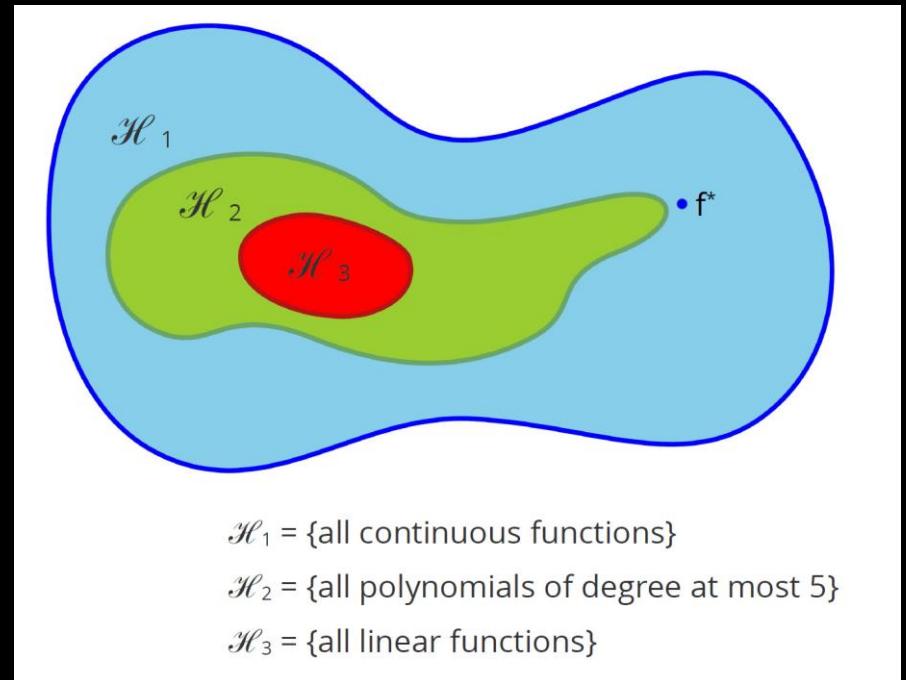
- in a nutshell -

Empirical risk minimization

- What does this tell us about the kind of \mathcal{H} we should be looking for?
 - \mathcal{H} should be “small” enough to ensure that $E_{\text{in}}(\hat{h}) \approx E_{\text{out}}(\hat{h})$, which lowers the estimation error
 - \mathcal{H} should be “large” enough to include good hypotheses so that the approximation error is low

Empirical Risk Minimization

- \mathcal{H}_3 is too small.
 - Large approximation error.
- \mathcal{H}_1 is too big.
 - Danger of overfitting, large estimation error.
- \mathcal{H}_2 is just right.

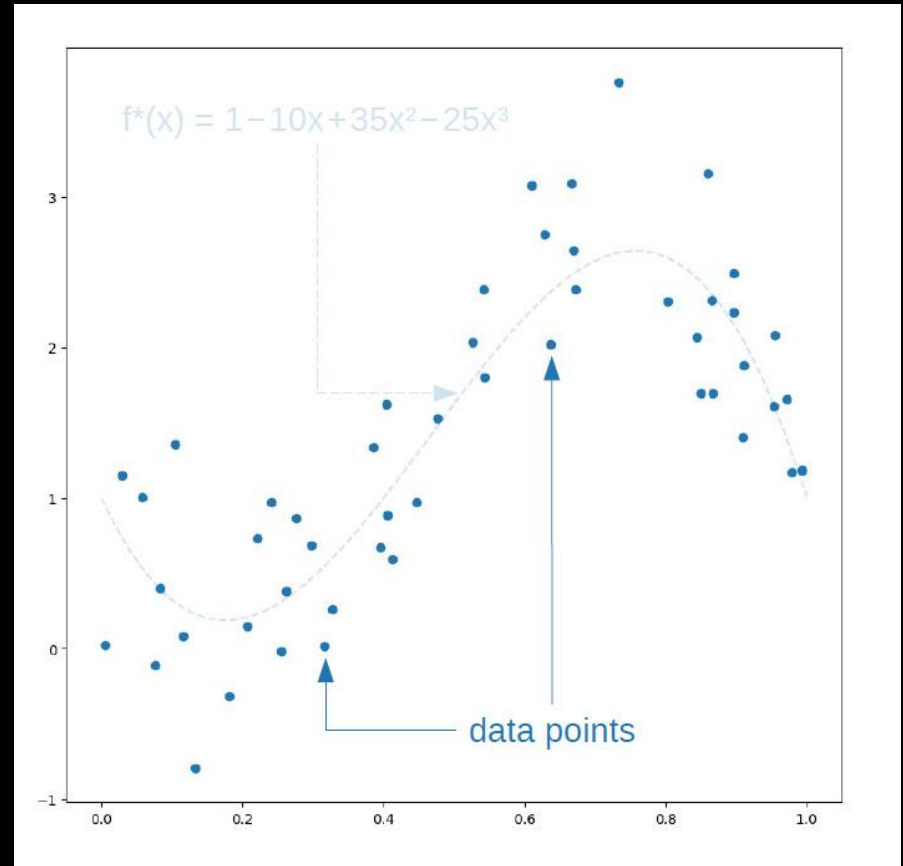


Empirical Risk Minimization Example

- Supposed the true model of some data is given by the following cubic.

$$y = 1 - 10x + 35x^2 - 25x^3 + \mathcal{E}$$

- x is the independent variable
- y is the dependent variable.
- \mathcal{E} is the non-deterministic part.
 - The Bayes risk of the system.



The True Model is the Optimal Predictor

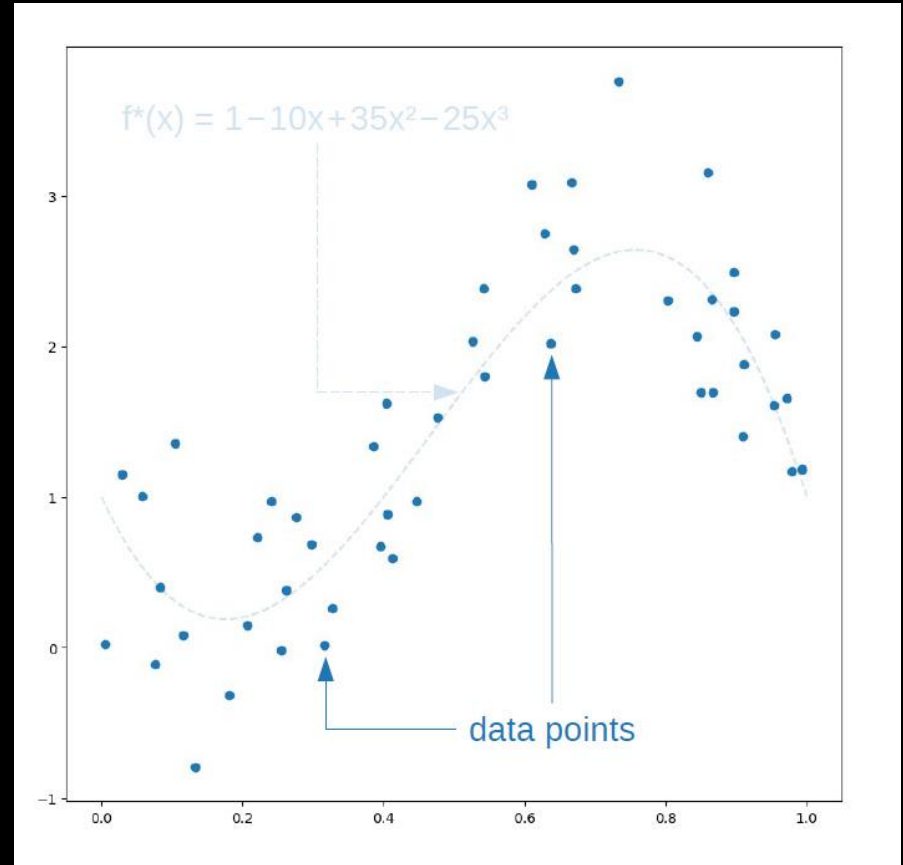
- This is the regression model that generated the data:

$$y = 1 - 10x + 35x^2 - 25x^3 + \varepsilon$$

- The optimal predictor is therefore:

$$f^*(x) = 1 - 10x + 35x^2 - 25x^3$$

- This is (usually) known only to God. We're specifying it here for illustrative purposes.
- The job of machine learning is to recover f^* (or as close as possible) from the data itself.



The Hypothesis Set

- We'll consider different hypothesis sets, to see how some are too small, some too big, and some are just right in the sweet spot.
- Let \mathcal{H}_p be the set of polynomials of degree p .

$$\mathcal{H}_0 \subseteq \mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \mathcal{H}_3 \subseteq \dots$$

- \mathcal{H}_0 are horizontal lines.
- \mathcal{H}_1 are straight lines.
- \mathcal{H}_2 are parabolas.
- Etc.

The best possible predictors from each class

Since we're playing God in this example, we can calculate exactly what h^* is.

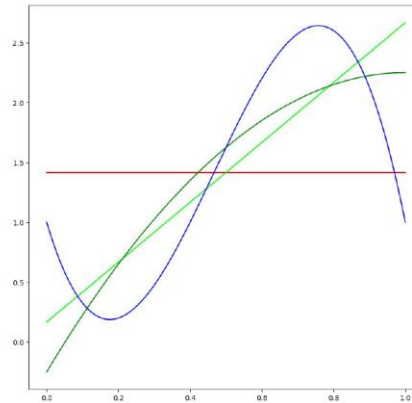
Recall h^* is the best predictor in \mathcal{H} .

We're not using a learning algorithm to find h^* .

The learning algorithm will find \hat{h} which it hopes is close to h^* .

h^* is shown in the figure for \mathcal{H}_0 through \mathcal{H}_3 with respect to the quadratic loss function.

- We can calculate the best possible predictors from each class:



$$h_0^*(x) = \frac{17}{12}$$

$$h_1^*(x) = \frac{1}{6} + \frac{5}{2}x$$

$$h_2^*(x) = -\frac{1}{4} + 5x - \frac{5}{2}x^2$$

$$h_3^*(x) = 1 - 10x + 35x^2 - 25x^3$$

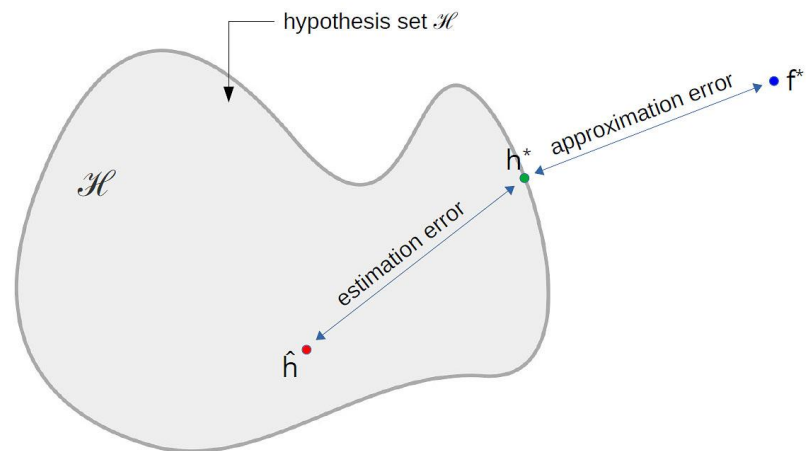
Minimal Risk

Next, for each \mathcal{H}_p we find the function \hat{h} that minimizes the error (the expected loss), with respect to the quadratic loss function.

In other words, \hat{h} is the function that minimizes the in-sample error.

In other words, \hat{h} is the function that minimizes the error with respect to the training data (and the quadratic loss function).

Keep in mind, as p grows, overfitting gets worse and consequently, the distance from \hat{h} to h^* increases.



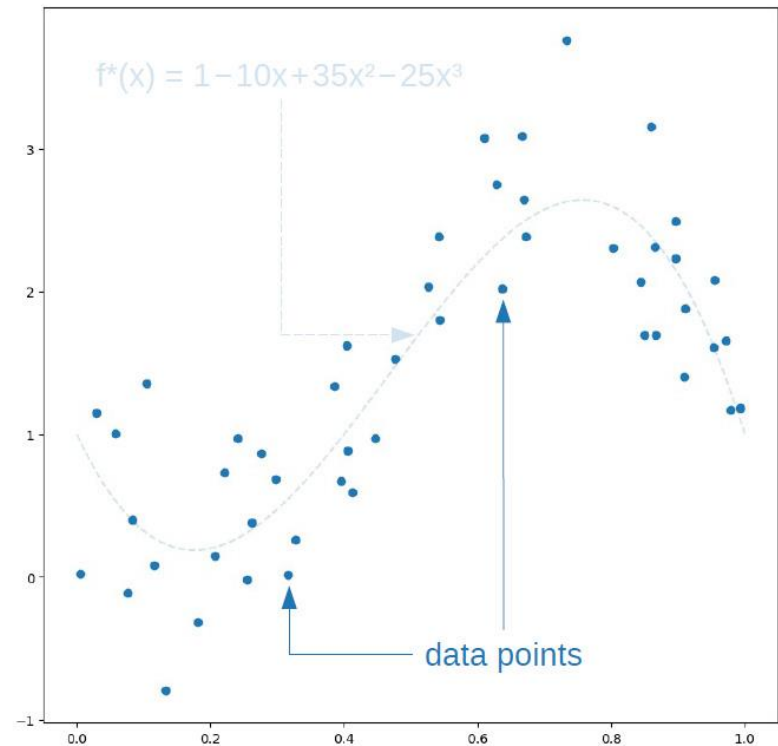
The Data and the True Model

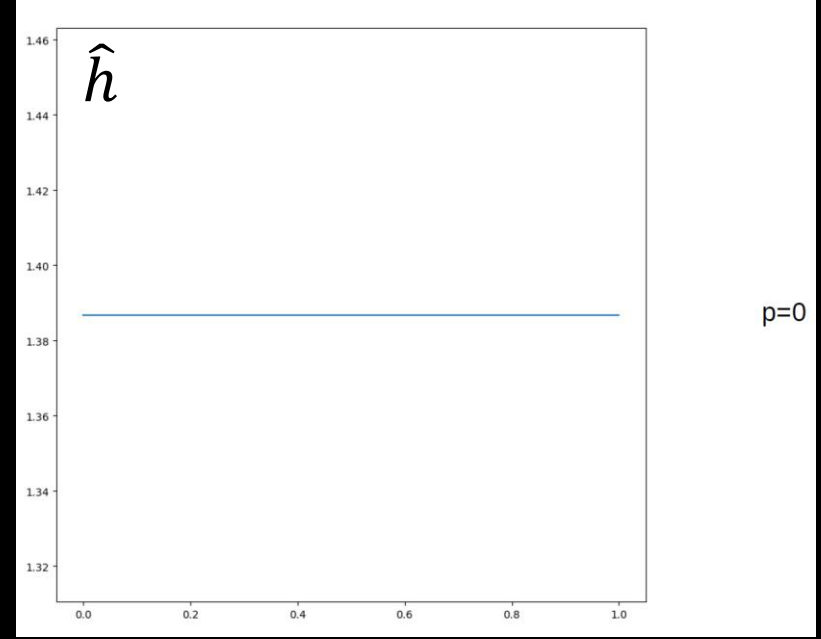
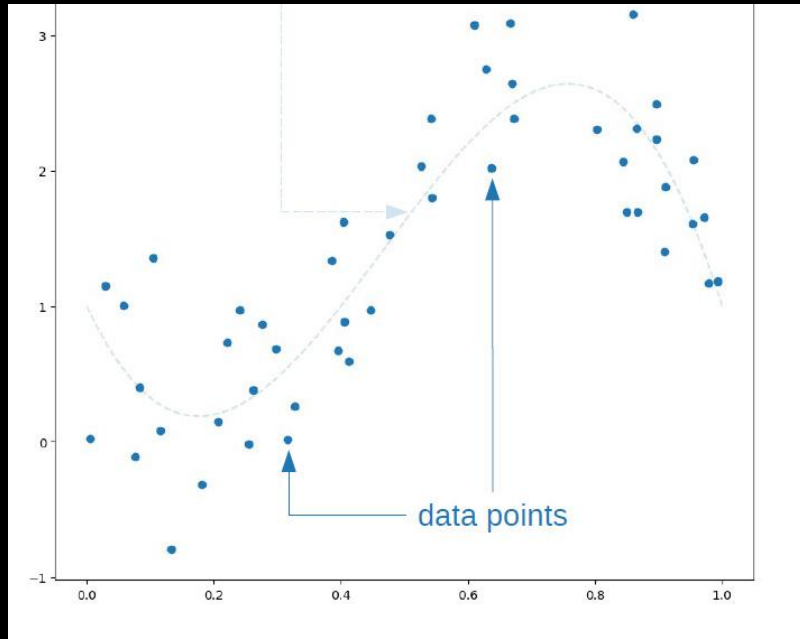
These points represent our subjects.

