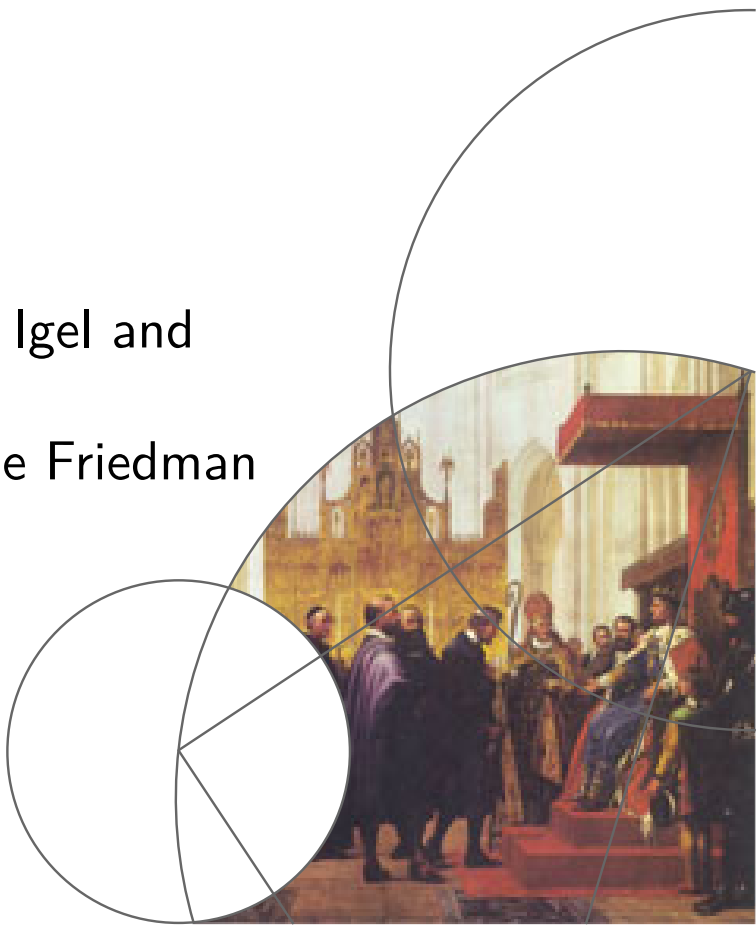
A quick introduction to machine learning concepts

Elements of machine learning

Jens Petersen

with content from Francois Lauze, Christian Igel and Elements of Statistical Learning by Trevor Hastie, Robert Tibshirani, and Jerome Friedman



About this lecture

- Basic concepts in machine learning
- Large parts may be overlapping with contents covered in courses
  - Data Science
  - Modelling and Analysis of Data (MAD)

What is machine learning?

What is machine learning?

The scientific field

> *Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead.*
>
> *wikipedia.org*

What is machine learning?

The scientific field

> *Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead.*
>
> *wikipedia.org*

Or the process:
Machines that learn from data, as opposed to being programmed, how to solve problems.

What is machine learning?

The scientific field

> *Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead.*
>
> *wikipedia.org*

Or the process:
Machines that learn from data, as opposed to being programmed, how to solve problems.

What do we mean by learning?

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E*
>
> *Michell, T. (1997). Machine Learning.*

ML succes stories

- Image classification (ImageNet, facial recognition, diagnosis from medical scans, ...)
- Image and video generation
- Language models - Text generation, Machine translation
- Playing games (AlphaZero, AlphaGo, AlphaStar, ...)
- Speech recognition
- Handwriting recognition

ML succes stories

- Image classification (ImageNet, facial recognition, diagnosis from medical scans, ...)
- Image and video generation
- Language models - Text generation, Machine translation
- Playing games (AlphaZero, AlphaGo, AlphaStar, ...)
- Speech recognition
- Handwriting recognition

What do these tasks have in common?

