Analysis III: Regression & Classification

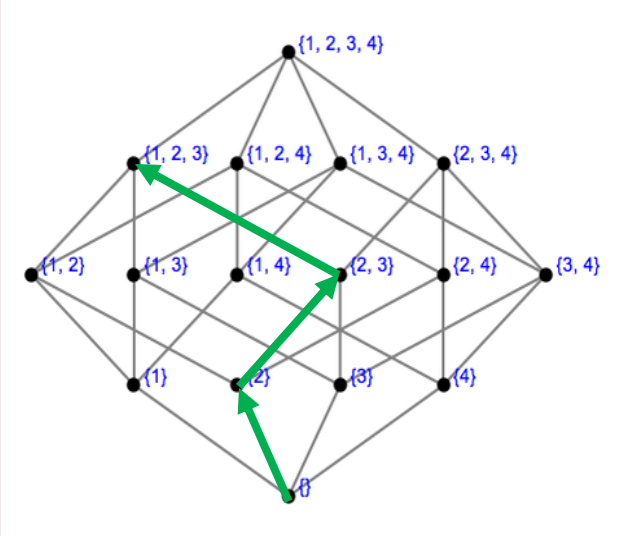
5ARB0: DATA ACQUISITION & ANALYSIS

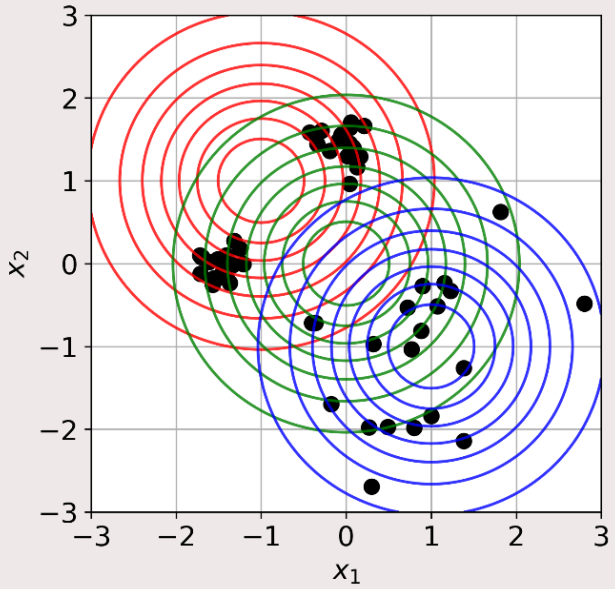
Wouter Kouw

Dept EE – Signal Processing Systems group

Recap

Last time: feature selection/extraction and unsupervised learning.





TU/e

Outline

Regression

- Linear & polynomial regression
- Overfitting & regularization

Classification

- K-Nearest neighbours
- (Multinomial) logistic regression

Function estimation from data

Q: How does an electric car's battery capacity relate to its price?

Figure from https://www.carmagazine.co.uk/electric/ev-car-battery-capacity-tech/

Function estimation from data

Q: Will you reach your travel goal with your car's current charge?

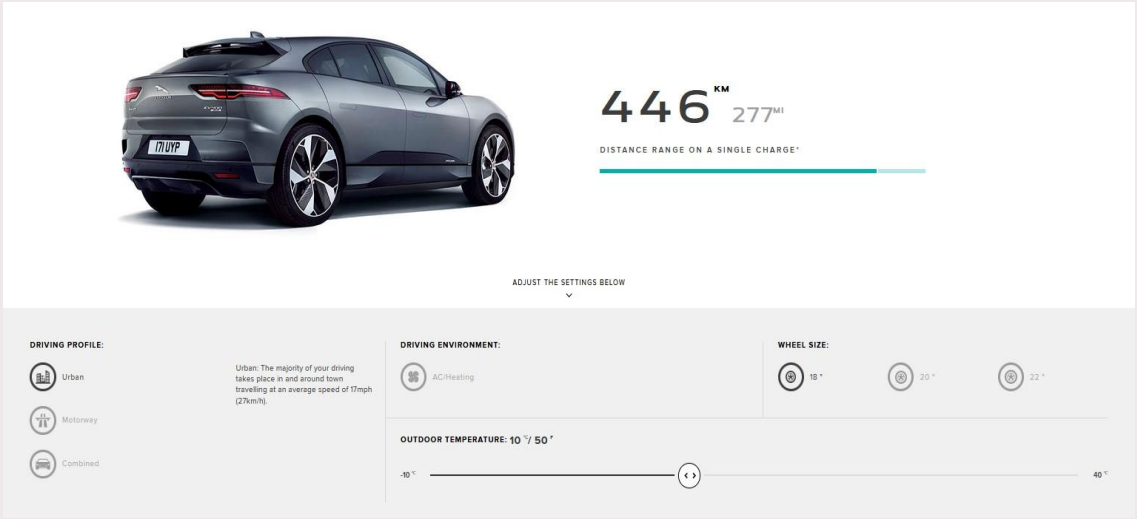


Figure from https://www.jaguar.com/electric-cars/range.html

Function estimation from data

There exists some functional relationship between input x and output y.

- The input x often contains many context-sensitive descriptors.
- The output y may be a sensor reading, a categorization, or a human decision.

Q: What if we cannot identify the true *causal* relationship? Are we still interested?

Yes, because – in many cases – being able to make accurate predictions is sufficient.

For predictions, we need only capture the correlation between x and y.

Statistical modelling

We may estimate functions from data through *statistical models* and *statistical inference*.

A statistical model details relationships between (random) variables and includes assumptions on the data-generating process.

- Example: suppose you go to a bridge overlooking a highway. The type of cars $\boldsymbol{y}$ you see depends on the time of day $\boldsymbol{x_1}$, where the highway is located $\boldsymbol{x_2}$, the weather $\boldsymbol{x_3}$, etc.

Statistical inference is the study of mathematical procedures for obtaining estimates of unobserved variables using data.

- Examples: *maximum likelihood* and *expectation-maximization*.

Statistical modelling

Typically, statistical modelling involves the following steps:

1. Collect a set of observations of input-output pairs.
2. “Fit” one or more statistical models to “training data”.
3. Evaluate the models on a “validation data” set and select the best performing one.
4. Deploy the model to make output predictions for future “test data”.

Regression

Regression **refers to statistical models for continous-valued variables y.**

The equation structure for a regression model is simple:

$$y = f(x) + \varepsilon$$

The output y is a function f of the input x plus noise ε.

- The input x is known as the *independent variable,* or *covariate.*
- The output y is known as the *dependent variable, outcome,* or *response.*

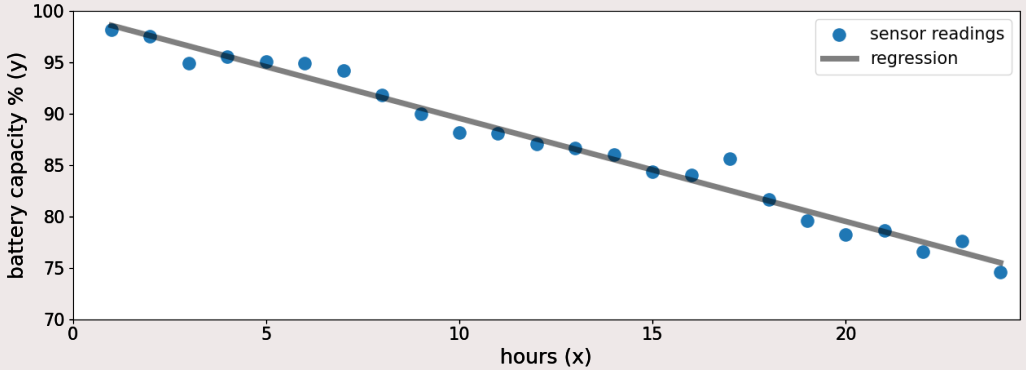
The in-/outputs can be vectors of different dimensions (c.f. SISO, MISO, SIMO, MIMO).

Linear regression

A linear regression model is defined as follows:

$$f_\theta(x) = \theta_0 + \theta_1 x$$

The parameter θ_1 is called the *slope coefficient* and θ_0 the *intercept*.



Linear regression

In the multivariate input case, i.e., $\mathbb{R}^D \rightarrow \mathbb{R}$, the regression function becomes:

$$f_\theta(x_1, x_2, \dots, x_D) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_D x_D$$
$$= \theta_0 + \sum_{d=1}^{D} \theta_d x_d$$

where x_d refers to the d-th dimension or element of the feature vector.