This is the training data.

The real model is a cubic and we're going to fit a 0-degree polynomial to it, and then a 1st degree, then a 2nd, etc.

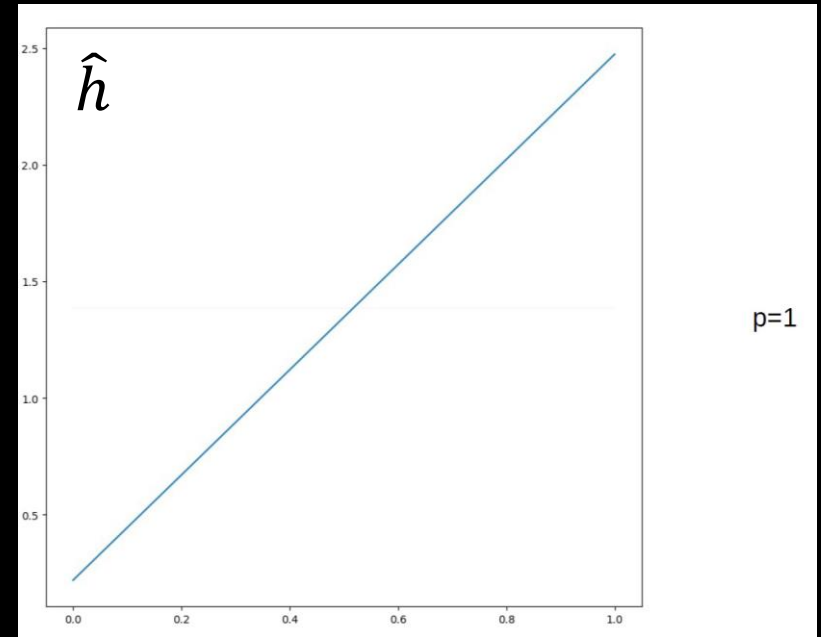
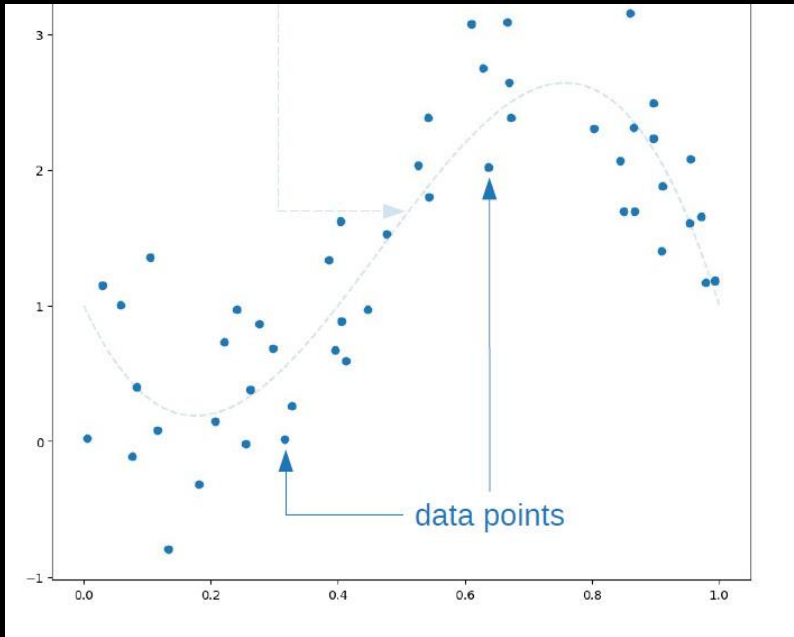
Including degrees that are higher than the true model: 4th, 5th, 6th, etc. to see how overfitting gets worse as \mathcal{H}_p grows.





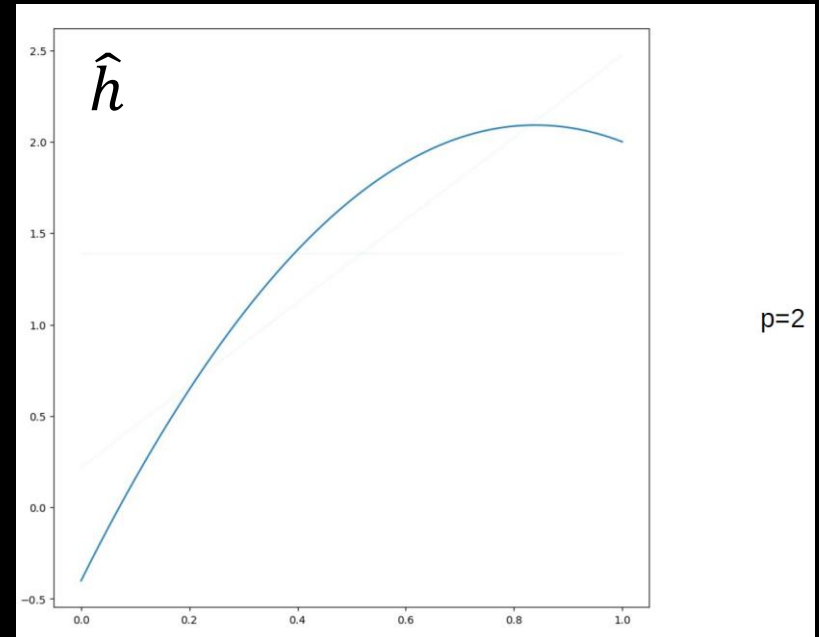
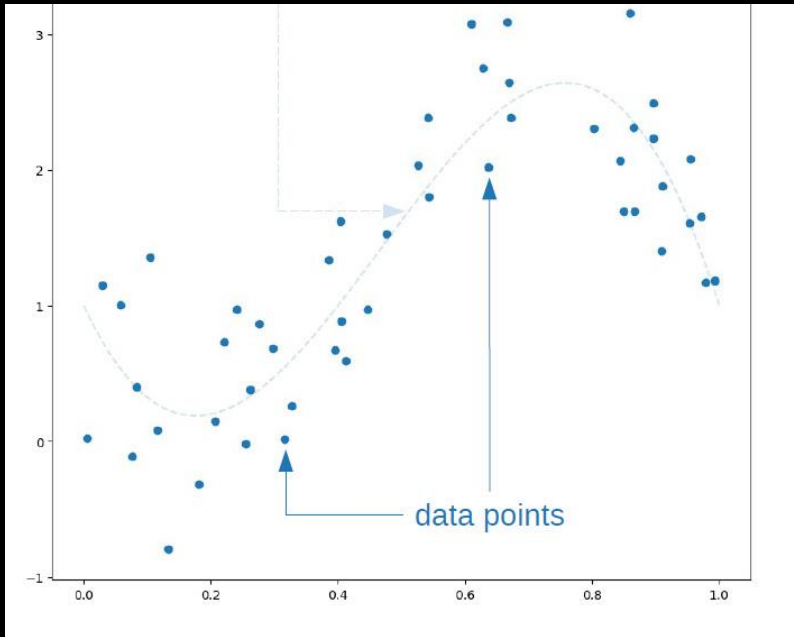
$$\mathcal{H}_0$$

The best horizontal line that fits the training data



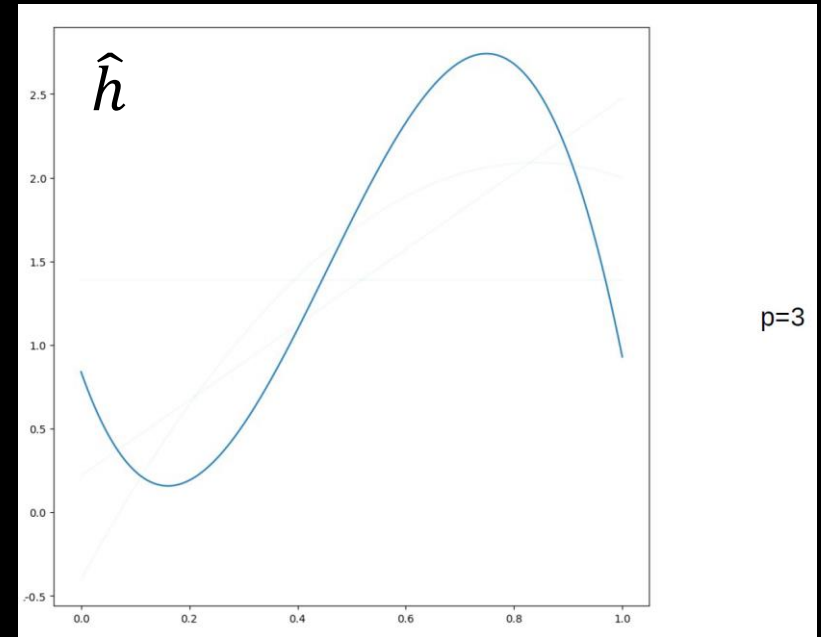
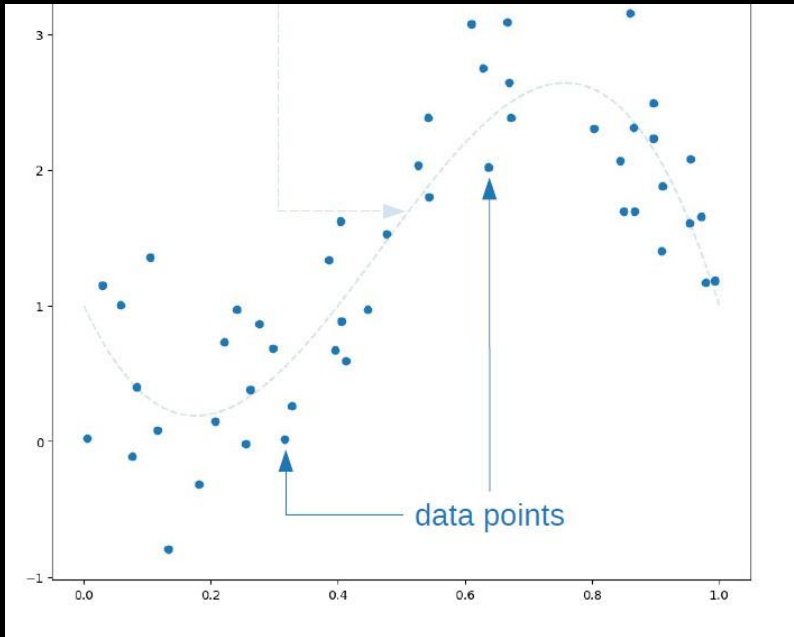
$$\mathcal{H}_1$$

The best straight line that fits the training data



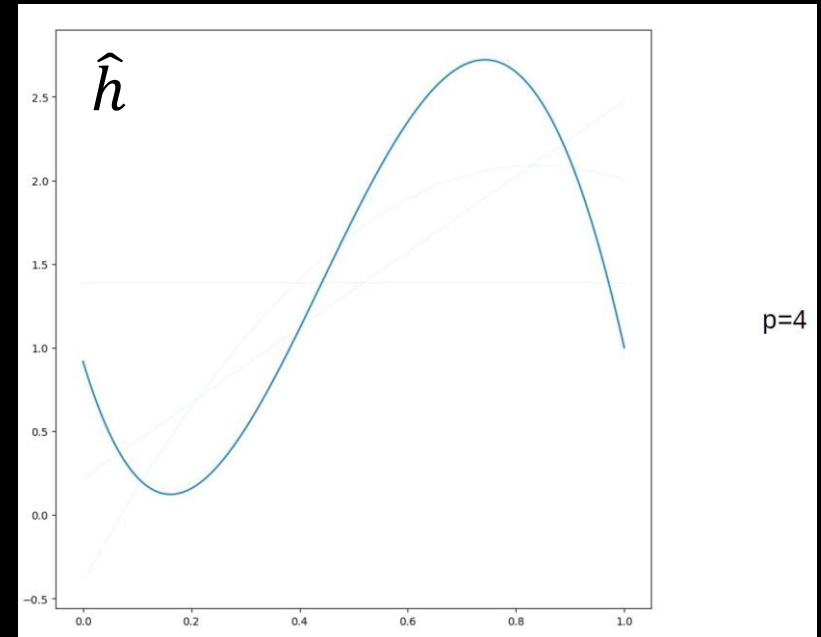
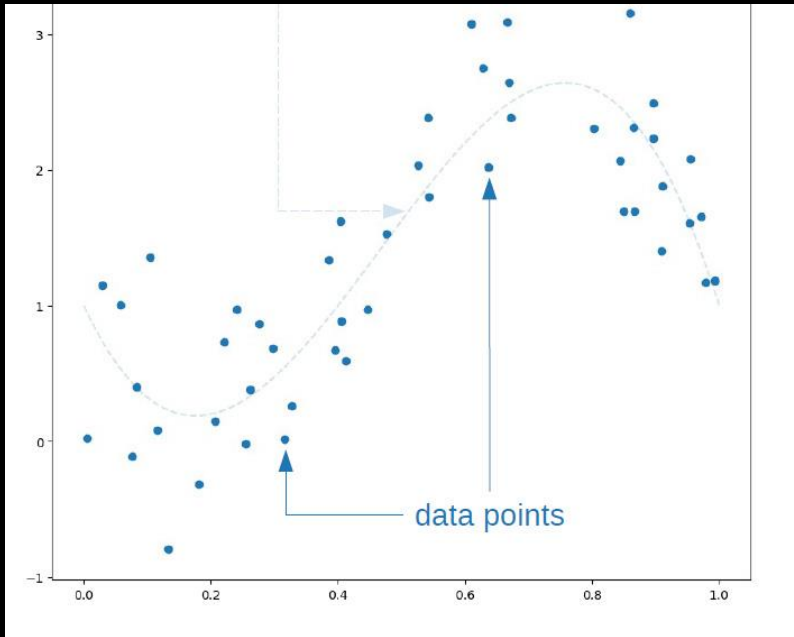
$$\mathcal{H}_2$$

The best parabola that fits the training data



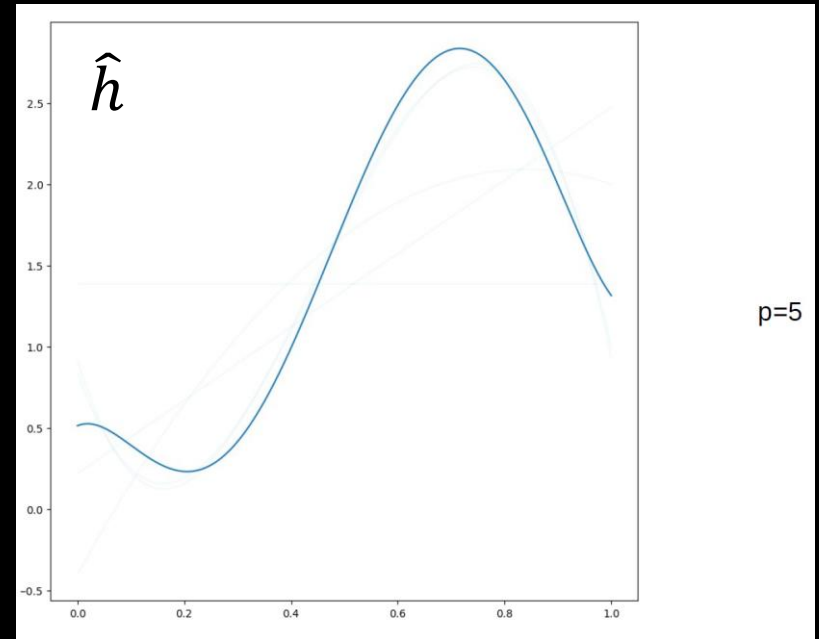
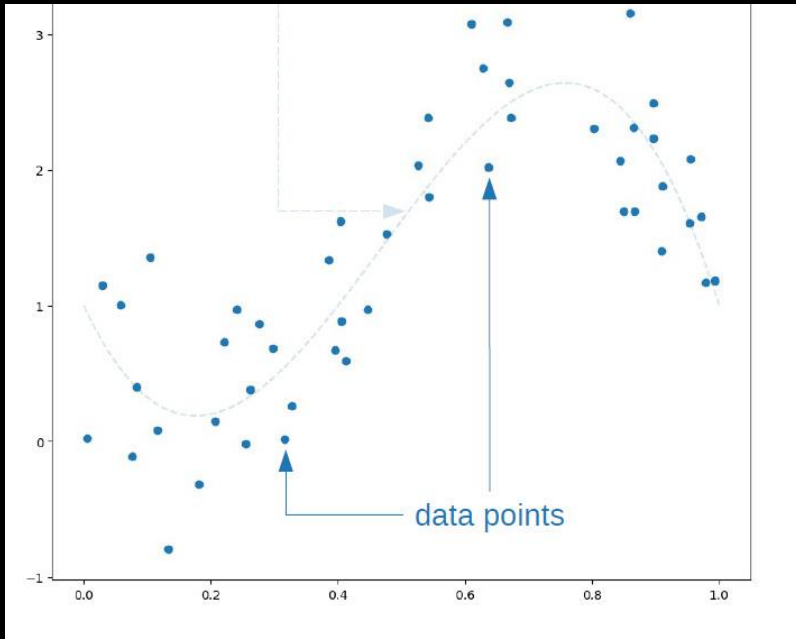
\mathcal{H}_3