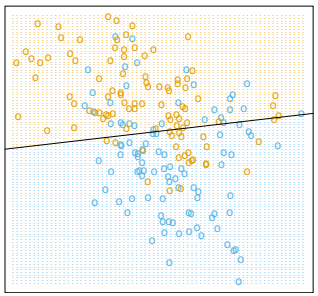
When to use ML?

When the task

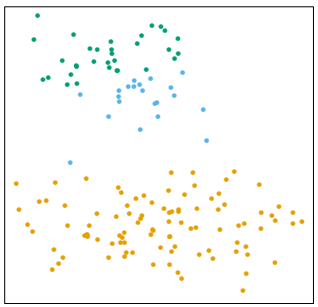
- is repetitive (machines do not tire and lose focus).
- is hard to algorithmically solve
- has example data to learn from
- performance can be measured

Types of learning

Supervised - learning from given examples of input and output (Linear regression of two classes with decision boundary)

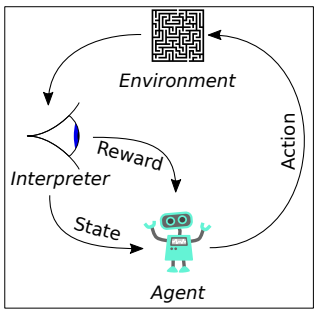


Unsupervised - learning from given examples of input alone (KMeans clustering with three clusters)



1

Reinforcement - learning how to take actions in a changing environment to maximize a notion of reward. (Outside the scope of this course.)



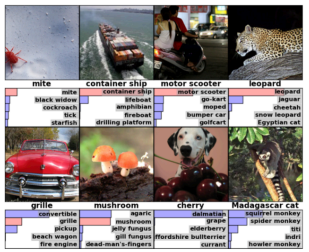
1

[1] By Megajuice, CC0, https://commons.wikimedia.org/w/index.php?curid=57895741

Types of supervised learning

Classification

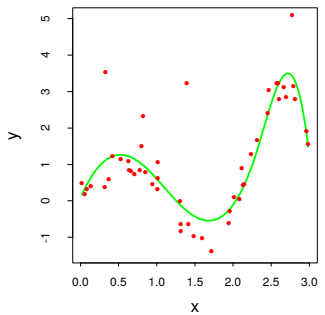
- Output variables are categorical / factors $Y = \{y_1, ..., y_n\}, 2 \leq n < \infty$.
  - No explicit ordering
  - Examples: objects in images, ...



[1]

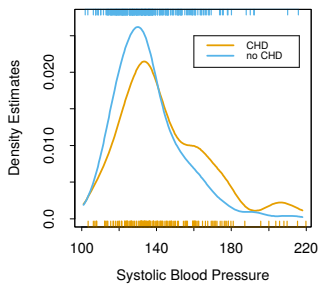
Regression

- Output variables are ordered / quantitative, $Y \subseteq \mathbb{R}^n, 1 \leq m < \infty$
  - Examples: disease severity, ...



Density estimation

- Output variables are probability distributions, $p(y|x)$ on Y



[1] Krizhevsky et al., Advances in neural information processing systems. 2012.

Examples of unsupervised learning

- Learning associations
  - example: $p(Y|X)$, customer bought X, what is the probability that customer also buys Y?
- Clustering
  - example: grouping similar objects
  - e.g. based on distance or (dis)similarities
  - with methods such as K-means, Gaussian Mixtures (soft K-means)
- Density estimation
  - example: where are objects likely to be?
- Dimensionality reduction
  - with methods such as principal Components, self-organizing maps, multidimensional Scaling

Learning machines

- A *hypothesis* $h : \mathcal{X} \to \mathcal{Y}$ maps inputs to outputs, a hypothesis class $\mathcal{H}$ is a set of such functions

  Example?

Learning machines

- A *hypothesis* $h : \mathcal{X} \to \mathcal{Y}$ maps inputs to outputs, a hypothesis class $\mathcal{H}$ is a set of such functions

  Example?