The data variable can be augmented, $\tilde{x} = [1 \;\; x]^T$, which allows for absorbing the intercept into the inner product:

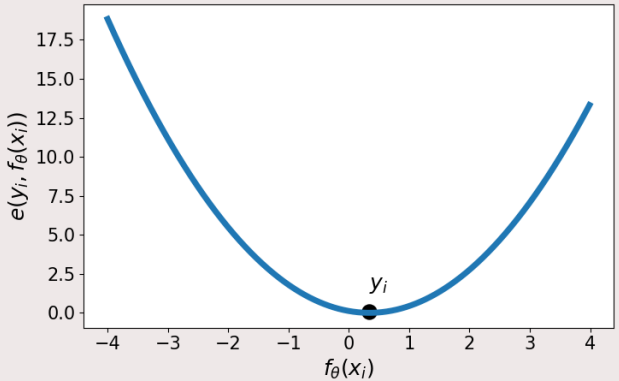
$$f_\theta(x) = \theta^T \tilde{x}$$

Least-squares

The *error* is the squared difference between the i-th response and the i-th prediction:

$$\left(y_i - f_\theta(x_i)\right)^2$$

The more you deviate, the larger the error:



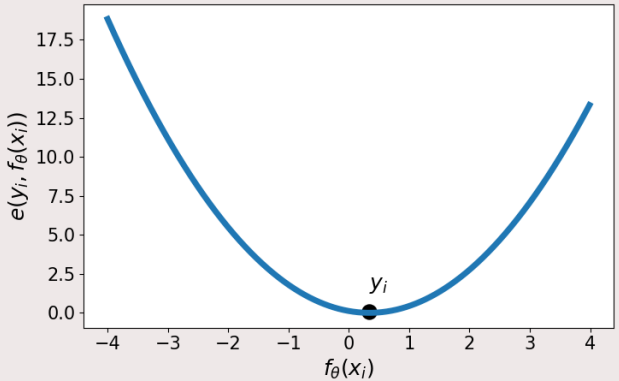
TU/e

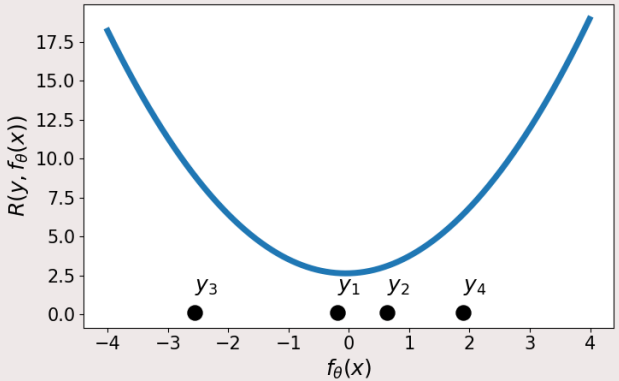
Least-squares

For N data points, $(X, Y) = \{(x_i, y_i)\}_{i=1}^{N}$, we average the errors:

$$R(f_\theta(X), Y) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - f_\theta(x_i)\right)^2$$

The mean squared error is still parabolic, but its minimum is no longer 0:





Least-squares

Finding the best regression function is a minimization problem:

$$\boldsymbol{\theta^* = \arg\min_{\theta \in \Theta} R(f_\theta(X), Y)}$$

where we minimize over the parameter space Θ of the regression function.

First, we have to take the derivative with respect to the parameters: $\frac{\partial}{\partial \theta} R(f_\theta(X), Y)$.

For linear regression, this is:

$$\frac{\partial}{\partial \theta} \frac{1}{N} \sum_{i=1}^{N} (y_i - \theta^T \tilde{x}_i)^2 = \frac{1}{N} \sum_{i=1}^{N} 2(y_i - \theta^T \tilde{x}_i)(-\tilde{x}_i^T)$$

Least-squares

Setting the derivative to 0 yields an analytic solution:

$$\frac{1}{N}\sum_{i=1}^{N} 2\left(y_i - \theta^T \tilde{x}_i\right)\left(-\tilde{x}_i^T\right) = 0$$

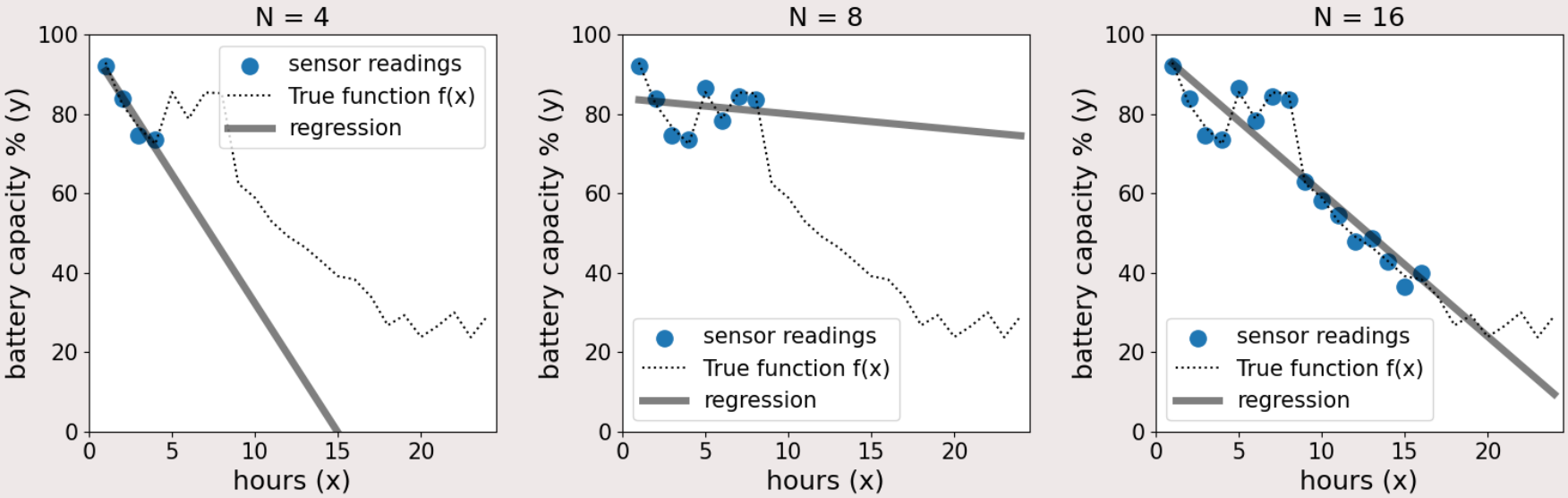
$$\frac{1}{N}\sum_{i=1}^{N} 2y_i\tilde{x}_i^T = \frac{1}{N}\sum_{i=1}^{N} 2\theta^T\tilde{x}_i\tilde{x}_i^T$$

$$\left(\sum_{i=1}^{N} \tilde{x}_i\tilde{x}_i^T\right)^{-1}\left(\sum_{i=1}^{N} y_i\tilde{x}_i\right) = \theta$$

In the limit of $N \to \infty$, this estimator will converge to the true coefficients*.

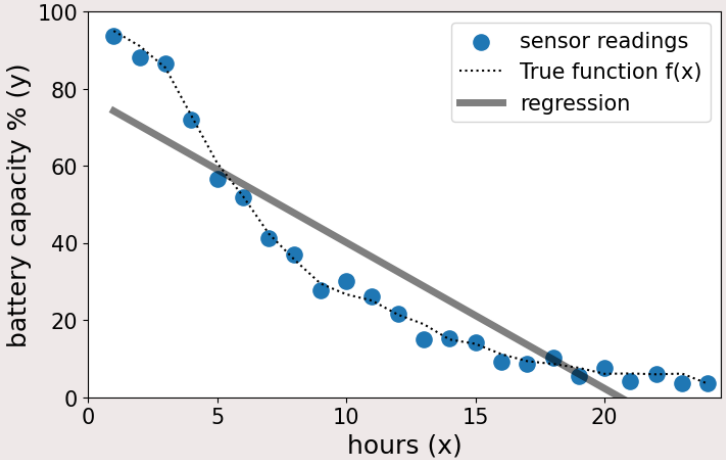
*https://www.statlect.com/fundamentals-of-statistics/OLS-estimator-properties

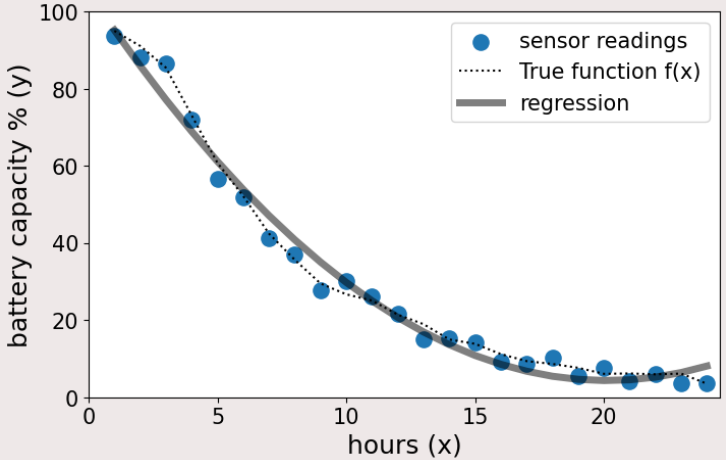
Linear regression



Curve fitting

Q: What if you want to fit a curve instead of a line?





We can just define a different type of regression function.