The best cubic that fits the training data



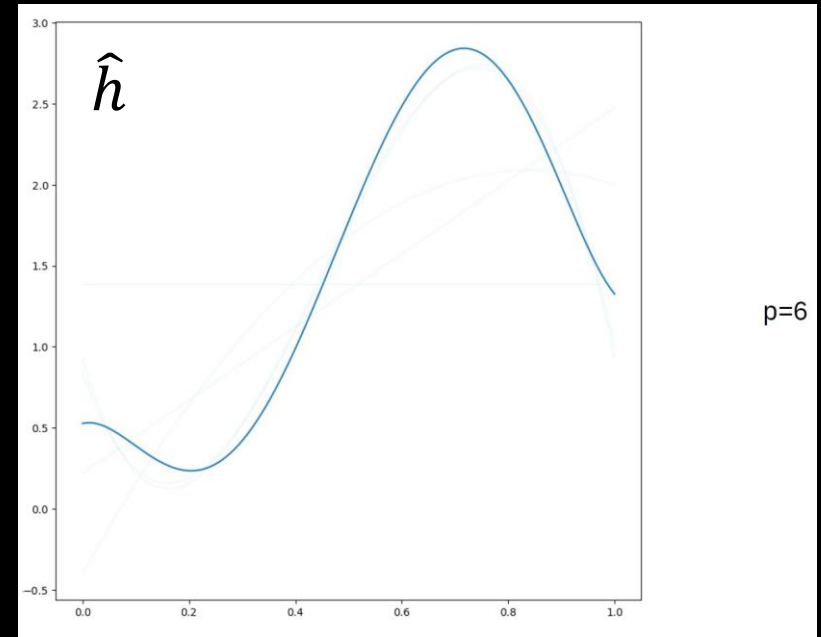
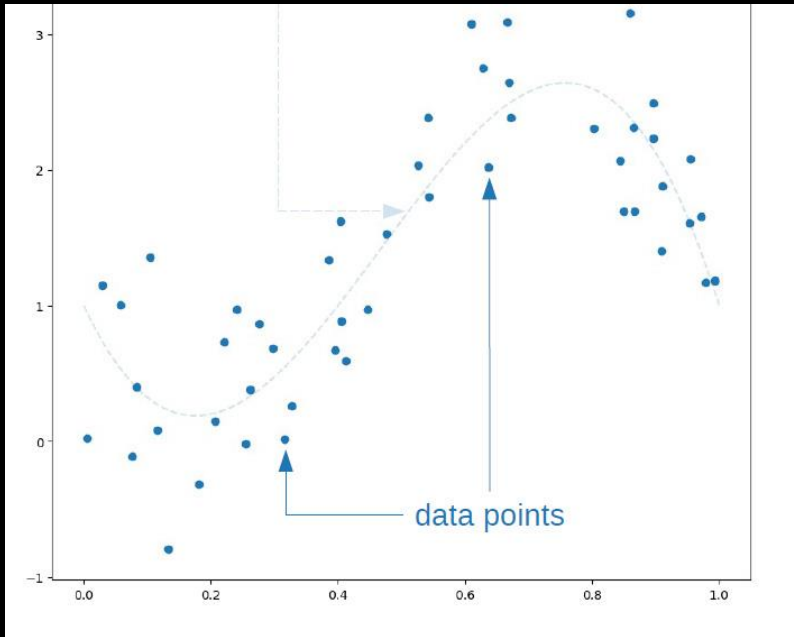
$$\mathcal{H}_4$$

The best quartic that fits the training data



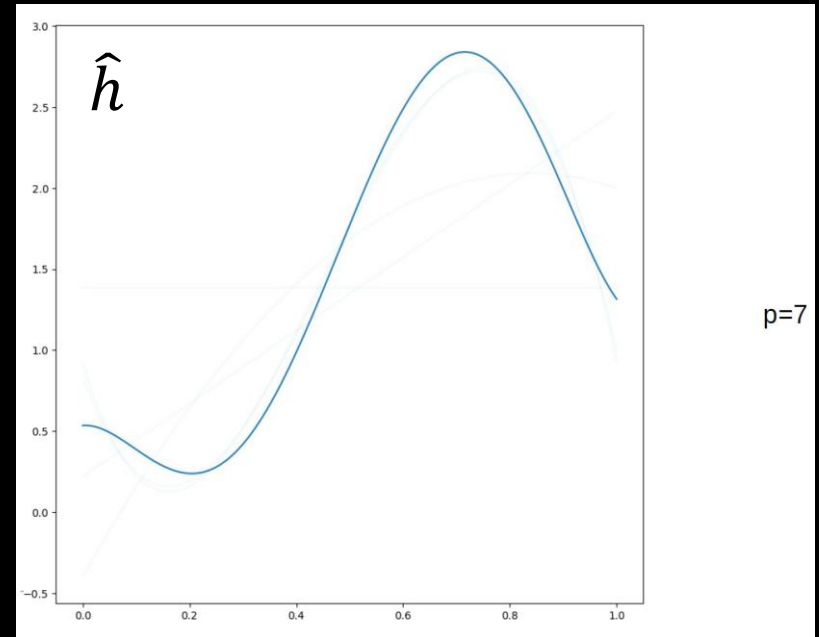
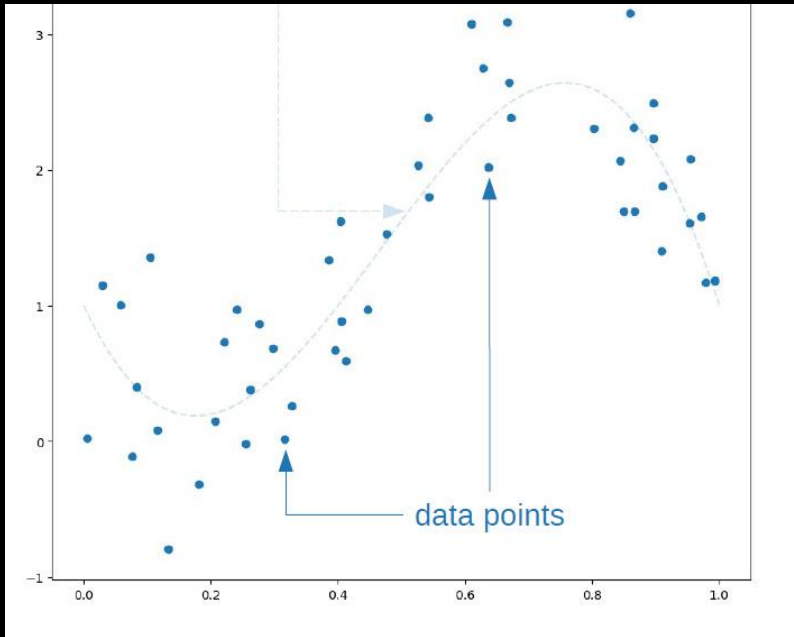
\mathcal{H}_5

The best quintic that fits the training data



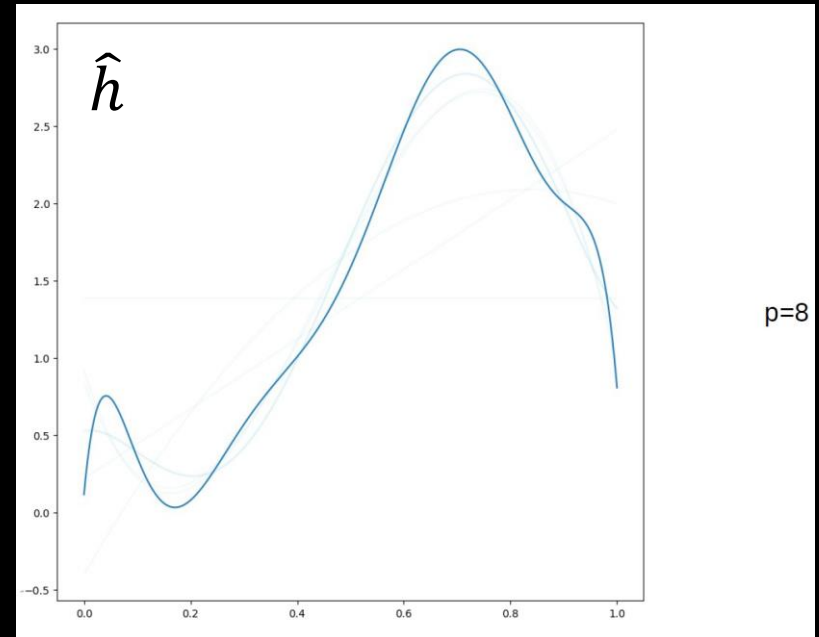
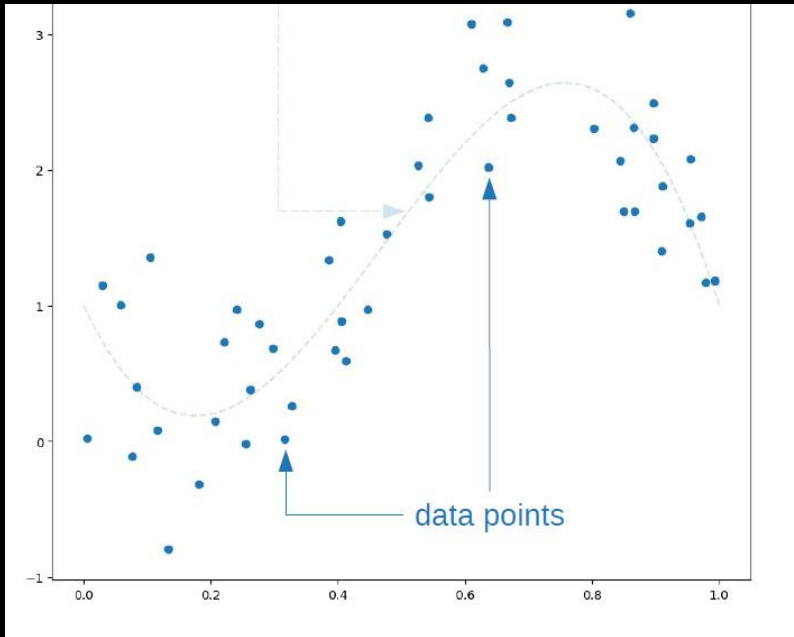
\mathcal{H}_6

The best 6th degree polynomial that fits the training data



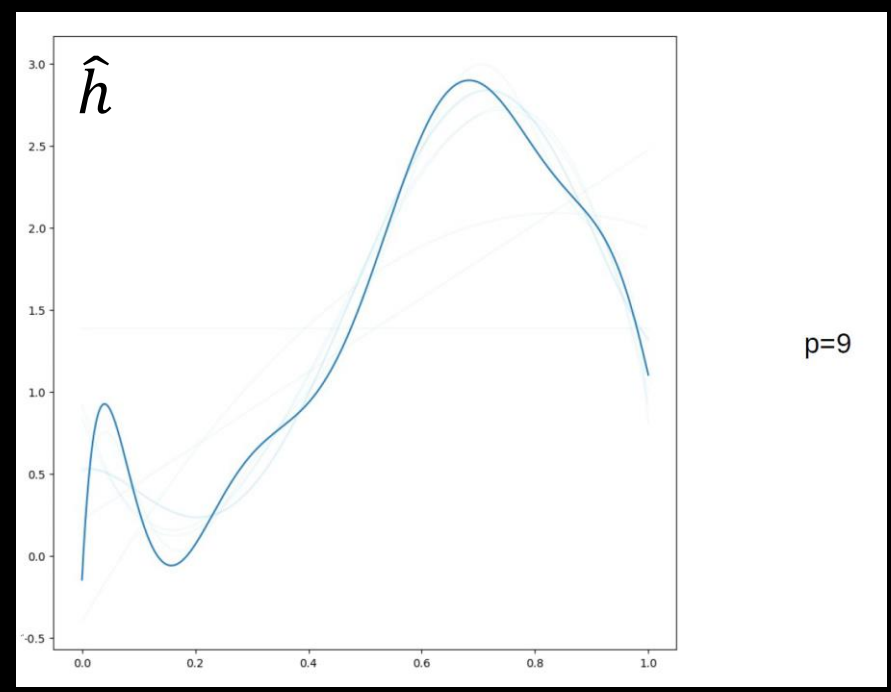
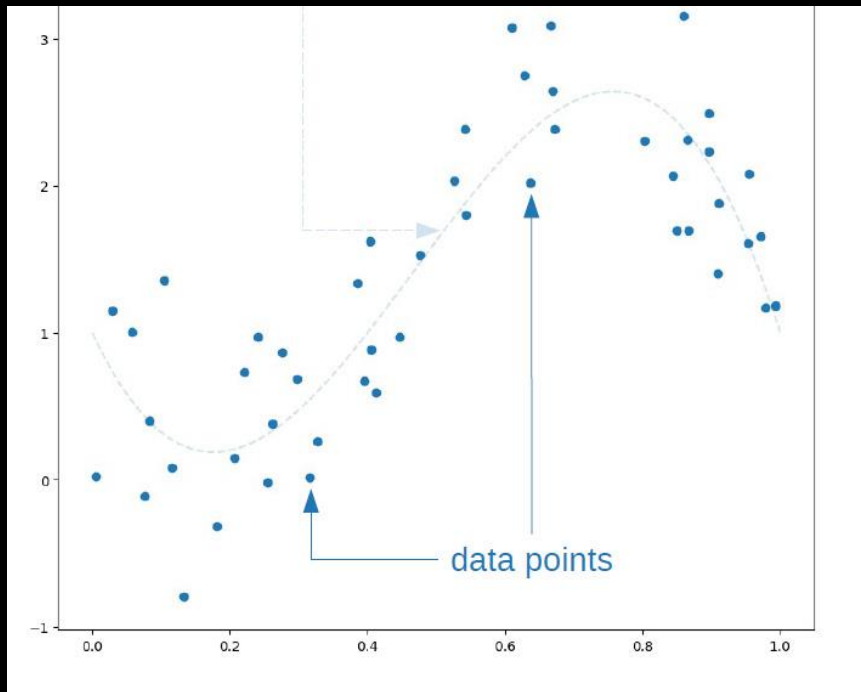
\mathcal{H}_7

The best 7th degree polynomial that fits the training data



\mathcal{H}_8

The best 8th degree polynomial that fits the training data
This is the first time we start to see evidence of overfitting.



\mathcal{H}_9

The best 9th degree polynomial that fits the training data

Overfitting is already a serious issue at degree 9, which is still far below the number of data points.

This shows the overfitting issue cannot be taken lightly even if when there is a fair amount of data.