- Applying a learning algorithm means coming up with a hypothesis from the hypothesis class given sample data (formally, it maps from $\{((x_1, y_1), \ldots, (x_n, y_n)) \mid 1 \leq i \leq n < \infty, x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$ to $\mathcal{H}$).

Loss/risk functions

The quality of the prediction of a hypothesis is quantified by a *task-dependent* loss function.

A function $L : Y \times Y \to [0, \infty[$ with the property $L(y, y) = 0$ for $y \in Y$ is called a *loss function*.

Loss/risk functions

The quality of the prediction of a hypothesis is quantified by a *task-dependent* loss function.

A function $L : Y \times Y \to [0, \infty[$ with the property $L(y, y) = 0$ for $y \in Y$ is called a *loss function*.

Typical loss for classification is the 0-1 loss:

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

Goal of learning

Goal: Given sample data, we want to find a hypothesis h which minimizes the expected loss, also called the **risk**.

That is, we want to find h, such that

$$R(h) = \mathbb{E}\{L(y, h(x))\}$$

is minimized.

Goal of learning

Goal: Given sample data, we want to find a hypothesis h which minimizes the expected loss, also called the **risk**.

That is, we want to find h, such that

$$R(h) = \mathbb{E}\{L(y, h(x))\}$$

is minimized.

The *empirical risk* of h on sample data S ($|S| = n$) is defined as:

$$R_s(h) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i))$$

Typical loss functions

- Classification: Under the 0-1 loss $L(y, h(x)) = \mathbb{I}\{h(x) \neq y\}$ the risk of hypothesis h is the probability of error

$$R(h) = P(h(X) \neq Y) = \mathbb{E}\{\mathbb{I}\{h(X) \neq Y\}\}$$

and the empirical risk on sample data S ($|S| = n$) counts errors:

$$R_s(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{h(x_i) \neq y_i\}$$

(indicator function $\mathbb{I}\{\cdot\}$ is 1 if its argument is true and 0 otherwise)

Typical loss functions

- Classification: Under the 0-1 loss $L(y, h(x)) = \mathbb{I}\{h(x) \neq y\}$ the risk of hypothesis h is the probability of error

$$R(h) = P(h(X) \neq Y) = \mathbb{E}\{\mathbb{I}\{h(X) \neq Y\}\}$$

and the empirical risk on sample data S ($|S| = n$) counts errors:

$$R_s(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{h(x_i) \neq y_i\}$$

(indicator function $\mathbb{I}\{\cdot\}$ is 1 if its argument is true and 0 otherwise)

- Regression: Under the squared error

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

the empirical risk corresponds to the sum-of-squares error

Bayes risk and consistency

- Minimum risk over *all possible measurable functions* h is the Bayes risk

$$R^{\text{Bayes}} = R(h^{\text{Bayes}}) = \inf_h R(h)$$

  usually differs from $R^* = R(h^*) = \inf_{h \in \mathcal{H}} R(h)$

- Algorithm a is *consistent* if $\lim_{n \to \infty} R_s(a(S)) = R^{\text{Bayes}}$ almost surely

.

Bayes risk and consistency

- Minimum risk over *all possible measurable functions* h is the Bayes risk

$$R^{\text{Bayes}} = R(h^{\text{Bayes}}) = \inf_h R(h)$$

  usually differs from $R^* = R(h^*) = \inf_{h \in \mathcal{H}} R(h)$

- Algorithm a is *consistent* if $\lim_{n \to \infty} R_s(a(S)) = R^{\text{Bayes}}$ almost surely

Almost surely here means $P(lim_{n \to \infty} R_s(a(S)) = R^{\text{Bayes}}) = 1$.

Bayes risk and consistency

- Minimum risk over *all possible measurable functions* h is the Bayes risk

$$R^{\text{Bayes}} = R(h^{\text{Bayes}}) = \inf_h R(h)$$

  usually differs from $R^* = R(h^*) = \inf_{h \in \mathcal{H}} R(h)$

- Algorithm a is *consistent* if $\lim_{n \to \infty} R_s(a(S)) = R^{\text{Bayes}}$ almost surely

.