Polynomial regression

If we define a polynomial basis expansion, $\varphi: \mathbb{R} \rightarrow \mathbb{R}^{M+1}$;

$$\varphi(x) = [1 \quad x \quad x^2 \quad \dots \quad x^M]$$

Then we can write a polynomial regression function as:

$$f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots \theta_M x^M = \theta^T \varphi(x)$$

This holds for multi-variable polynomials as well. For example:

$$\varphi(x_1, x_2) = [1 \quad x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad x_1 x_2 \quad x_1^2 x_2 \quad x_1 x_2^2 \quad x_1^3 \quad x_2^3 \quad \dots]$$

Polynomial regression

Finding the optimal parameters for a polynomial regression problem is still:

$$\boldsymbol{\theta^* = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \theta^T \varphi(x_i) \right)^2}$$

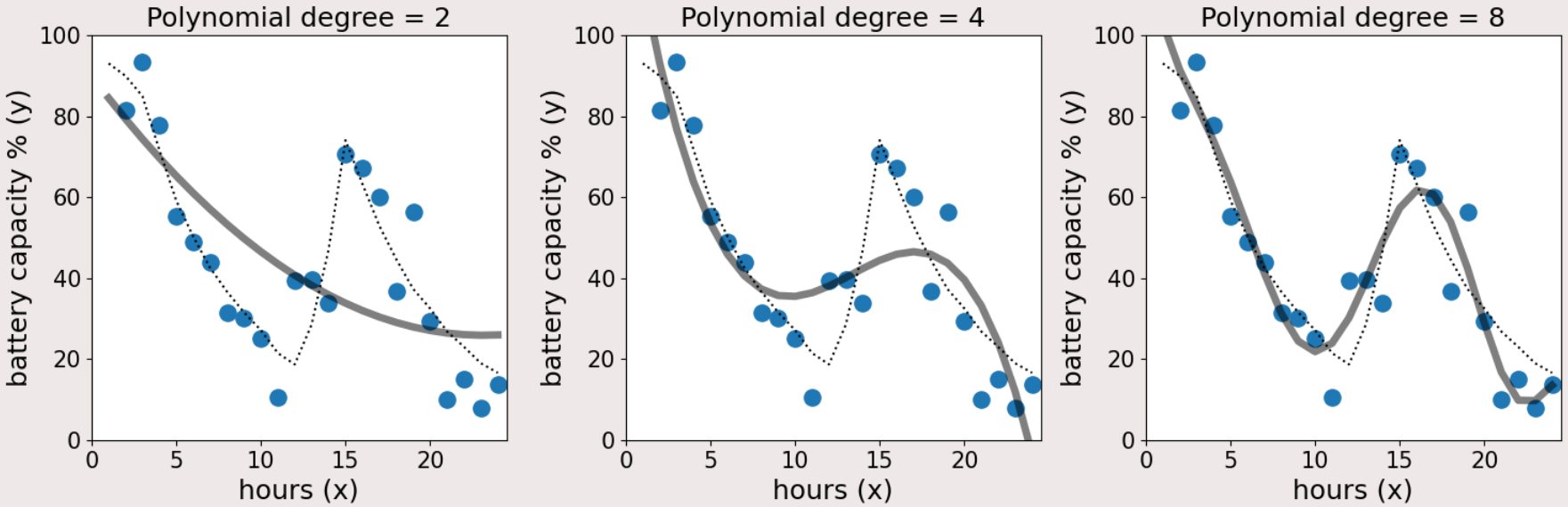
Note that polynomial regression models are still linear in their parameters.

This implies that we can obtain a similar analytic solution to the minimization problem:

$$\boldsymbol{\theta^* = \left(\sum_{i=1}^{N} \varphi(x_i) \varphi(x_i)^T \right)^{-1} \left(\sum_{i=1}^{N} y_i \varphi(x_i) \right)}$$

Polynomial regression

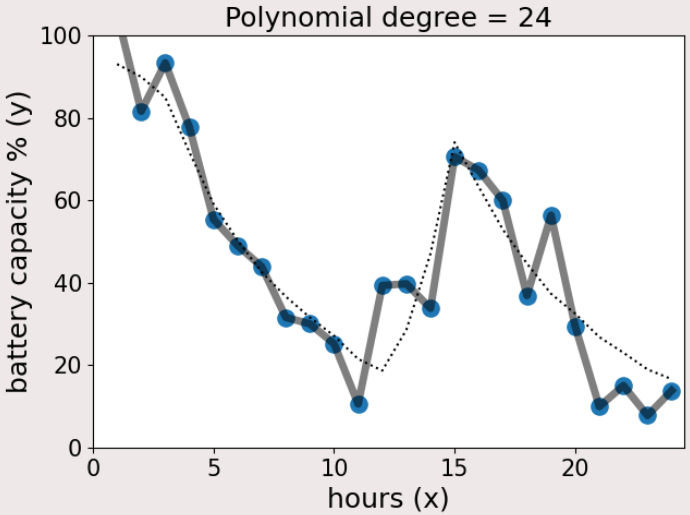
Suppose you recharged the battery halfway and applied various polynomials:



Polynomial regression

If we *can* fit highly complex functions, then why don't we just always do that?

Q: What would happen if I fit a 24-th order polynomial to our data set with $N = 24$?



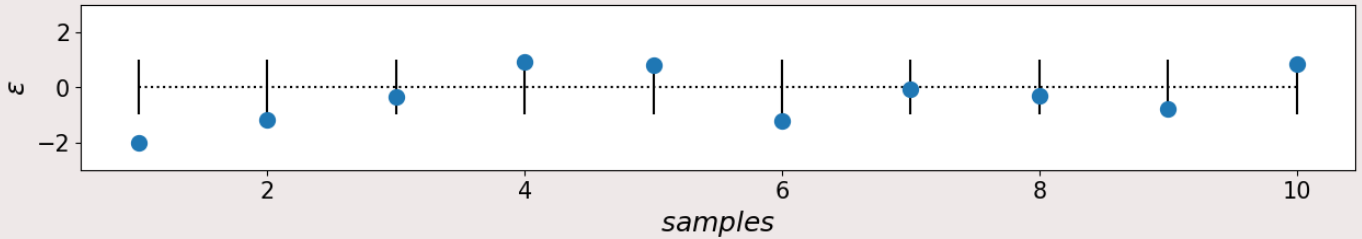
- **Least-squares estimators may fit to noise.**
- **This is typically visually discovered as the regression function being too highly-varying to capture a few points.**
- **This problem is known as *over-fitting*.**

TU/e

Fitting to noise

Least-squares estimators over-fit because they are not aware of the noise ε.

Noise manifests itself as seemingly "random" perturbations or deviations from $f(x)$.



But these perturbations are just a collection* of unaccounted factors.

- Examples: gusts of wind affecting a drone's flight path, discretization error on a sensor.

*https://www.statlect.com/asymptotic-theory/central-limit-theorem

Fitting to noise

Our function does not have to pass through each observation *exactly.*

Q: How do we inform the model of this desire?

It is possible to push / incentivize an optimizer to find solutions we favour *a priori.*

- For example, smaller coefficients produce curves that vary more slowly.
- We can express the total size of the coefficients with an *p-norm*:

$$\|\boldsymbol{\theta}\|_p = \sqrt[p]{\sum_{m=1}^{M} \theta_m^p}$$

Regularization

If we add the squared 2-norm of the parameters to our regression objective function,

$$R(f_\theta(X), Y) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \theta^T\varphi(x_i)\right)^2 + \lambda\|\theta\|_2^2,$$

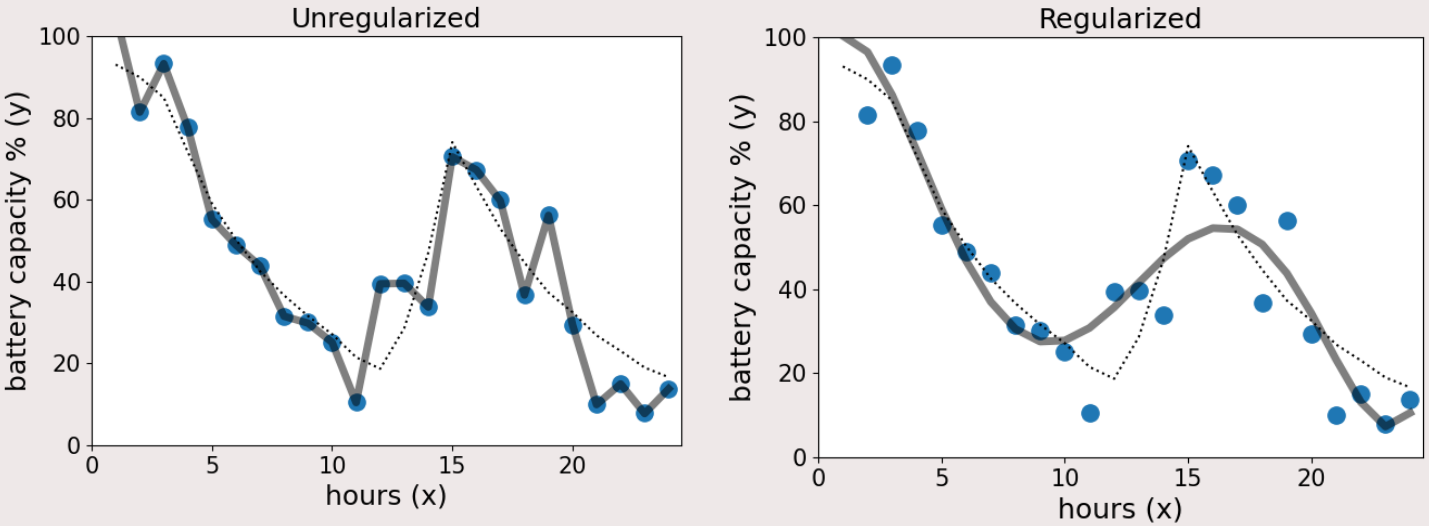
then the regression model is called "L_2-regularized regression" or "ridge regression".

If we go through the minimization derivations again, we obtain:

$$\theta^* = \left(\lambda I + \sum_{i=1}^{N}\varphi(x_i)\varphi(x_i)^T\right)^{-1}\left(\sum_{i=1}^{N} y_i\varphi(x_i)\right)$$

Regularization

Regularization can combat over-fitting and produce favoured solutions.