Continuing

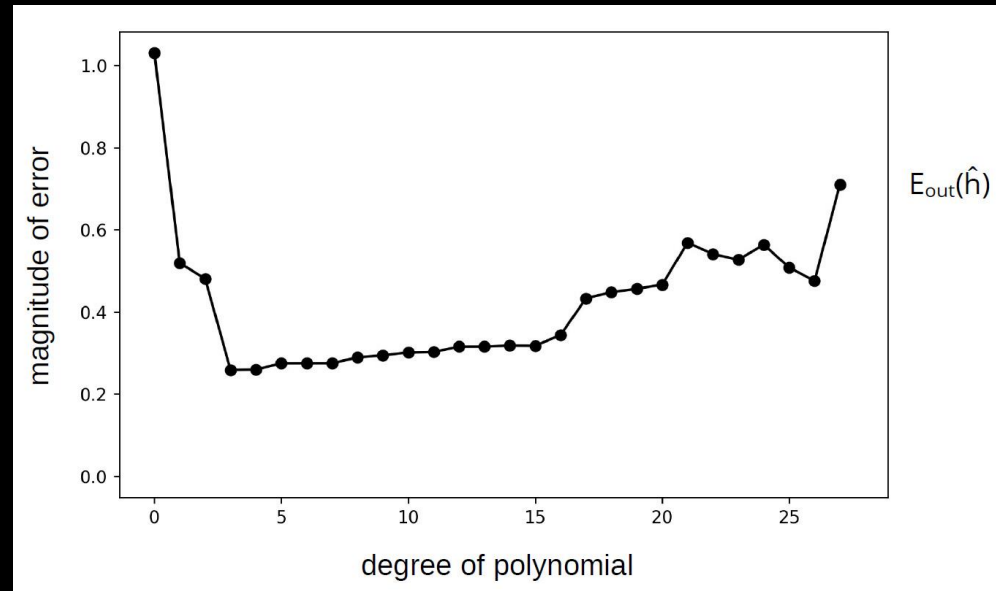
- We're going to take this all the way to 27th degree polynomials.
- But we're going to stop showing the best fit \hat{h} .
- Instead, we will graph the various out-of-sample errors as functions of the degree.
- We'll start with $E_{\text{out}}(\hat{h})$.

$$E_{\text{out}}(\hat{h})$$

This is the total error on the final chosen best performer, for each hypothesis set \mathcal{H}_p

p is the horizontal axis

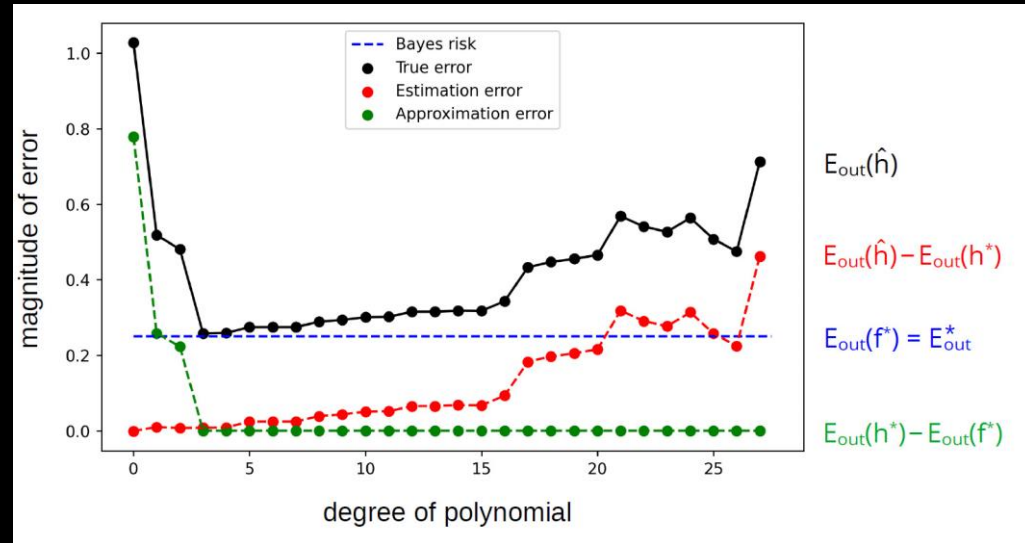
- The error is very high for $p = 0, 1$ and 2.
- It hits its minimum at $p = 3$ since that's the true model.
- Then it raises from there, because of overfitting.
- It bottoms out at 0.25 which is the Bayes (minimal) possible error.



Approximation, Estimation and Bayes Errors

The total error is the sum of the three components.

- The blue dashed line is the Bayes risk, which does not depend on \mathcal{H} .
- The orange line is the estimation error, which starts low and climbs due to overfitting.
- The green line is the approximation, which drops to zero when $p = 3$ and \mathcal{H} contains the true model.



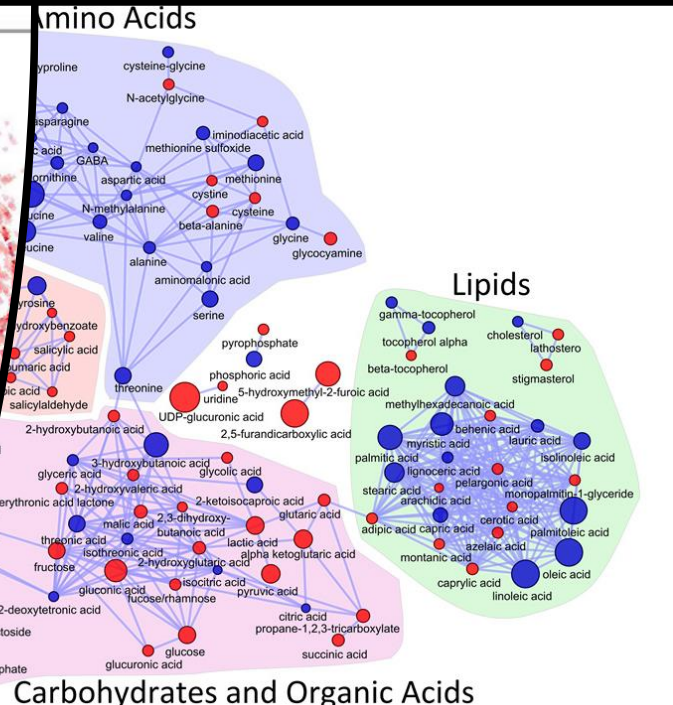
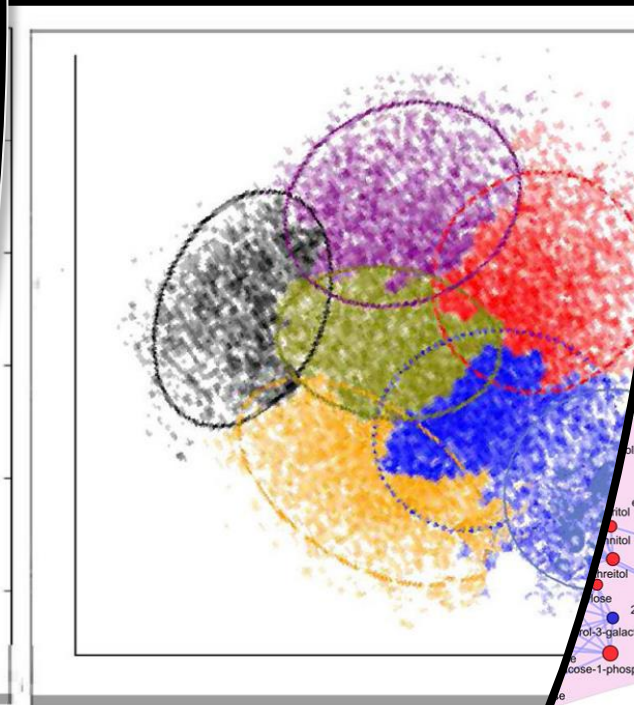
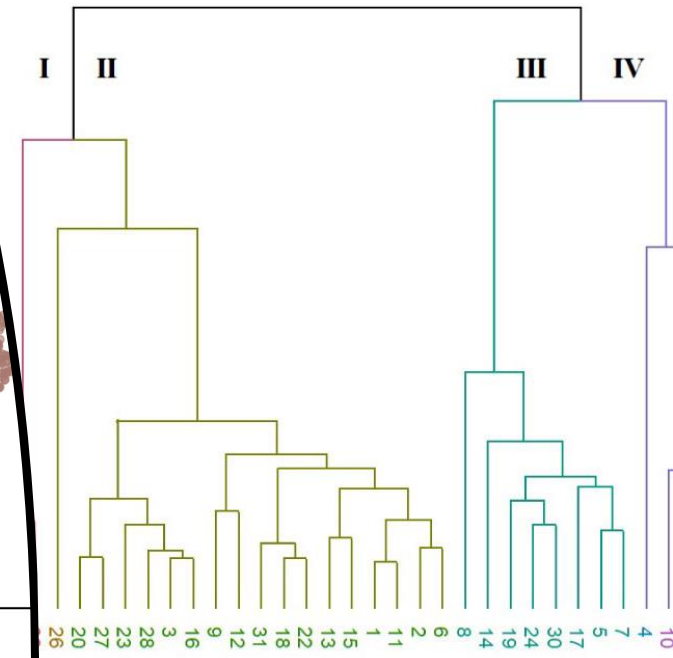
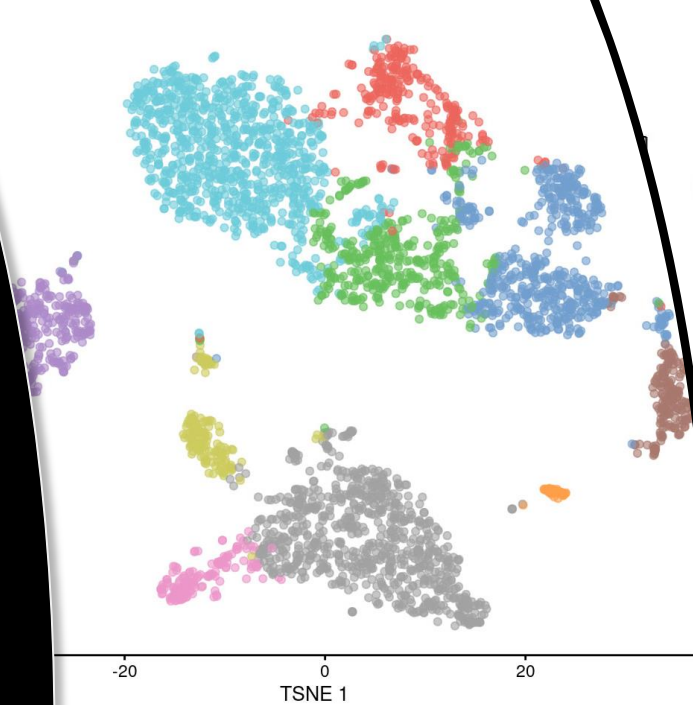
Conclusions

- Fitting training data well means $E_{\text{in}}(h) \approx 0$.
 - ≈ 0 means “is close to zero”.
 - That is different from making good predictions about new data, which means $E_{\text{out}}(h) \approx 0$.
- Complex hypotheses sets lead to overfitting the test data resulting in $E_{\text{in}}(h) \approx 0$ while $E_{\text{out}}(h)$ is not close to zero and may be quite large.
- Overly simple hypotheses sets cannot fit the data well, meaning $E_{\text{in}}(h)$ is large *for all* $h \in \mathcal{H}$.

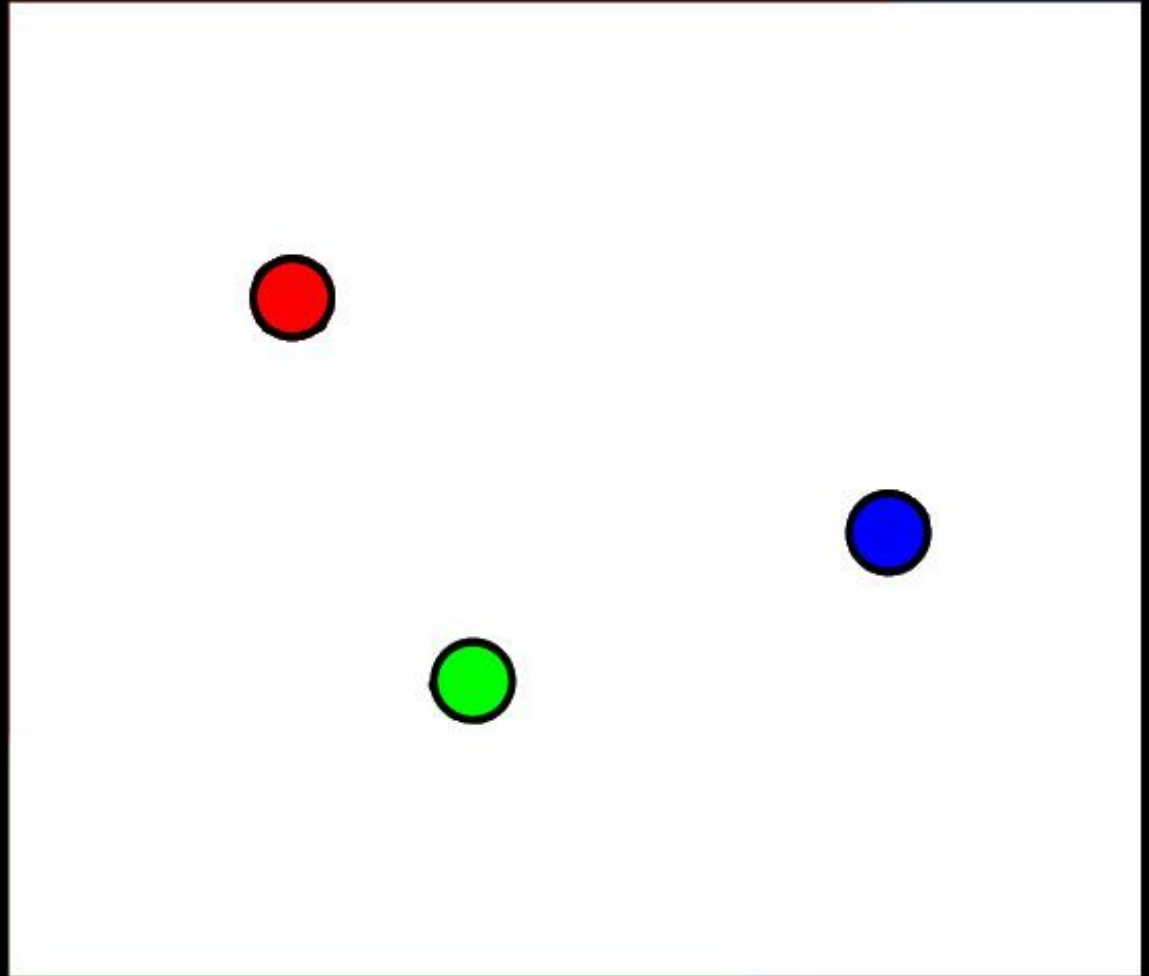
NEW TOPIC

Unsupervised Methods

Unsupervised
Clustering plays a
big role in biology
and bioinformatics.



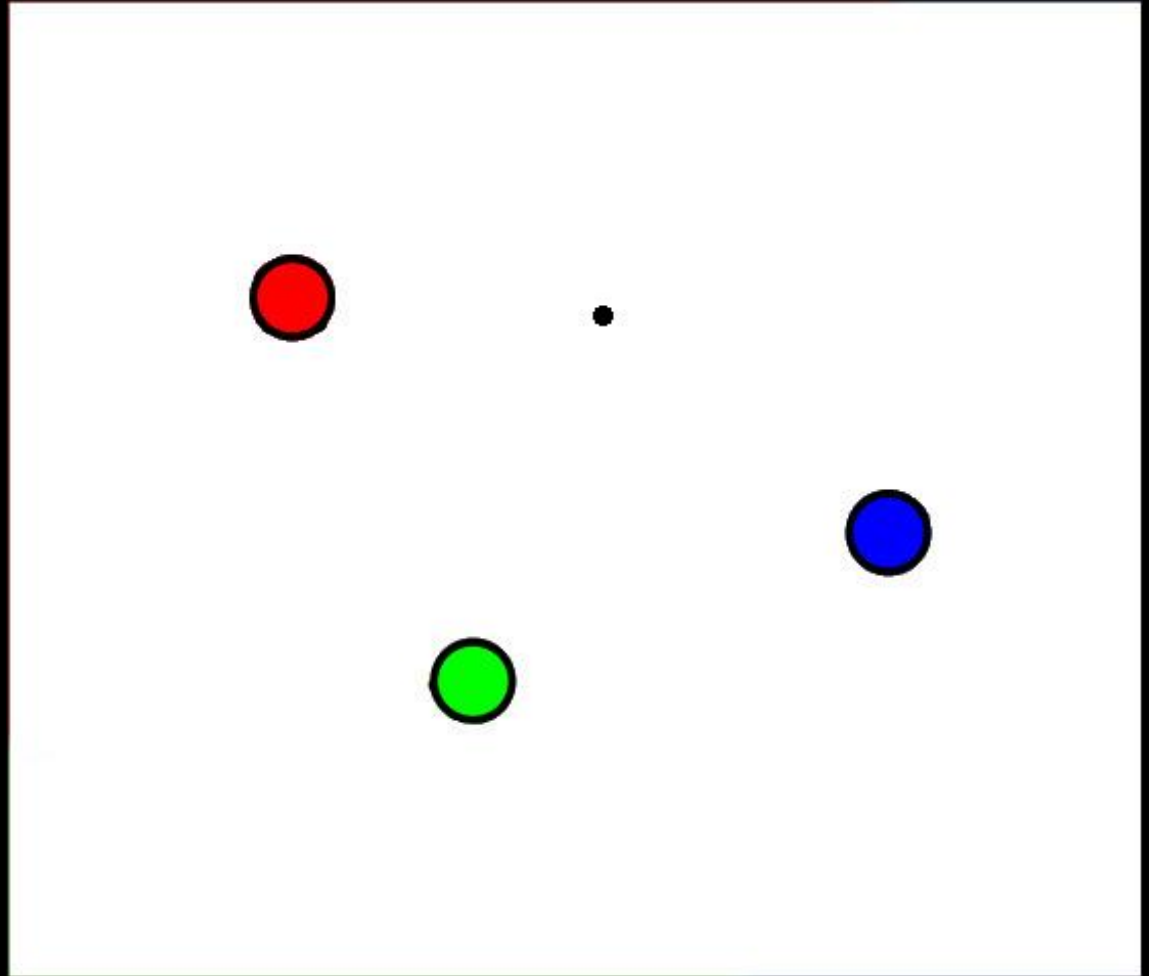
Centroids



Consider plotting
several points in the
plane.

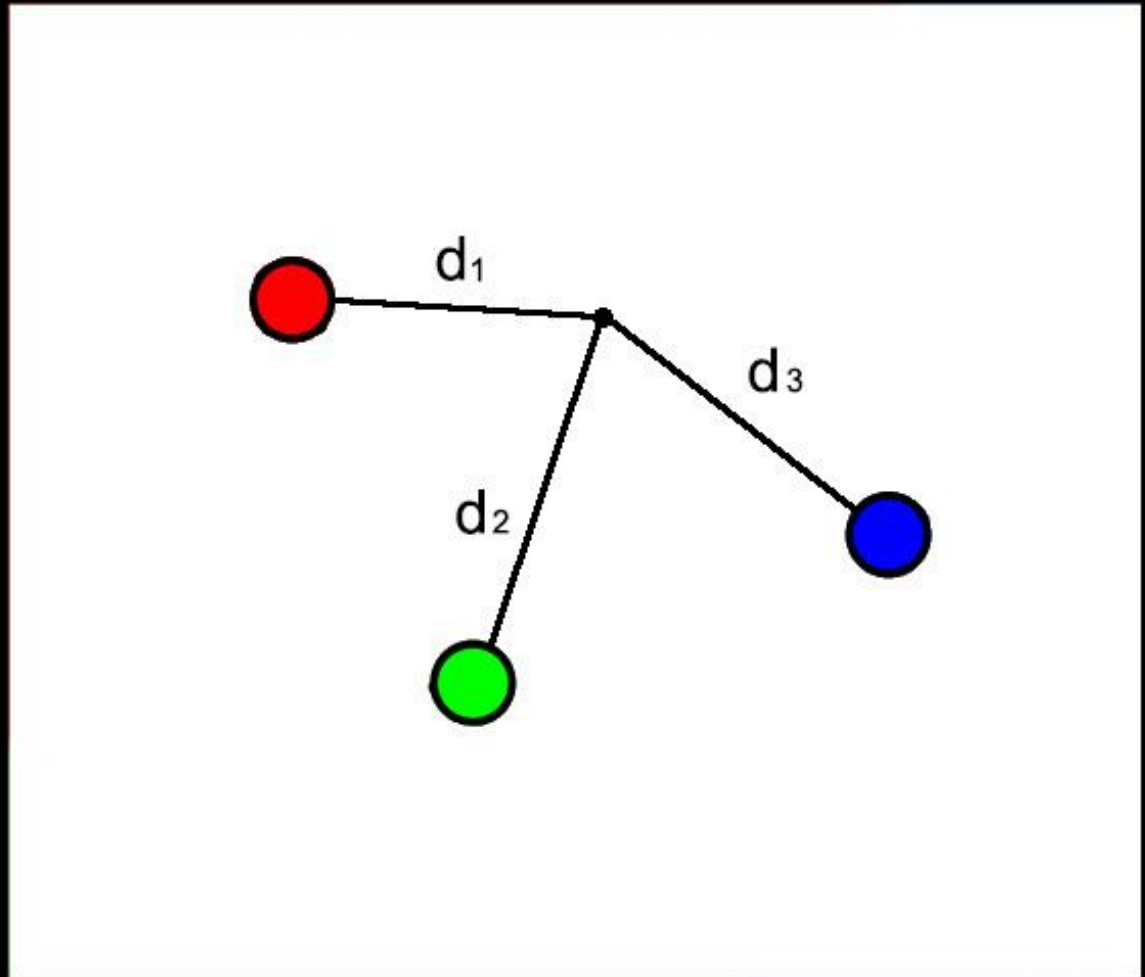
Call them “centroids”

Centroids



Every additional point in the plane is closest to one of the centroids.

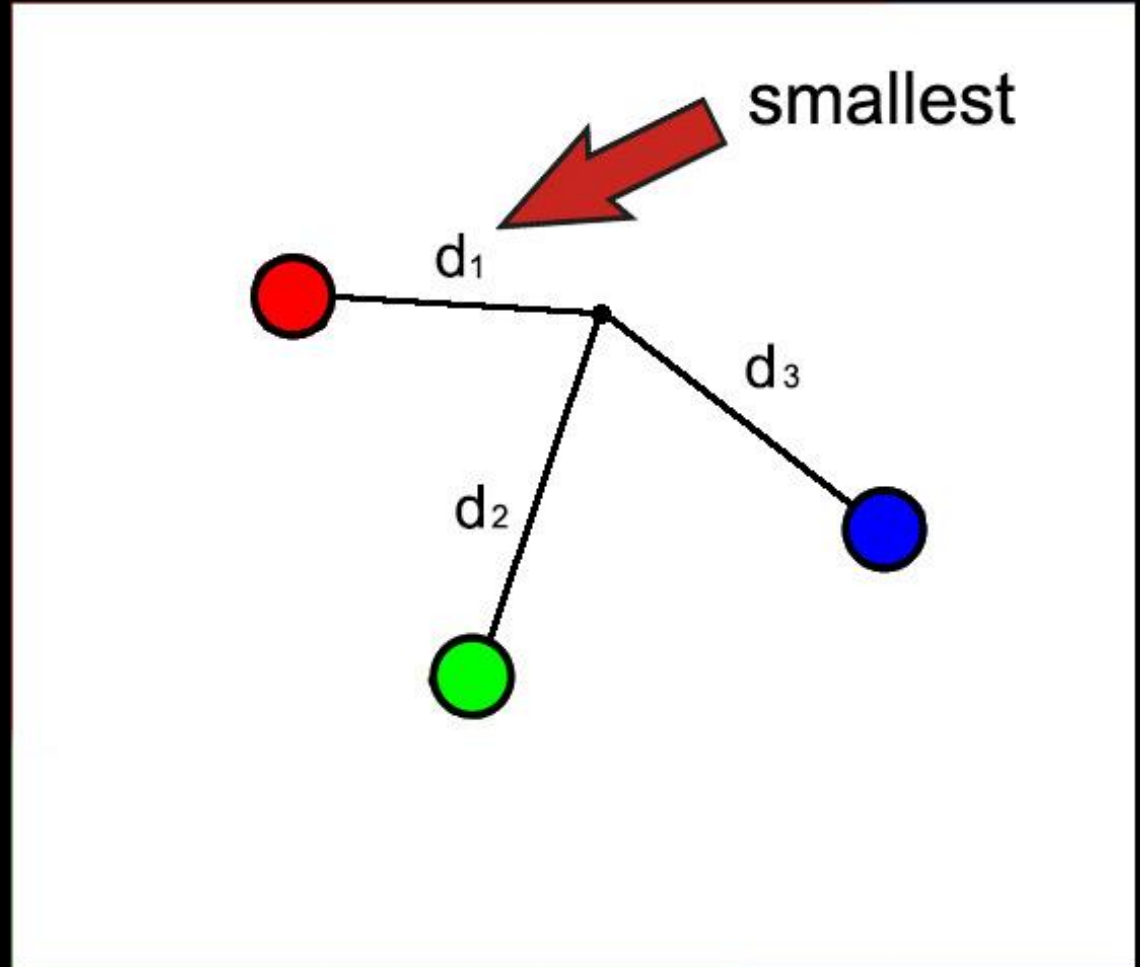
Centroids



Calculate the three distances d_1, d_2, d_3 .

For most points, one of the distances will be smaller than the other two.

Centroids



Suppose d_1 is the smallest.

$$d_1 < d_2 \text{ and } d_1 < d_3.$$

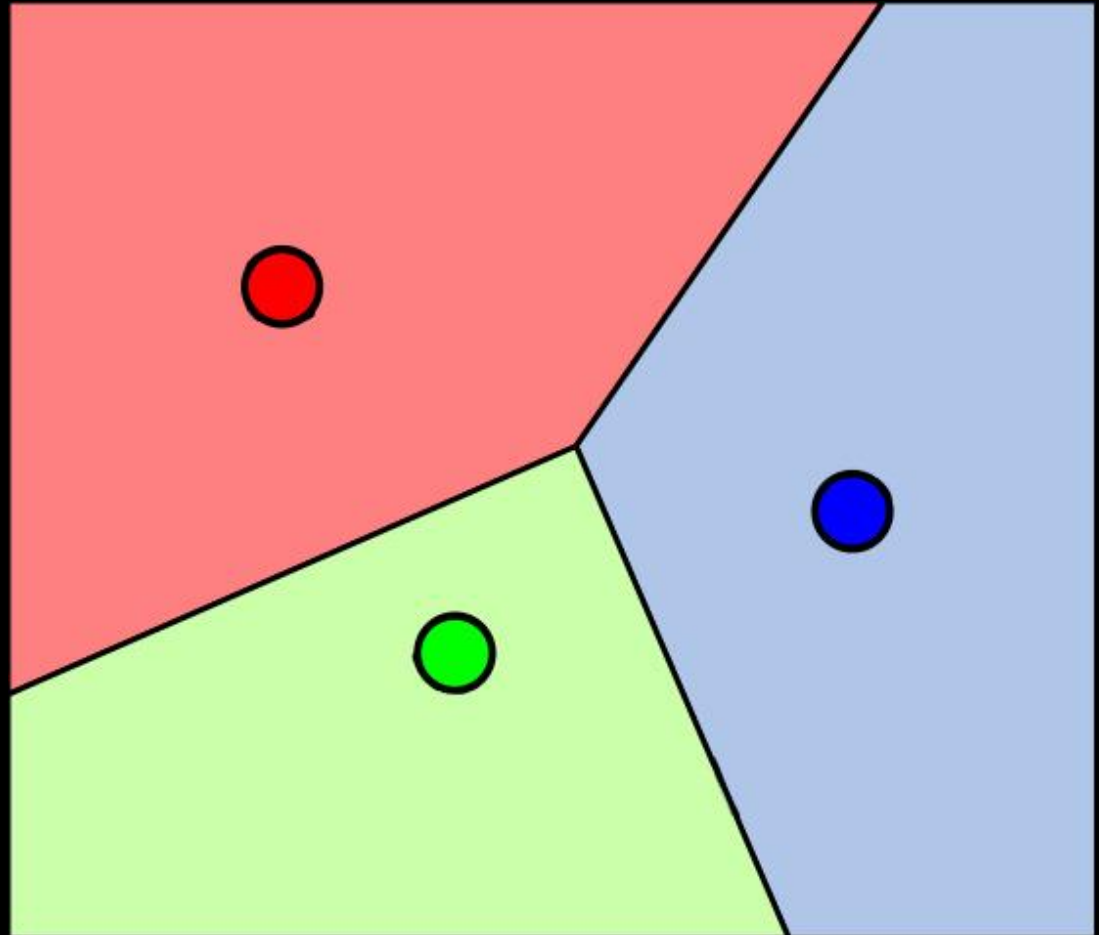
Then color the point red.

Centroids

Therefore, three points partition the plane into three regions.

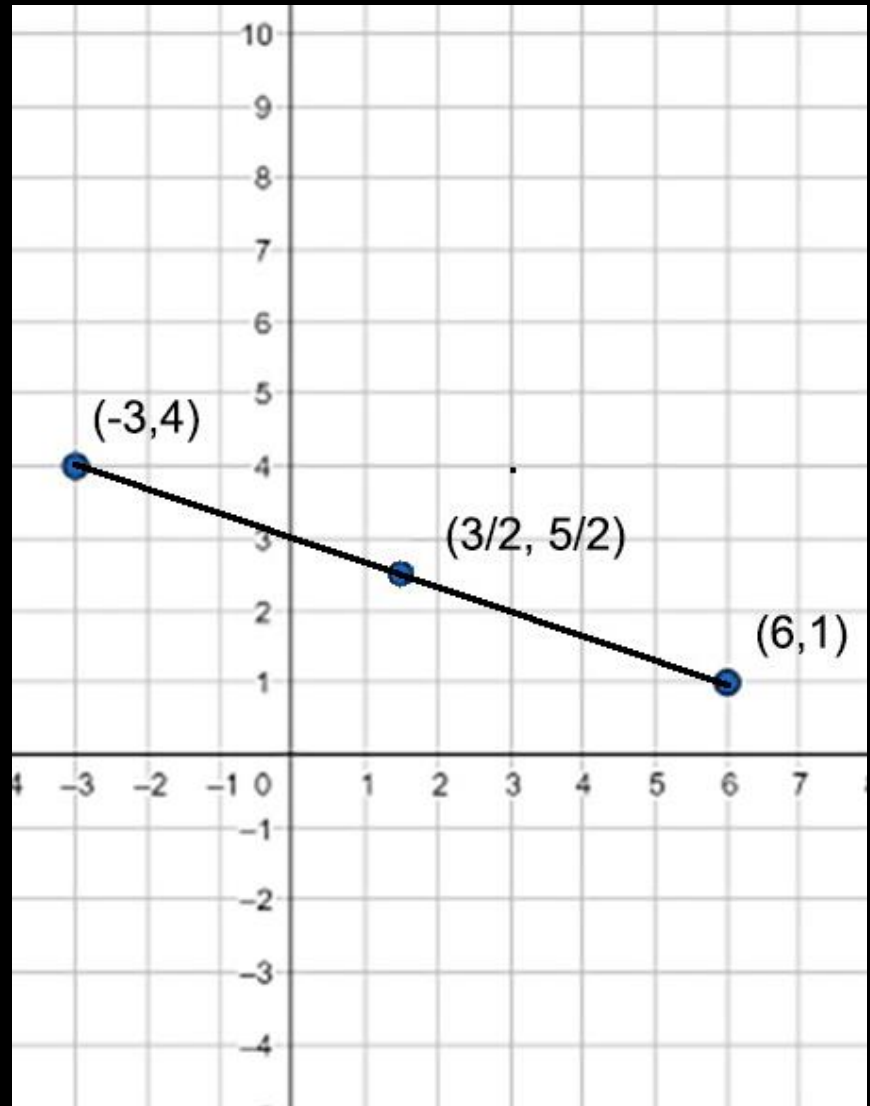
The black lines represent the ties.

In general, n 'centroids' partition the plane into n regions.



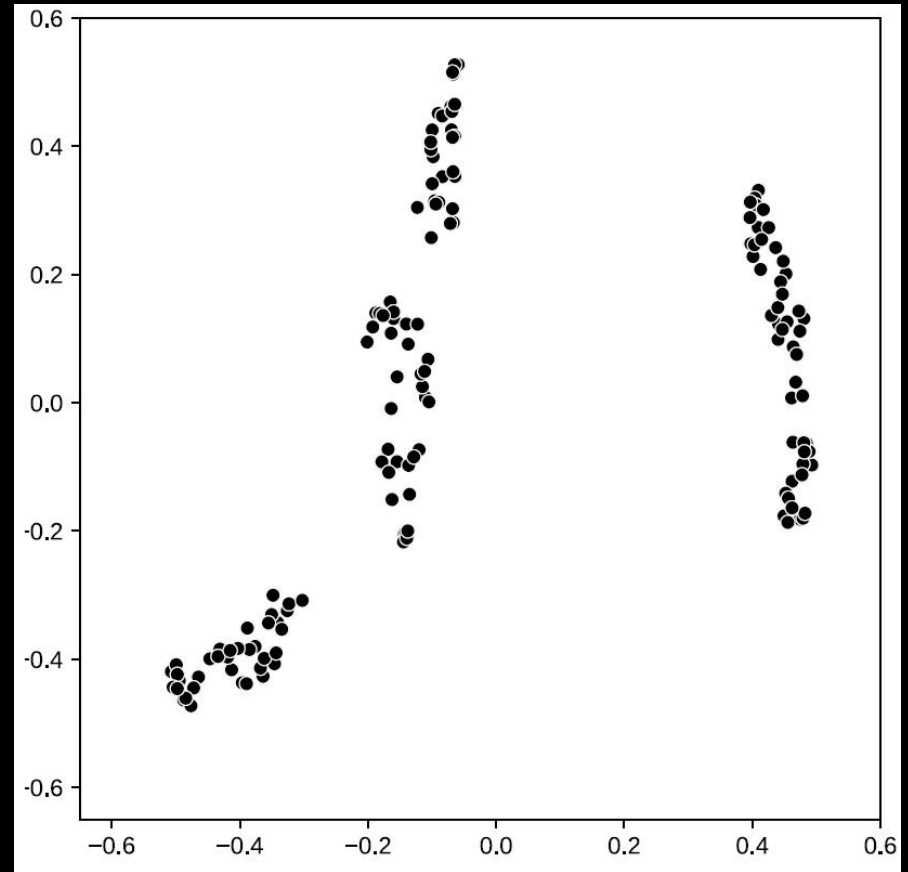
Means in the Plane

- What do we mean by “the mean” of a bunch of points in the plane?
- Points have two coordinates.
- The x -coordinate of the mean is the mean of their respective x -coordinates.
- The y -coordinate of the mean is the mean of their respective y -coordinates.