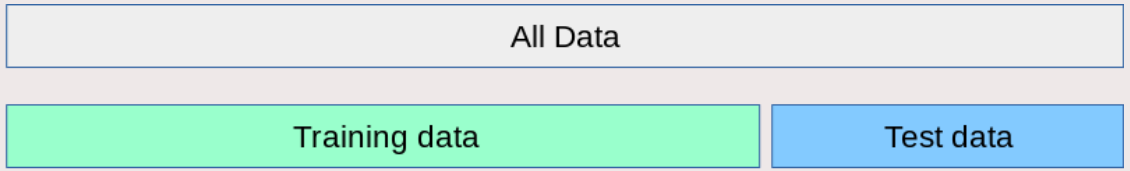


Model validation

Q: How do we determine the regularization parameter λ?

We know that the model has over-fitted when its performance on new data is worse than its performance on already seen data.

- We can split our data, train on a subset and evaluate on a different *disjoint* subset.



- We can then select the λ which performs best.

Figure taken from: https://scikit-learn.org/stable/modules/cross_validation.html

Model validation

However, there is still a problem: now the model will over-fit to the test set.

The appropriate way of validating the model (and selecting the best hyperparameters) is by splitting the data multiple times into training and validation *folds:*

- This is known as *k-fold cross-validation.*
- The optimal cross-validation procedure is for $k = N$, a.k.a. *leave-one-out cross-validation.*
- The final test is still based on a separate test data set.

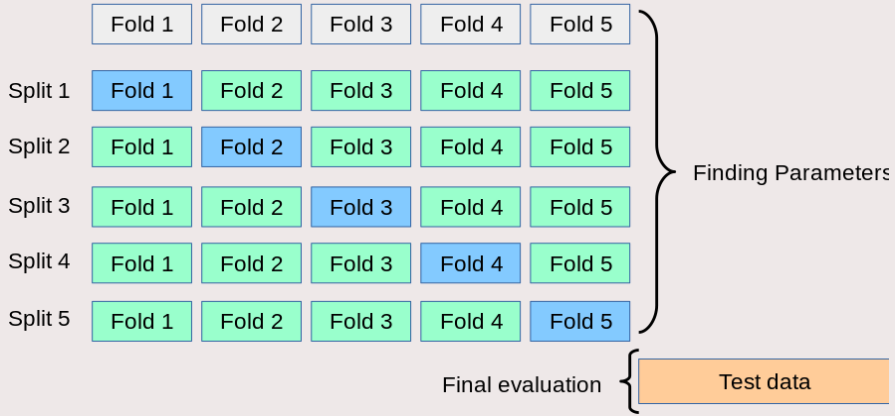
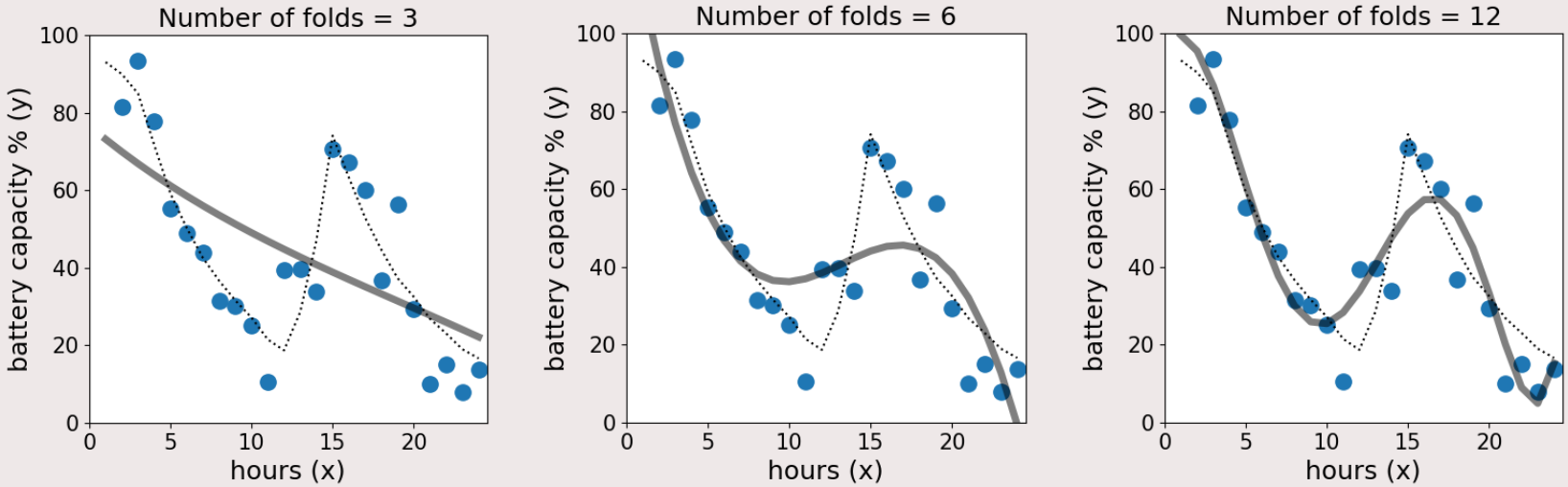


Figure taken from: https://scikit-learn.org/stable/modules/cross_validation.html

Model validation

The more folds you use, the better the regularization parameter estimate.



TU/e

Summary regression

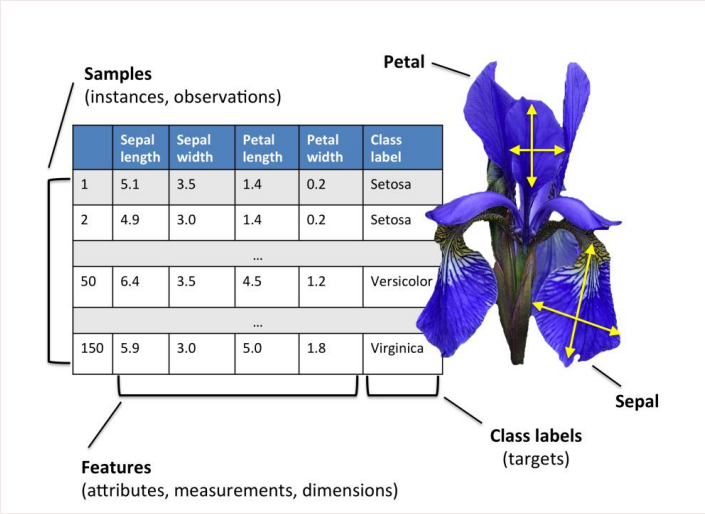
Regression models find functions from inputs x to continuous responses y.

- *Least-squares:* using regression functions that are linear in their parameters and adopting a squared error, we find an analytic solution for the optimal parameters.
- *Over-fitting:* least-squares estimators may fit to noise in the training data when the regression function is complex/flexible relative to the number of data points.
- *Regularization:* a family of procedures which allows us to favour certain solutions *a priori*, such as L_2-regularization producing curves that vary more slowly in time.
- *Cross-validation:* a set of procedures for estimating model hyperparameters (e.g., regularization parameter) based on splitting the data set and repeated validation.

Break

Classification

Suppose we have a data set of flowers labelled by species:



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

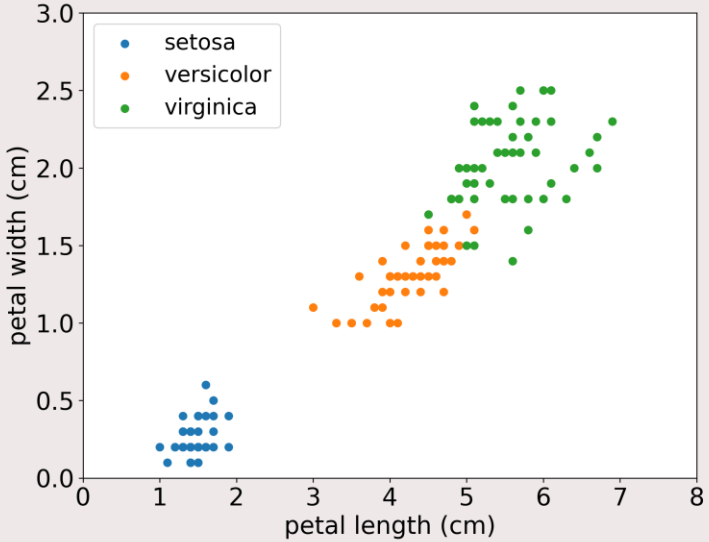
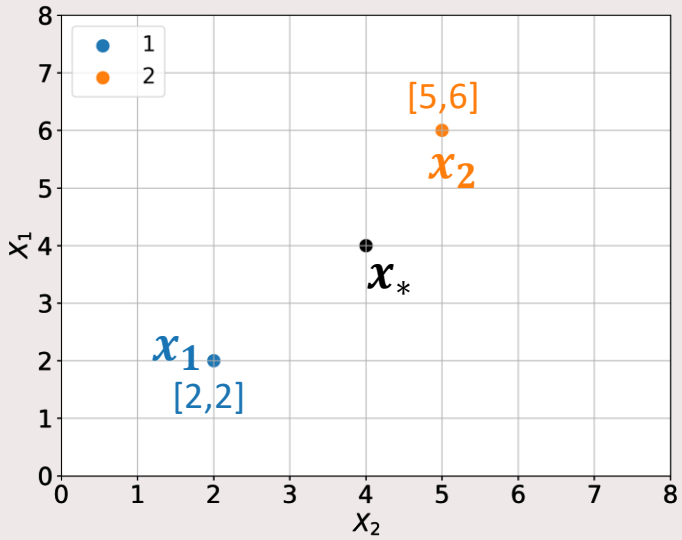


Figure from https://sebastianraschka.com/Articles/2015_pca_in_3_steps.html

Nearest Neighbours

Let's start with one the oldest algorithms in history: nearest neighbours*.