Unsupervised Clustering

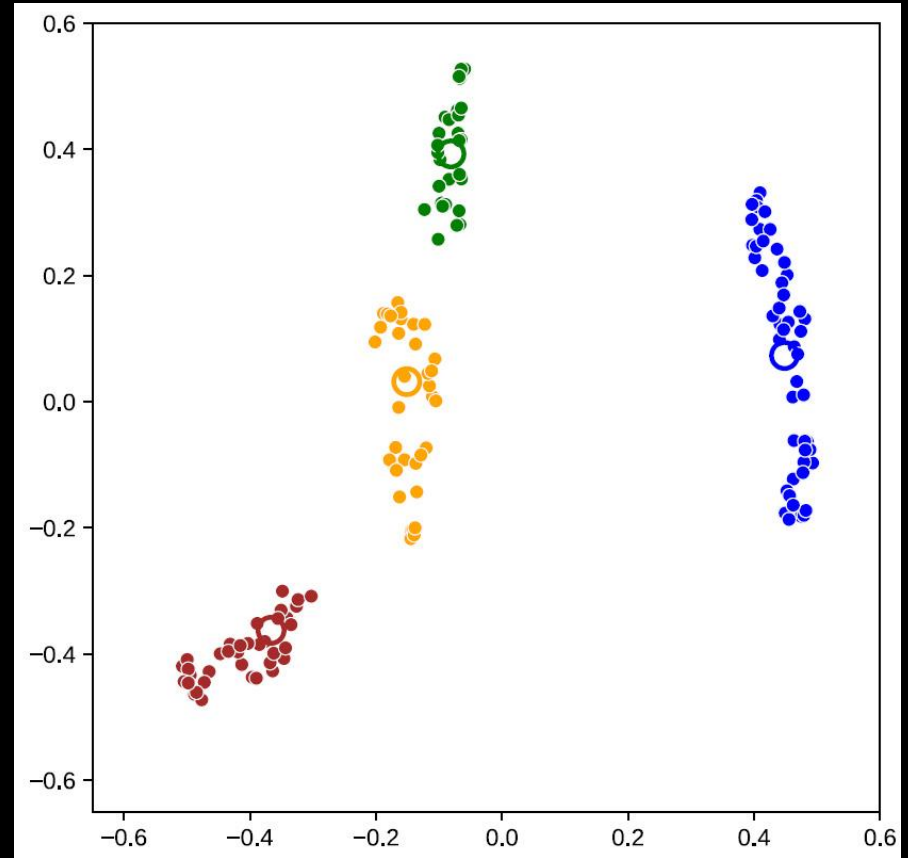
- There appear to be four clusters here.
- The goal is to make an algorithm that can figure that out and identify the members of each cluster.



Supervised Clustering

If we knew the clustering ahead of time, then we could make centroids be the means of each of the clusters.

That would be
“supervised clustering”

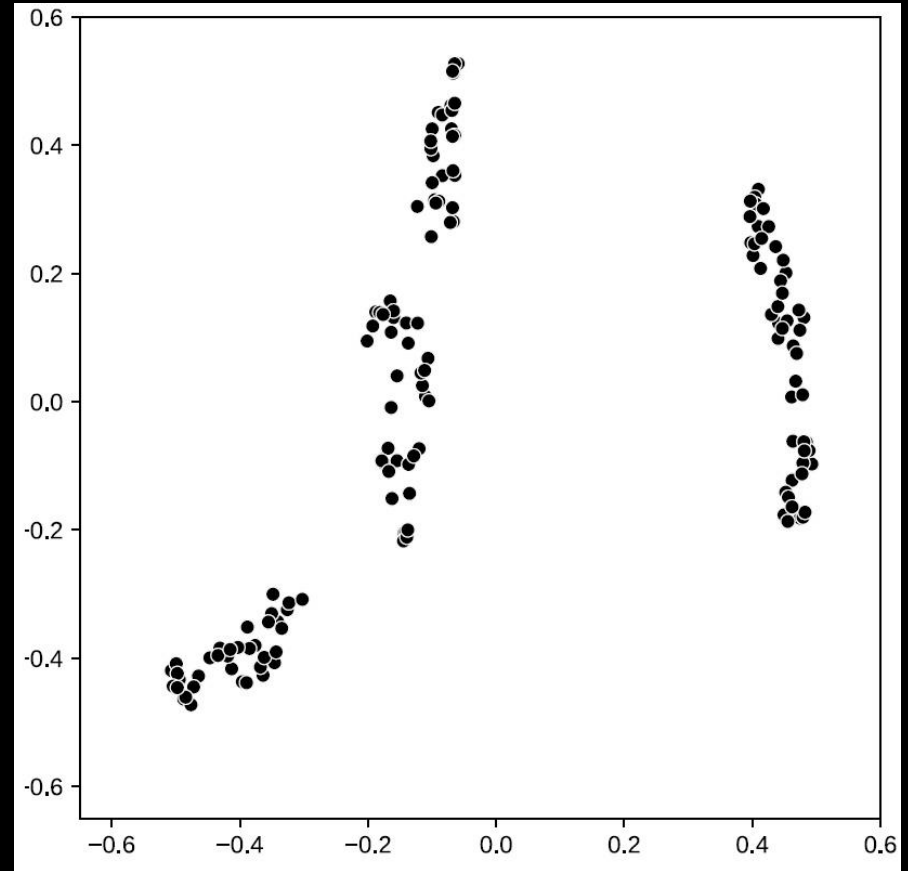


K-Means Clustering

Unsupervised Clustering

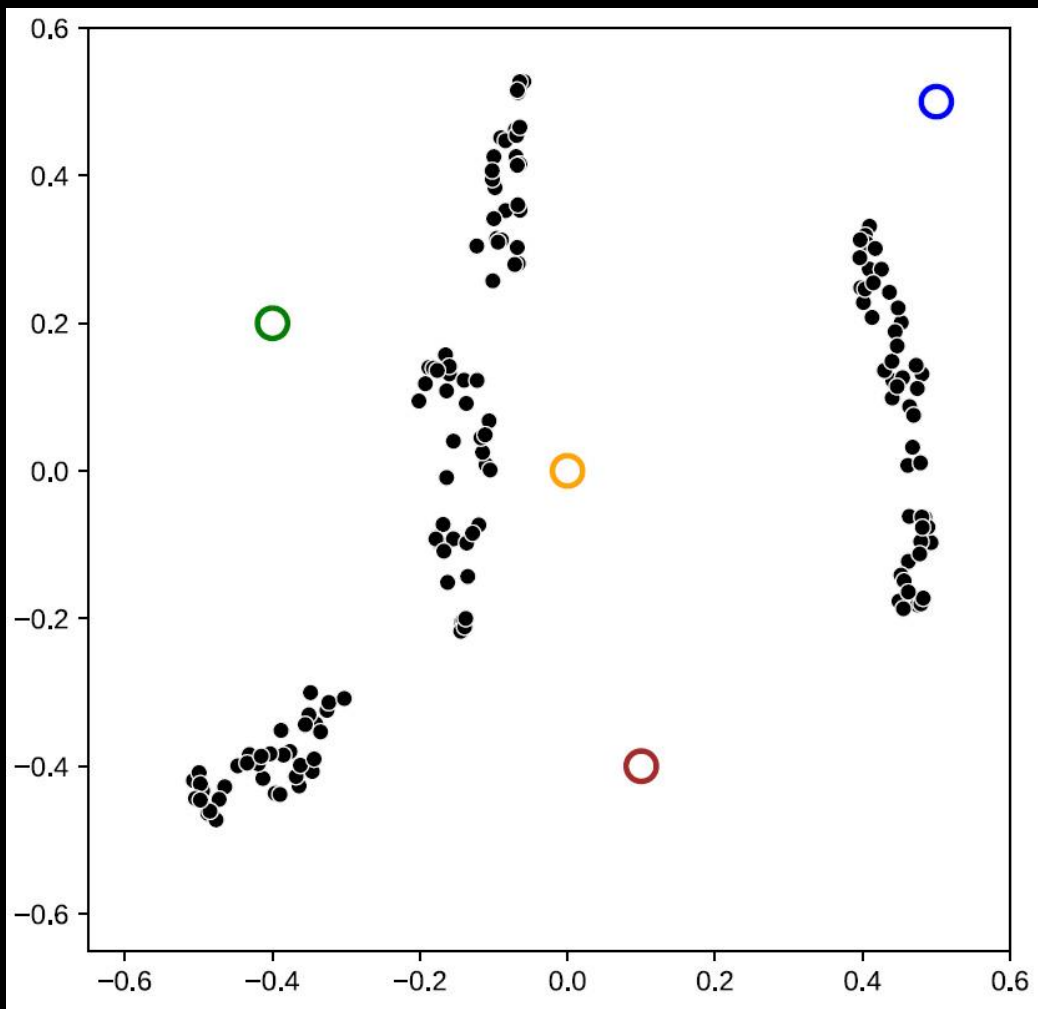
K-Means clustering separates the points into *K* clusters according to some optimization criterion.

Assume for now that *K*, the number of clusters, is known – we'll see how to determine *k* later.



Learning vs. Optimization

- When we have training data, data where we know the truth, the goal is usually “learning”.
 - We call that “supervised learning”.
- When we do not know the truth, such as in clustering, we don’t “learn” instead we “optimize”.
 - And we call it “unsupervised learning”.
 - We establish a metric that measures quality of something (usually clusters).
 - The metric can then be optimized (which means we search for values of parameters that make the metric as big, or as small, as possible, depending on which is better).



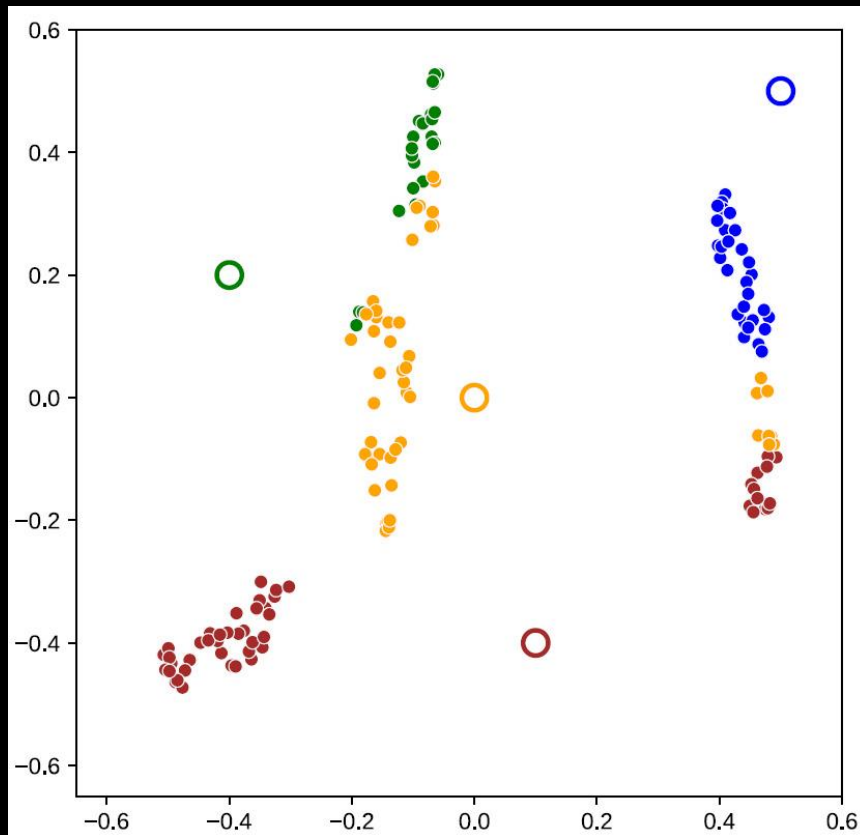
K-Means Clustering

Start by randomly assigning locations to four centroids.

K-Means Clustering

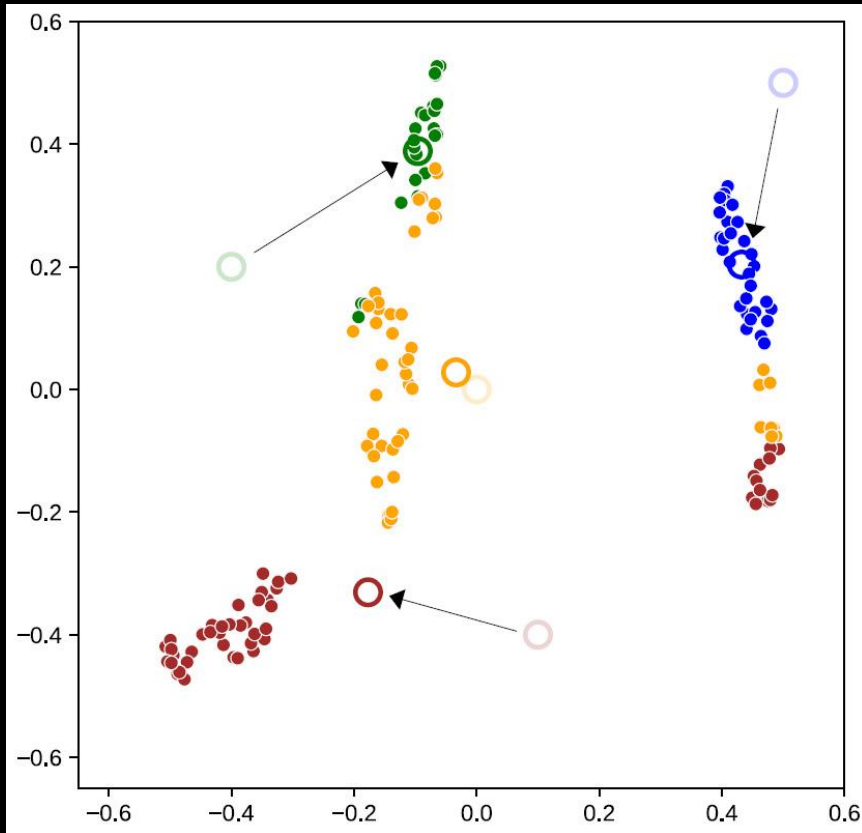
Assign the data points to the clusters.

Color each point by which centroid it is closest to.



K-Means Clustering

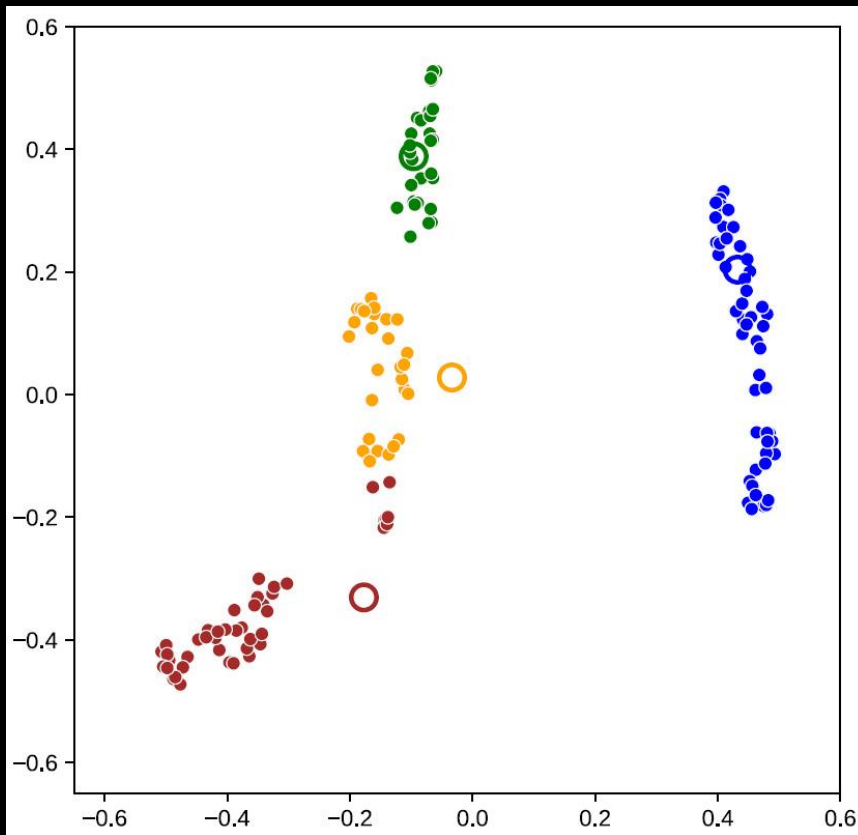
Move the four centroids to be at the means of the current clusters.



K-Means Clustering

Reassign points to clusters based on the new locations of the centroids.