To explain, we'll start with a simpler data set with just 2 labeled samples:



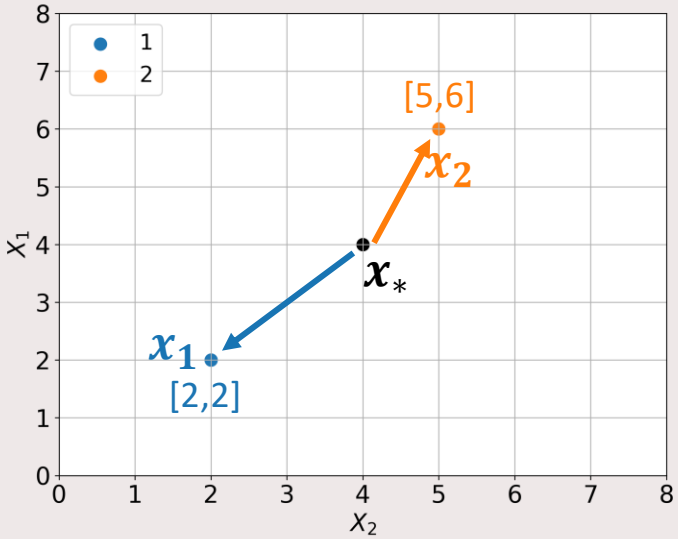
Now suppose we measure a new point.

This new point can be classified by assigning it the label of its nearest neighbour.

*Alhazen and the nearest neighbor rule (https://doi.org/10.1016/j.patrec.2013.10.022).

Nearest Neighbours

To quantify "nearest", we compute Euclidean distances between points.



$$d(x_1, x_*) = \sqrt{\sum_{d=1}^{D} (x_{1d} - x_{*d})^2} = \sqrt{8}$$

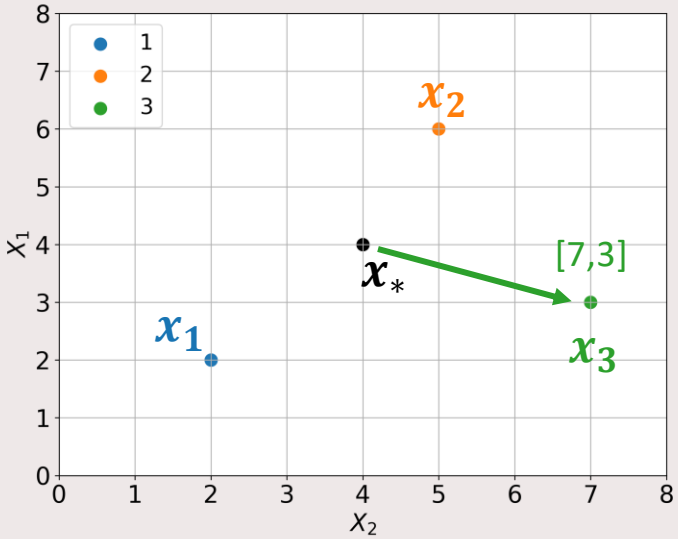
$$d(x_2, x_*) = \sqrt{\sum_{d=1}^{D} (x_{2d} - x_{*d})^2} = \sqrt{5}$$

x_* is classified as class 2, since x_2 is closer.

TU/e

Nearest Neighbours

Nearest neighbour naturally extends to as many classes as there are in the data set:



Euclidean distances:

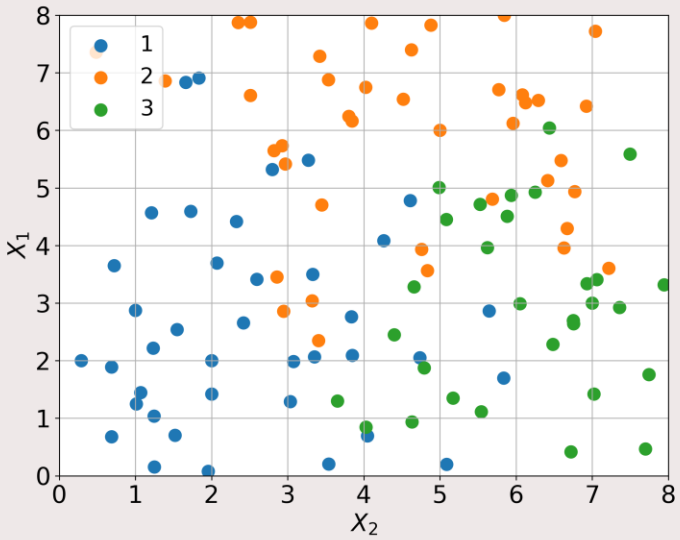
$$d(x_1, x_*) = \sqrt{8}$$
$$d(x_2, x_*) = \sqrt{5}$$
$$d(x_3, x_*) = \sqrt{10}$$

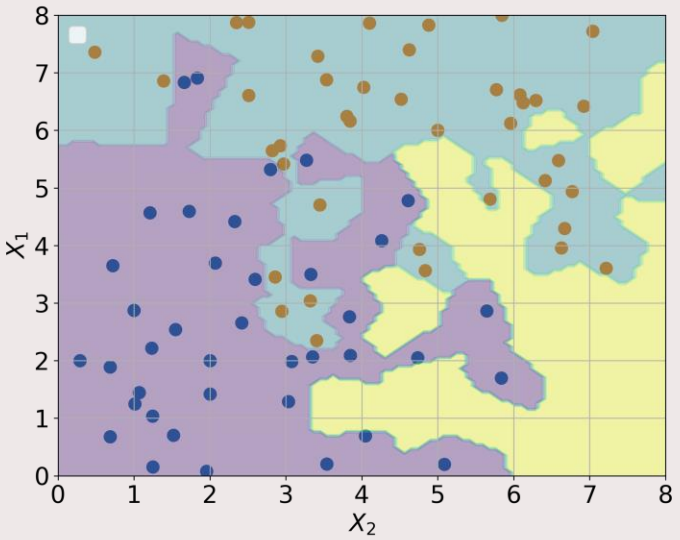
x_* is classified as class 2, since x_2 is still closer.

Nearest Neighbours

What about this data set? See any problems?

1-Nearest Neighbours is going to give a messy decision boundary -> overfitting.

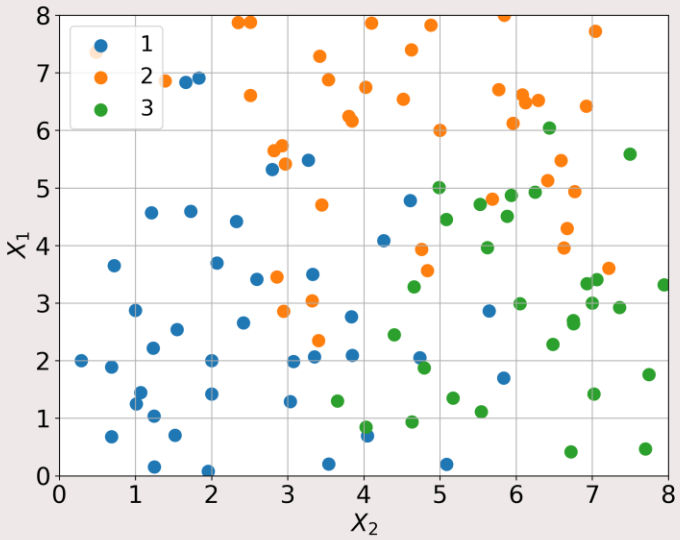


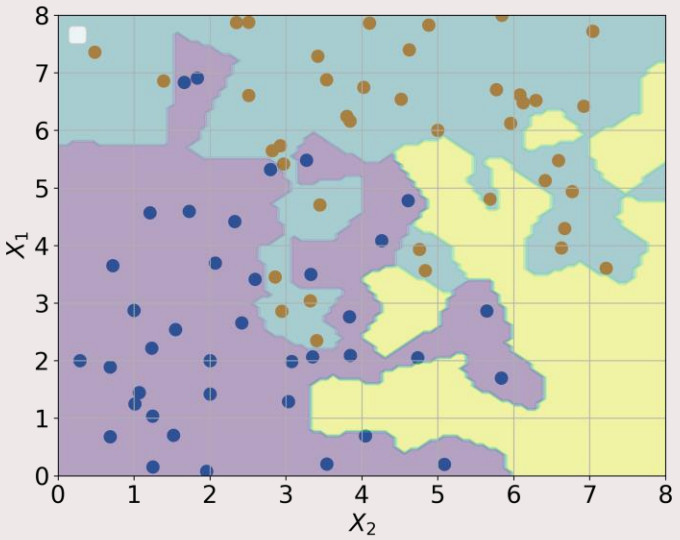


k-Nearest Neighbours

We can increase the number of neighbours used for the class assignment.

For k = 1:

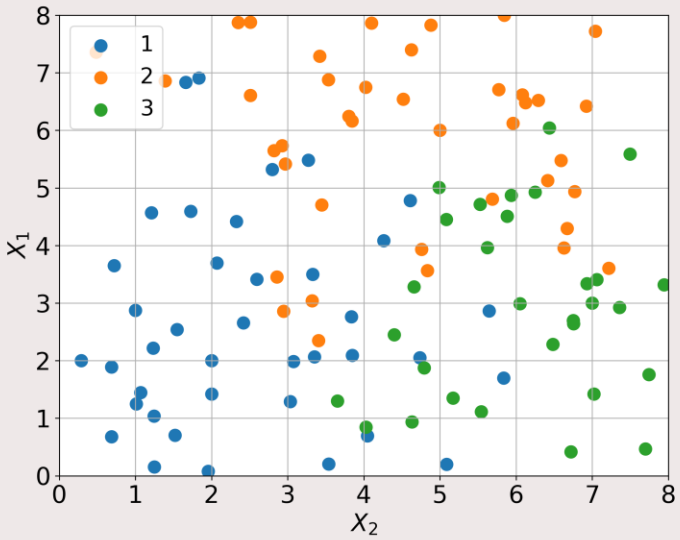


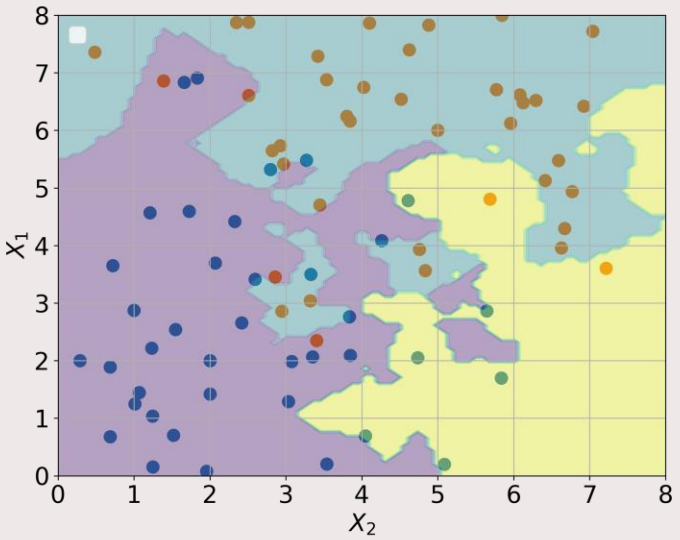


k-Nearest Neighbours

We can increase the number of neighbours used for the class assignment.

For k = 3:

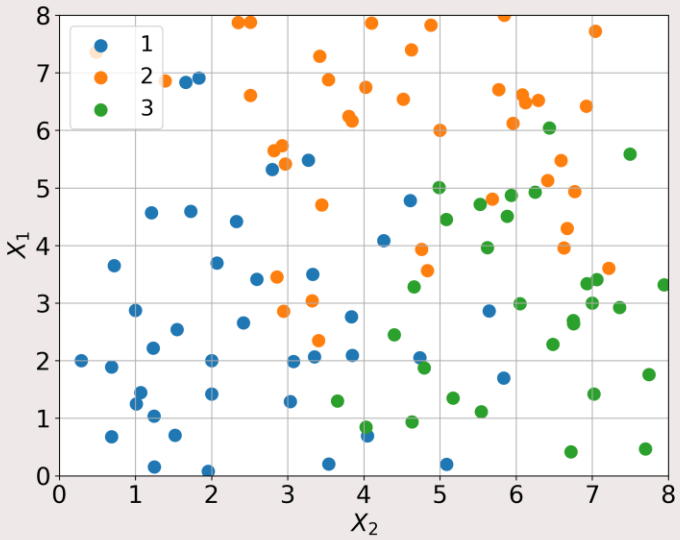


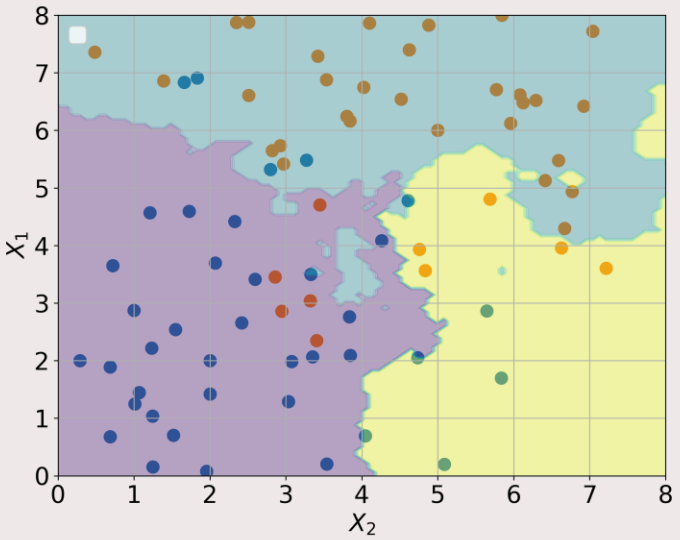


k-Nearest Neighbours

We can increase the number of neighbours used for the class assignment.

For k = 9:

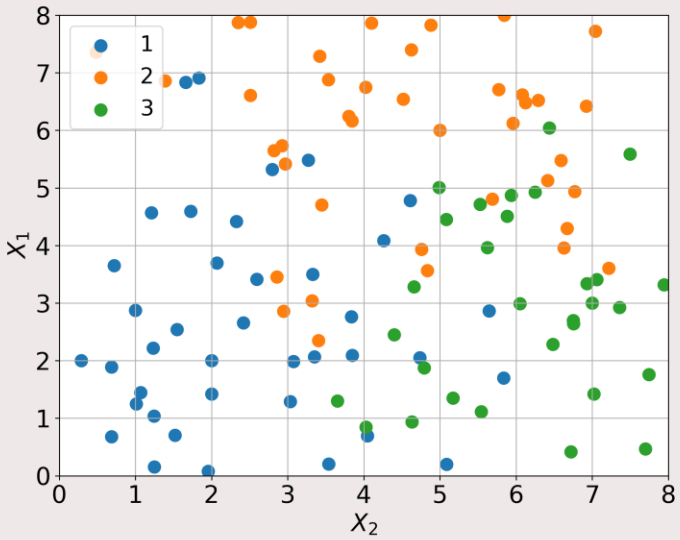


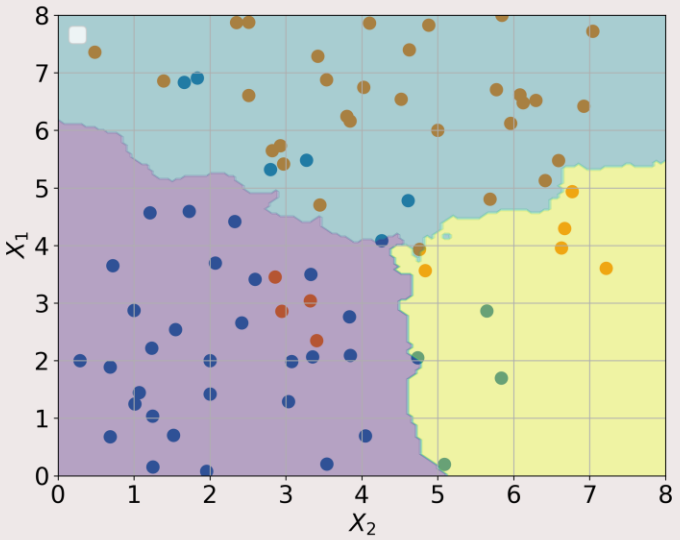


k-Nearest Neighbours

We can increase the number of neighbours used for the class assignment.

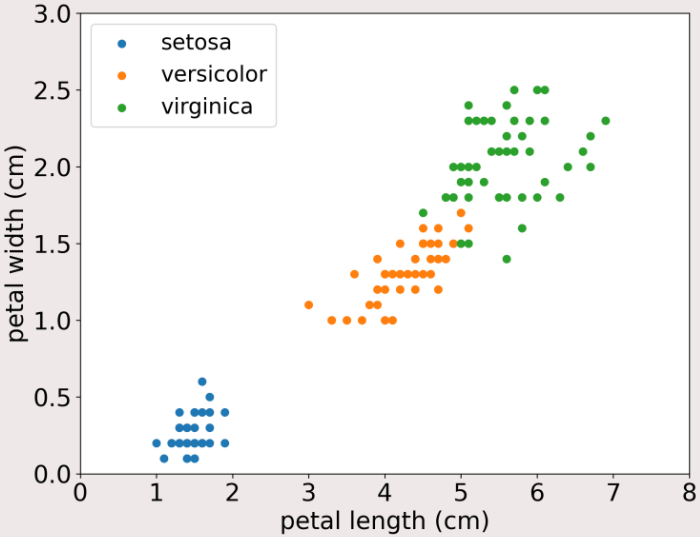
For k = 30:

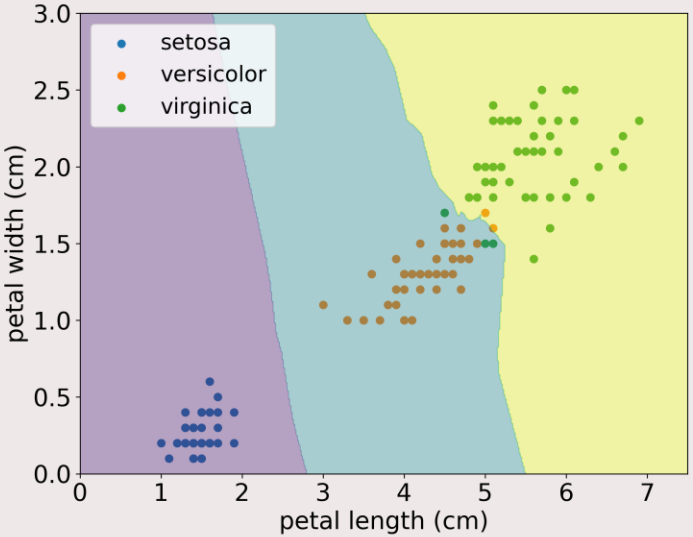




k-Nearest Neighbours

Now to return to the Iris data set, with a $k = 5$ nearest neighbour classifier:





Limitations of k-Nearest Neighbours

Q: What are the major limitations of k-NN?