You see this is an iterative process that improves at each iteration.

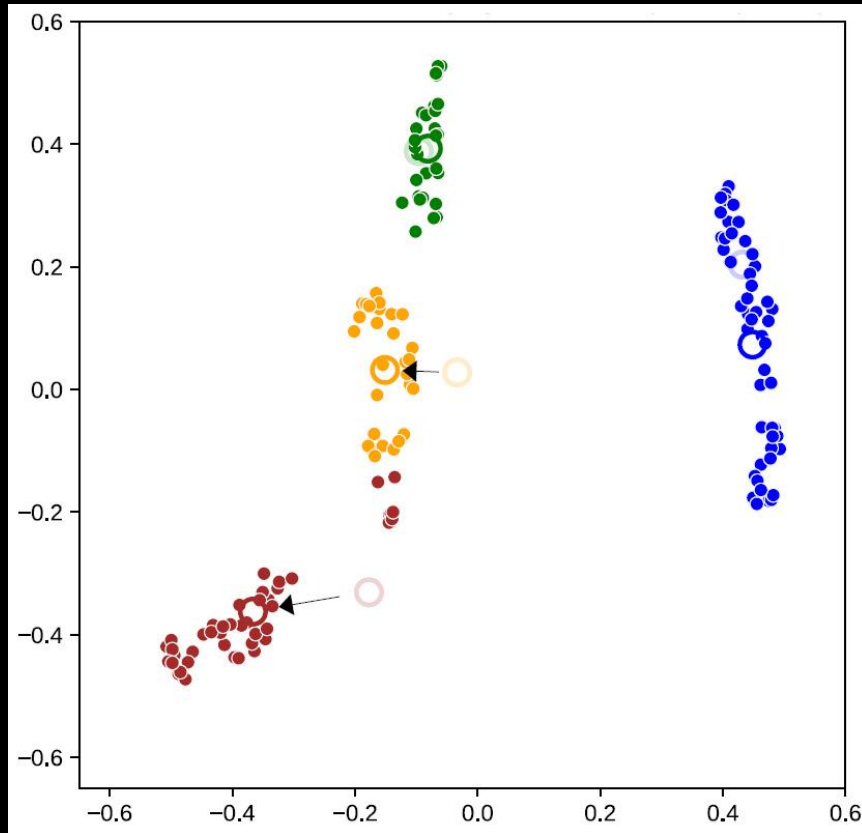


K-Means Clustering

Move the centroids again, to the means of the new clusters.

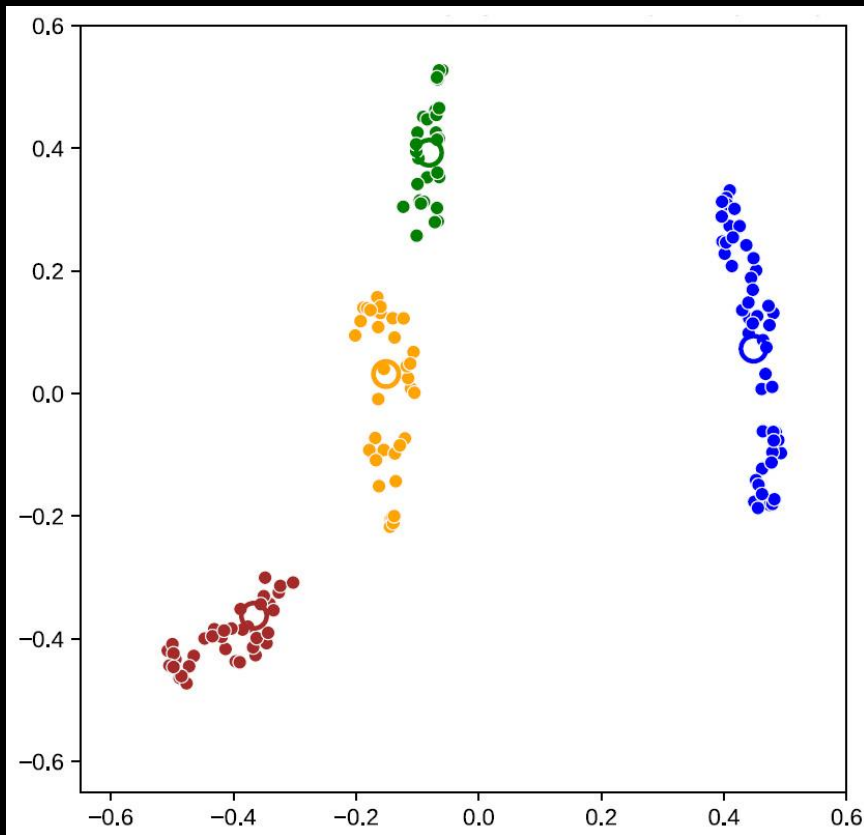
In just two iterations it's already close to optimal.

Optimal here means nothing changes upon subsequent iterations.



K-Means Clustering

Reassign points to clusters based on the new centroid positions.

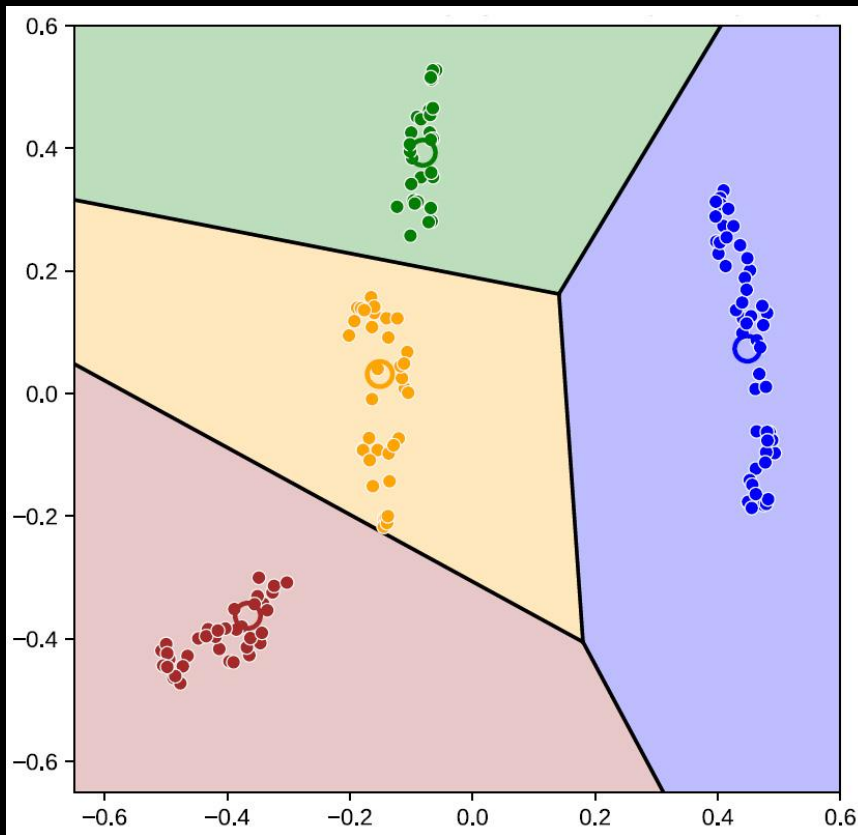


K-Means Clustering

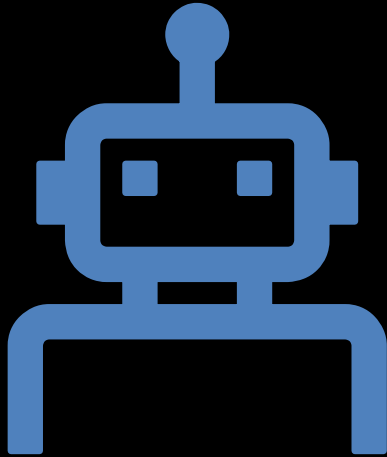
Looks right.

Further iterations might change the partition of the plane slightly but will no longer change the clusters.

Time to stop.



The Big Picture



Most unsupervised machine learning methods are based on iterative procedures such as this.

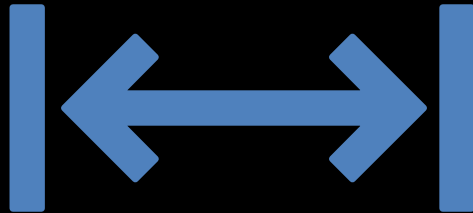
The computer starts by knowing nothing, it starts by taking a random guess at the parameters.

Here, the centroids are what's guessed at.

If the updating rules are sensible, the estimates of the parameters improve with each iteration.

I.e., the machine learns

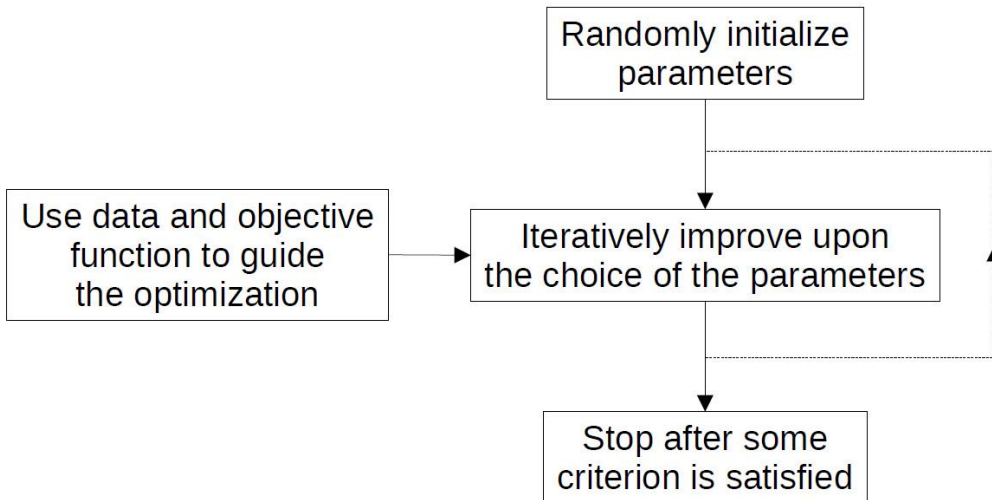
Objective Functions



- How do we know if the parameter updating procedure is sensible?
- We use an *objective function* that the algorithm aims to minimize
- In the case of k -means clustering we have, for clusters C_1, \dots, C_K

$$f(m_1, \dots, m_k) = \sum_{j=1}^k \sum_{x_i \in C_j} d(x_i, m_j)$$

- Learning \leftrightarrow optimization



The Big
Picture

Evaluation of Results

Machine learning algorithms try to decrease the objective function in each step

However, they are almost never guaranteed to find the global minimum of the objective function

Hence, we don't know if we are getting the "best" possible clustering

This raises the issue of evaluation of results

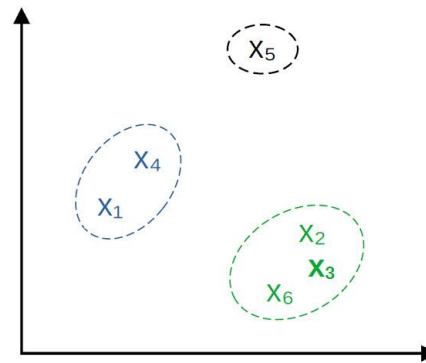


Evaluation of Results

- Compare with external sources of truth, such as GO annotations.
 - Clustering that give the best enrichment p -values is validating.
- Bootstrap — randomly resample from the data set and then rerun the clustering
- To get a handle on how variable the results are as we swap one training data set with another.
- Use statistical measures of purity and homogeneity of clusters.
 - Such as the Silhouette method (next slide)

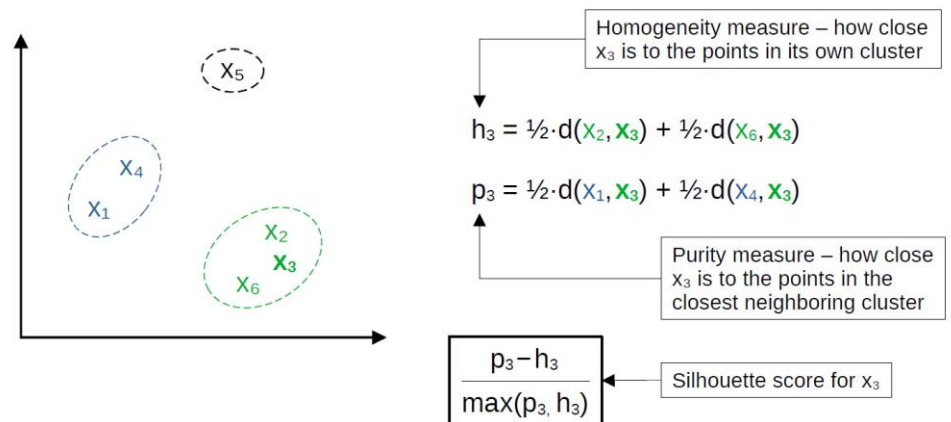
Silhouette Scores

- Silhouette score measures the quality of the assignment of a given point to its cluster



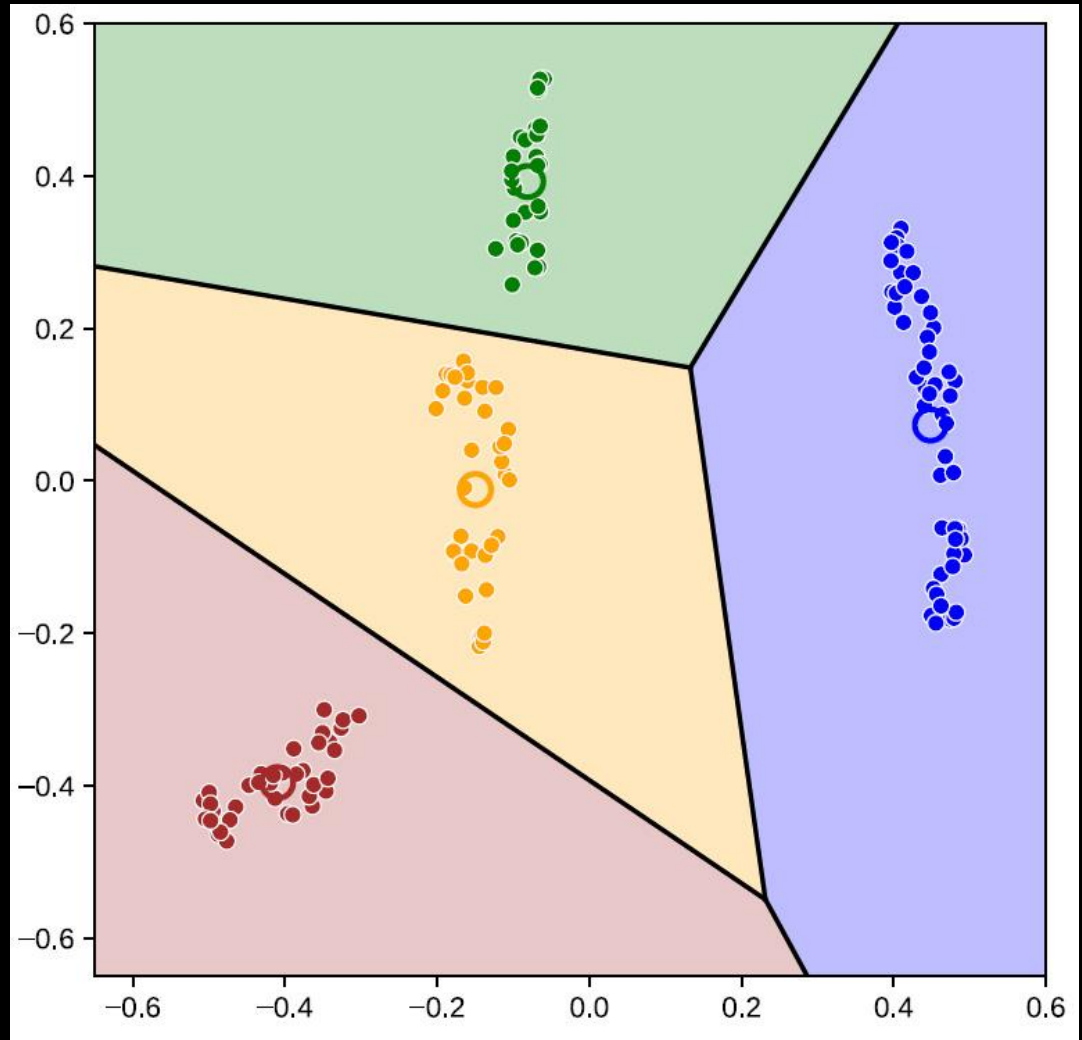
Silhouette Scores

- Silhouette score measures the quality of the assignment of a given point to its cluster