1. You have to compute the distance to all other points.

- To classify M new points using N labeled points, we need $\boldsymbol{O(NM)}$ computations.

2. All labeled points have to be kept in memory for use in later classification.

- Memory occupation grows $\boldsymbol{O(ND)}$ for D features per sample.

3. It may be unclear how many neighbours to use for a given problem setting.

Classification

We can tackle classification with statistical models using discrete-valued variables y.

The equation structure for a classification model is the same as for regression:

$$y = f(x) + \varepsilon$$

- The input x is now known as a *sample*, *feature vector* or *descriptor.*
- The output y is known as the *label, class,* or *target.*

The difference is how we compare the prediction $f(x)$ to the label y.

Logistic regression

In logistic regression, we think in terms of *class probabilities*.

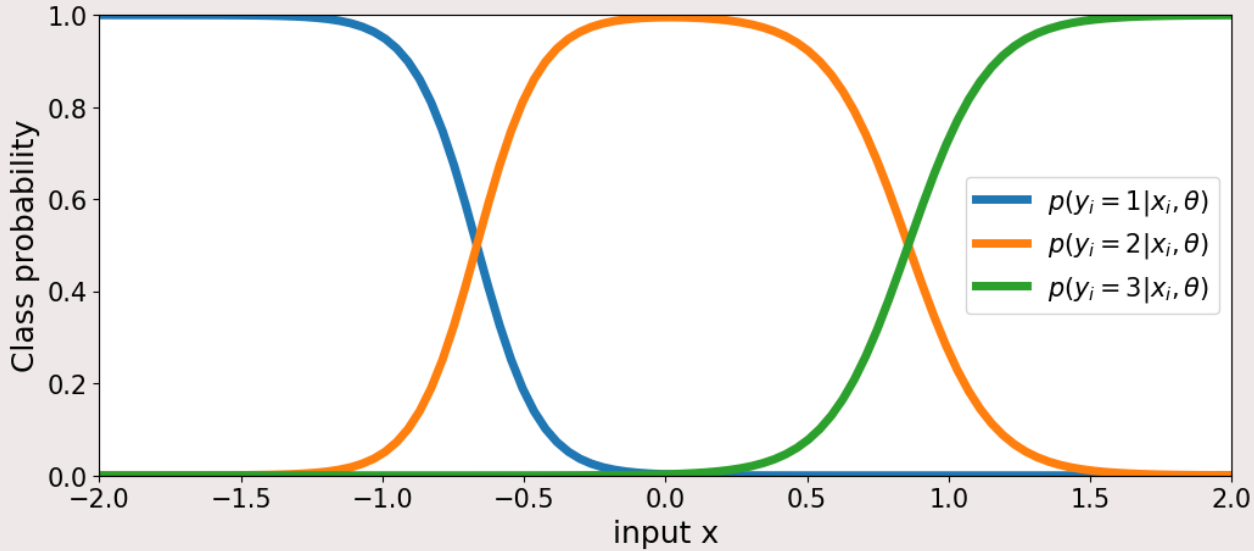
- When $y_i = k$, the probability $p(y_i = k \mid x_i)$ should be larger than $p(y_i \neq k \mid x_i)$.
- Using a linear function for each class, $\theta_k^T x$, we express the class probability as:

$$p(y = k \mid x, \theta) = \frac{\exp(\theta_k^T x)}{\sum_{k=1}^{K} \exp(\theta_k^T x)}$$

where $\theta = (\theta_1, \theta_2, \ldots, \theta_K)$ for K classes.

Logistic regression

Q: What does that class probability look like?



Logistic regression

We can apply these class probabilities to our data points with:

$$p(y_i \mid x_i, \theta) = \frac{\exp(\theta_{y_i}^T x_i)}{\sum_{k=1}^{K} \exp(\theta_k^T x_i)}$$

Note that this function is actually a likelihood, since the y_i is an observed variable.

If we have *iid* data, then we may write the likelihood of the full data set (X, Y) as:

$$p(Y|X, \theta) = \prod_{i=1}^{N} p(y_i \mid x_i, \theta)$$

Logistic regression

We can train a logistic regression model by maximum likelihood:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \boldsymbol{p}(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta})$$

For numerical reasons, we often minimize the negative log-likelihood instead:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} -\log \boldsymbol{p}(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta})$$

$$= \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{N}} \boldsymbol{\theta}_{\boldsymbol{y}_i}^{\boldsymbol{T}} \boldsymbol{x}_{\boldsymbol{i}} + \log\left(\sum_{\boldsymbol{k}=\boldsymbol{1}}^{\boldsymbol{K}} \exp\left(\boldsymbol{\theta}_{\boldsymbol{k}}^{\boldsymbol{T}} \boldsymbol{x}_{\boldsymbol{i}}\right)\right)$$

Logistic regression

We are interested in the gradient with respect to all class parameters:

$$\nabla_\theta = \left(\frac{\partial}{\partial \theta_1} \quad \cdots \quad \frac{\partial}{\partial \theta_K} \right)$$

If we look at a specific partial derivative,

$$\frac{\partial}{\partial \theta_k} \sum_{i=1}^{N} \theta_{y_i}^T x_i + \log \left(\sum_{k=1}^{K} \exp(\theta_k^T x_i) \right) = \sum_{i=1}^{N} [y_i = k] \, x_i + \frac{\exp(\theta_k^T x_i) x_i^T}{\sum_{k=1}^{K} \exp(\theta_k^T x_i)}$$

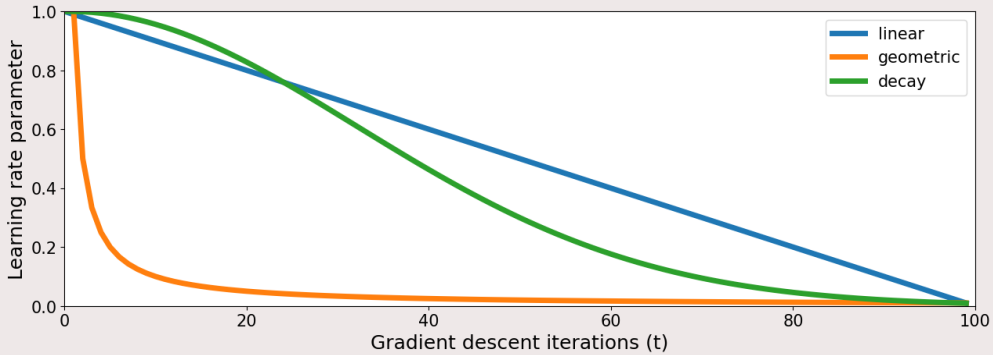
Then we realize that there is no analytic solution for this minimization problem.

Logistic regression

We will have to resort to procedures such as gradient descent:

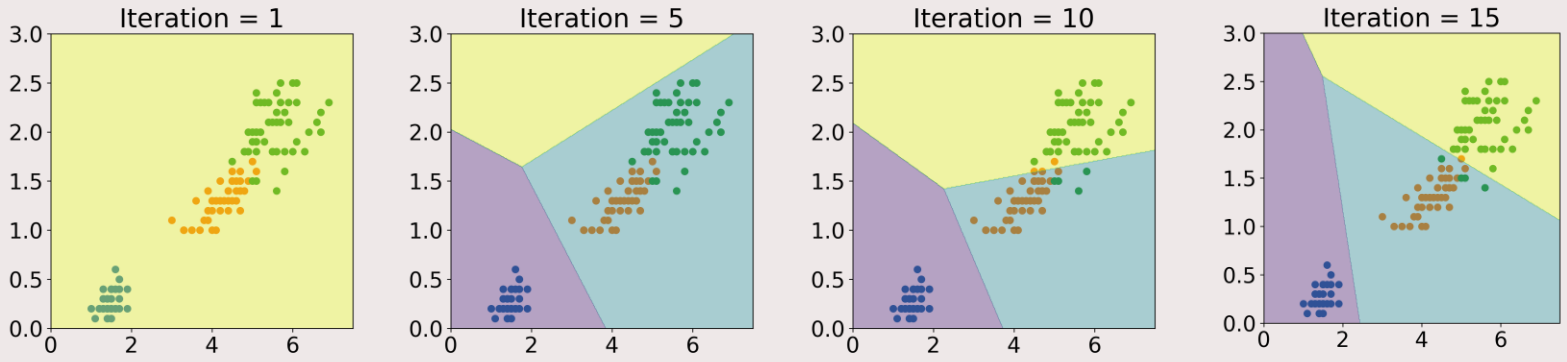
$$\theta_k^{t+1} = \theta_k^t - \gamma^t \frac{\partial(-\log p(Y|X,\theta))}{\partial \theta_k}\Big|_{\theta=\theta_k^t}$$

where γ^t is a "learning rate" parameter that shrinks over iterations t.



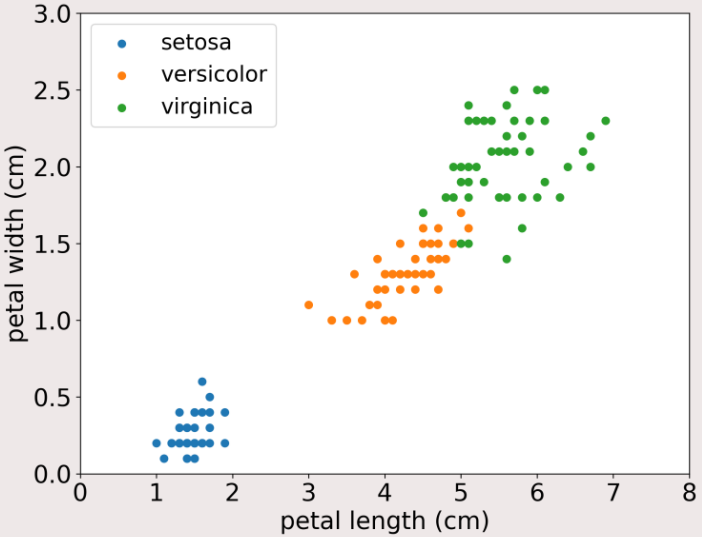
Logistic regression

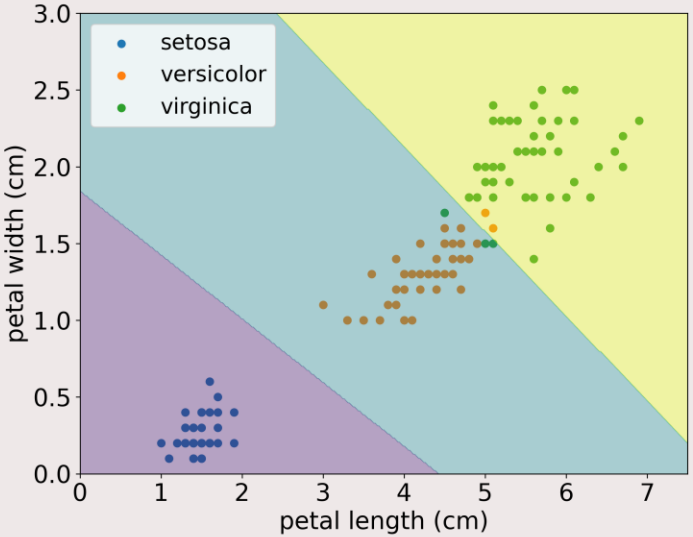
If we apply this procedure to the iris data set, we get:



Logistic regression

The final result makes intuitive sense:





Logistic regression

Q: What are the main limitations of logistic regression?

1. **Computing the gradient may be costly.**
2. **Gradient descent may require many iterations, e.g., 100.**
3. **Logistic regression still only incorporates linear classification functions.**

Ensembles

We can build a more complex statistical model by combining several simple ones.

The prediction of a set of trained regression models can be seen as a new feature.

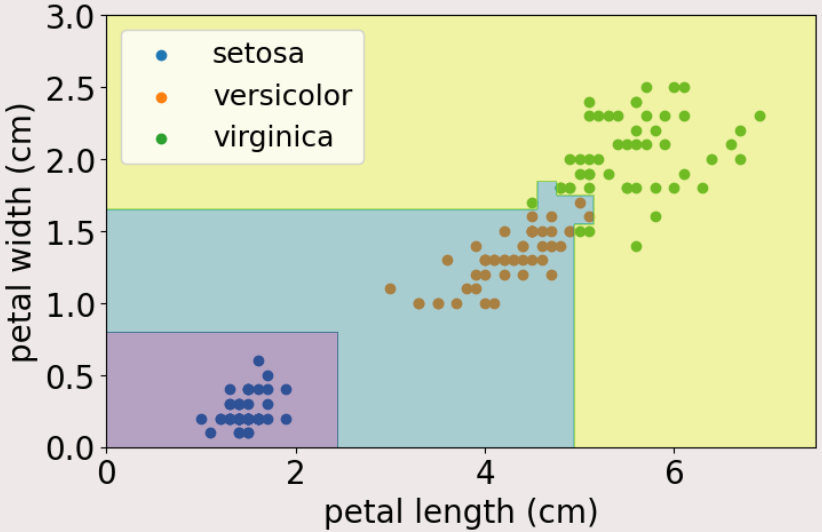
- We can find the optimal combination of trained regressors by least-squares.

Classification models can be combined through voting systems.

- If we have an odd number of trained classifiers making predictions, then we can use majority voting to decide a final prediction.

Ensembles

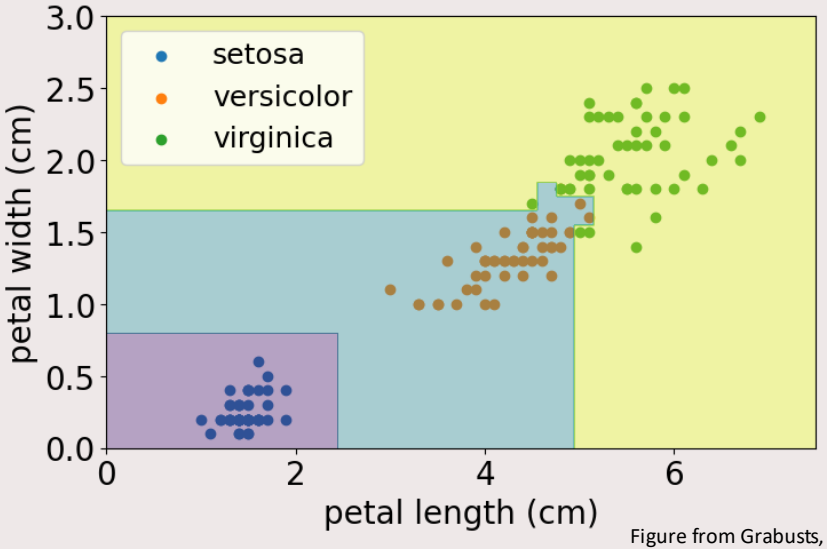
Boosting **is a sequential ensemble where simple classifiers are trained on misclassified data points of previous classifiers, only becoming more complex where needed.**



- **The class decision boundary is often just a threshold in parts of feature space that are "easy to classify".**
- **The decision boundary may be very flexible in parts of feature space that are "hard to classify".**

Ensembles

Decision trees **and** ***random forests*** **are ensembles that successively split the data points into classes sequentially along features.**



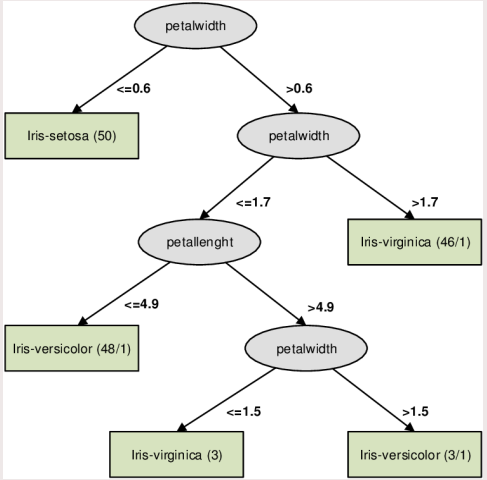
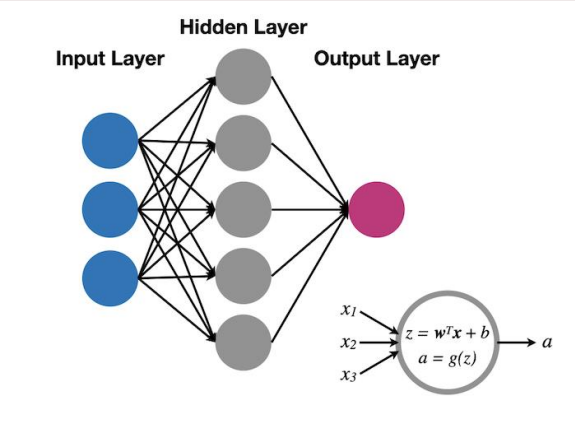


Figure from Grabusts, Borisov & Aleksejeva (2015). *Ontology-Based Classification System Development Methodology.*

Ensembles

We can apply the same basis expansion trick to our features before classification.

If the basis expansion is parameterized and the parameters are learned simultaneously, then we have what is called a *neural network* (technically also an ensemble).



- **The outputs of the basis functions are referred to as a "hidden layer".**
- **The derivative is passed through the basis expansions in order to update their parameters.**
- **This is known as *back-propagation*.**

Figure from https://scipython.com/blog/a-shallow-neural-network-for-simple-nonlinear-classification/

Summary classification

Classification models find functions from feature vectors x to discrete class labels y.

k-Nearest neighbours is a classic naturally multi-class non-linear classifier is based purely on distance to other data points.

- It is unfortunately costly in memory and computation time.

Logistic regression is a statistical model in which inference optimizes the class probabilities of the labeled data points.

- It is cheap in memory and computation, but only employs linear classification functions.

Nonlinear classification models are often constructed using ensembles of simpler models, such as majority voting, boosting, tree-based classifiers or neural networks.

Questions