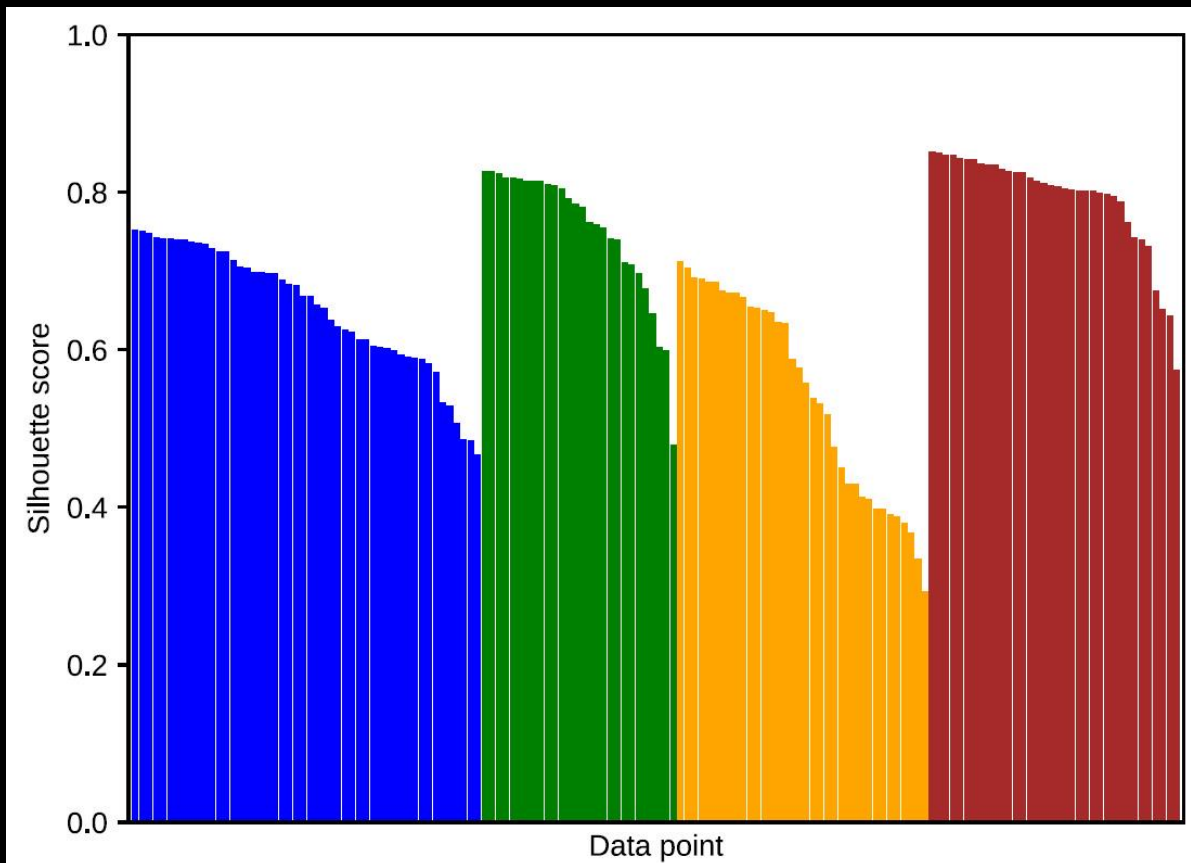
Our Clustering

The clustering from the k -means algorithm, with $k=4$.



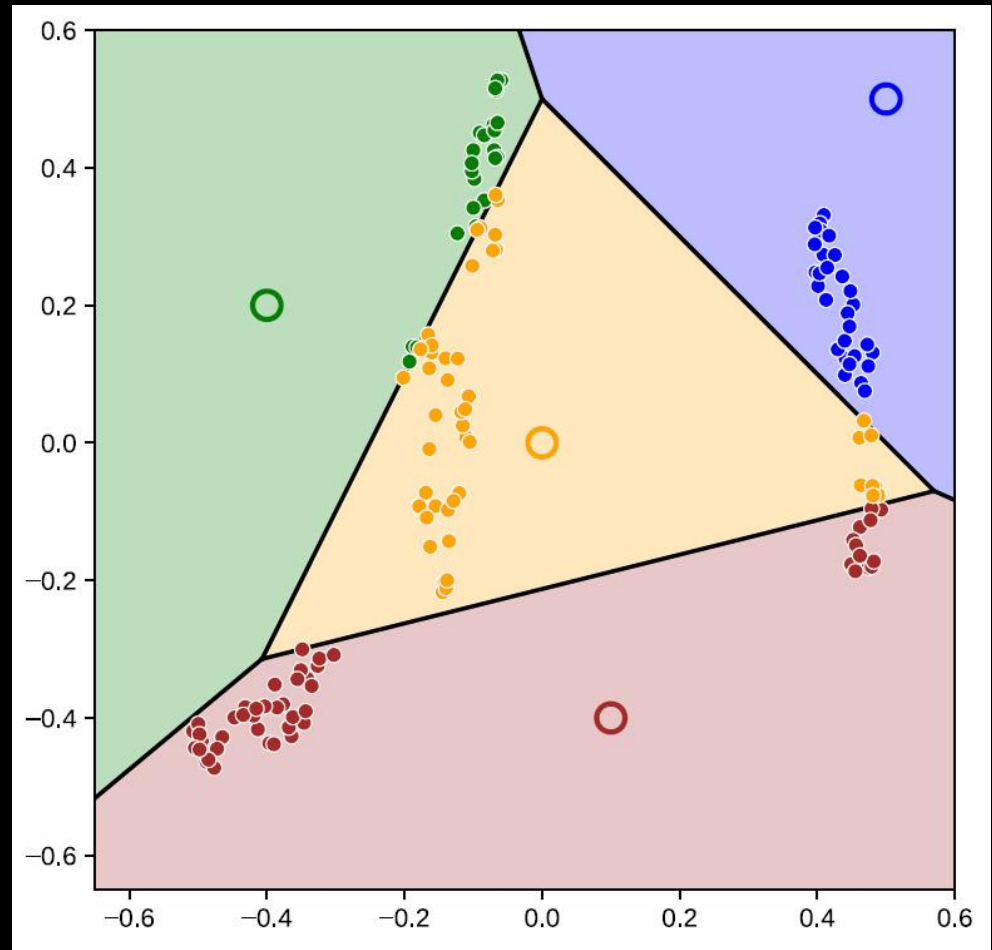
Silhouette Scores

Each of the 150 data points has its own silhouette score.



Average
Silhouette
Score = 0.68

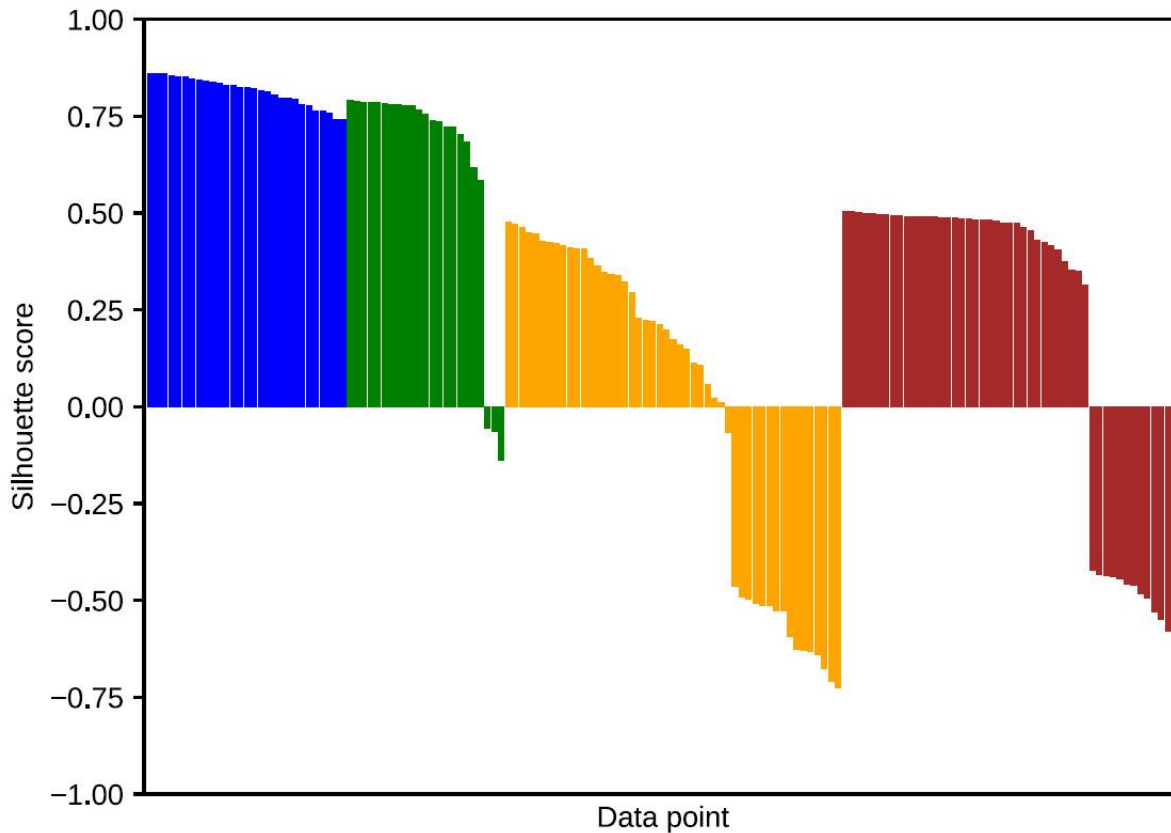
Bad Clustering



The clustering from four randomly placed centroids.

Silhouette Scores for Bad Clustering

Lower silhouette scores indicate worse clustering.



Average
Silhouette
Score = 0.32

Hyperparameters

In supervised learning we must usually specify the form of a model before we have parameters to train.

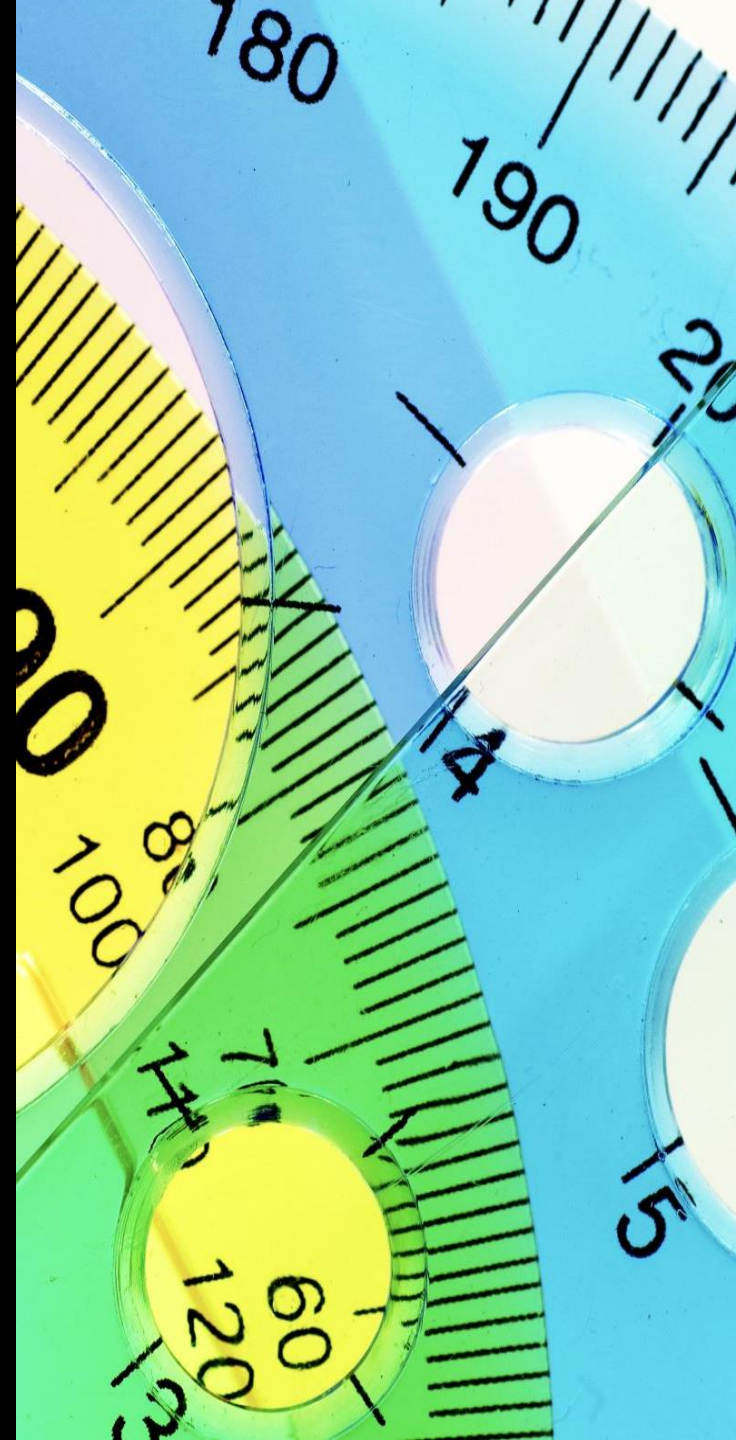
Unsupervised machine learning algorithms usually also require specifying preliminary properties, often themselves parameters.

These are called “hyperparameters”

For example, the K to use in K -means clustering.

Hyperparameters must be specified before the optimization loop can be initiated.

There are many tuning procedures used to determine optimal values of hyperparameters



K-Means Clustering Example

- In the case of k -means clustering, the number of clusters k is a hyperparameter.

- And the objective function is

$$f(m_1, \dots, m_k) = \sum_{j=1}^k \sum_{x_i \in C_j} d(x_i, m_j)$$

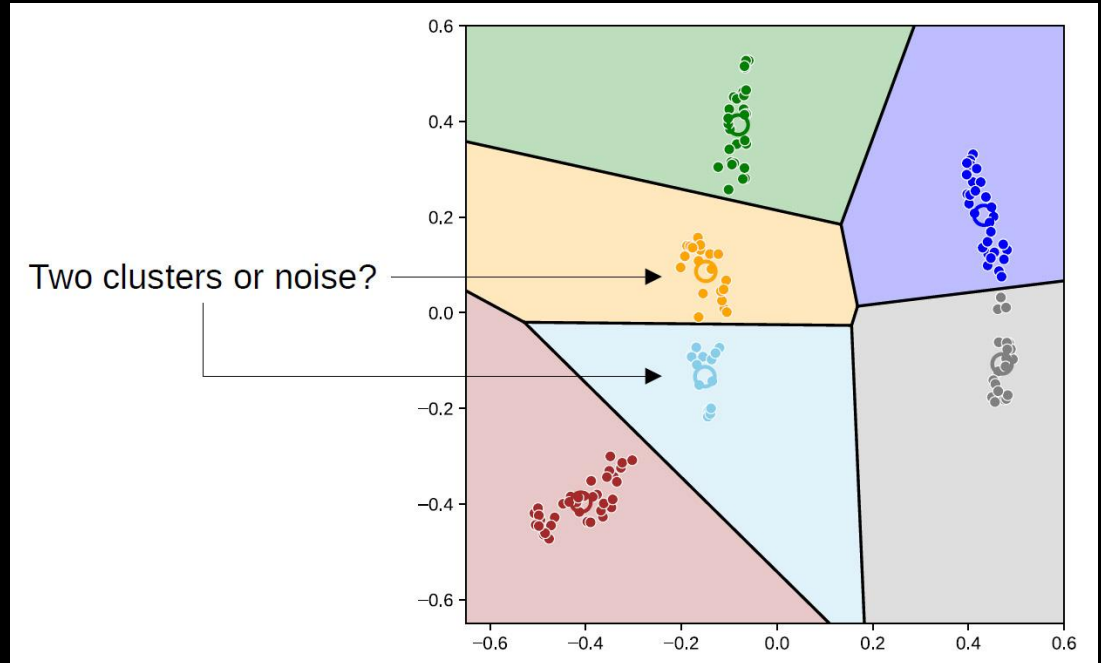
- The objective function is used after k is determined to determine the centroids.
- It gives a meaningful comparison between different clusterings, for a fixed k .
- But it does not give meaningful comparisons between different values of k .

Determining K

- Since objective functions cannot be used to compare between different values K , they cannot be used to determine the best value of k .
- If we increase the number of clusters, the objective function always decreases.
 - If we increase the number of clusters to be equal to the number of data points, then we can drive the objective function all the way down to zero.
- The name for this problem is “overfitting”.
 - Overfitting comes up also in unsupervised machine learning.

Overfitting

- Overfitting can start to happen with k even as small as six.
- The objective function is smaller than with $k=5$ but that does not mean it's better.



Average Silhouette Scores

The average silhouette score does allow for a meaningful comparison between different values of k .

Thus, allowing for k to also be learned from the data.