Some notes:

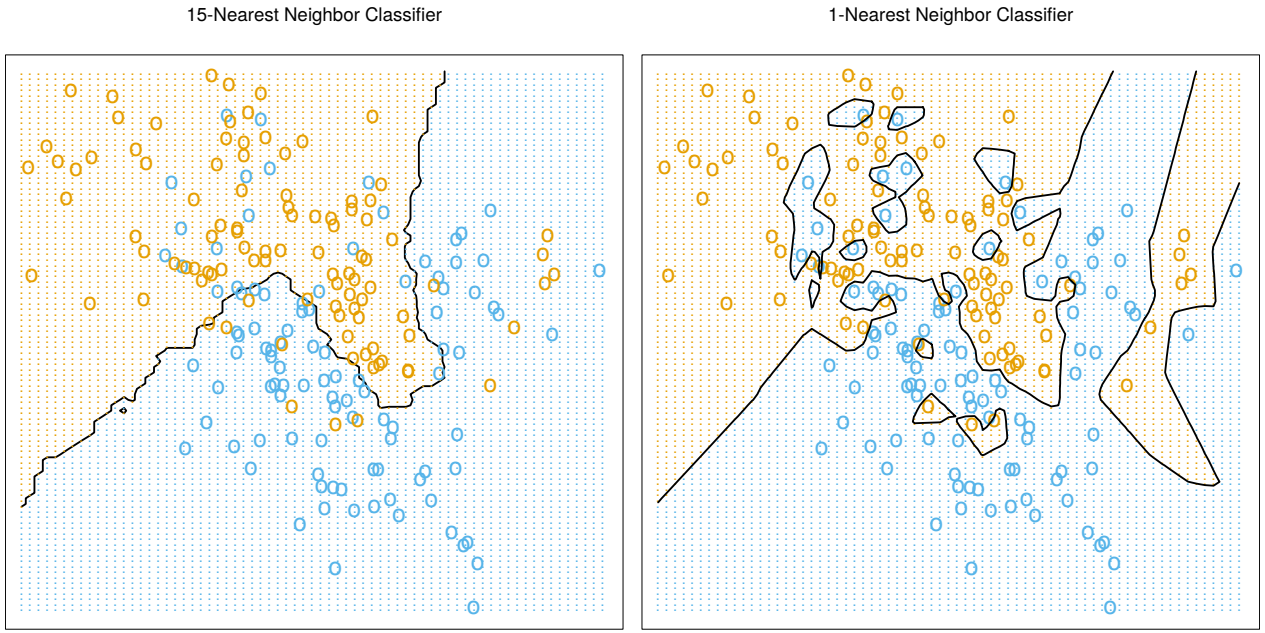
- you can think of the Bayes risk as the error an ideal model would make, that is, a model of the underlying true distribution.
- the learning process should ideally allow us to arrive at a model with errors that approaches the Bayes risk as the data set size increases.

Overfitting

Lets look at an example, are you all familiar with the kNN classifier?

Overfitting

We sample some data $S = \{x_i, y_i\}$, $i = 1, \ldots n$ from X with known labels $y_i \in \{\text{yellow}, \text{blue}\}$, $y_i \in Y$, and build a k-nearest neighbour classifier with k equal to 15 and 1.



Which classifier has

- the lowest empirical risk (average loss on 'training' sample S)?
- the lowest loss on some new sample $(x, y) \in (X, Y)$ that is $\notin S$?

Overfitting

Why do we see overfitting?

- Model is too complex/flexible
- The data has noise, errors, etc. that are irrellevant to the prediction

Overfitting

Why do we see overfitting?

- Model is too complex/flexible
- The data has noise, errors, etc. that are irrellevant to the prediction

How do we test if our model is overfitted?

Generalization error

We can measure the **generalization error** or **test loss**, the error over an independent test sample as

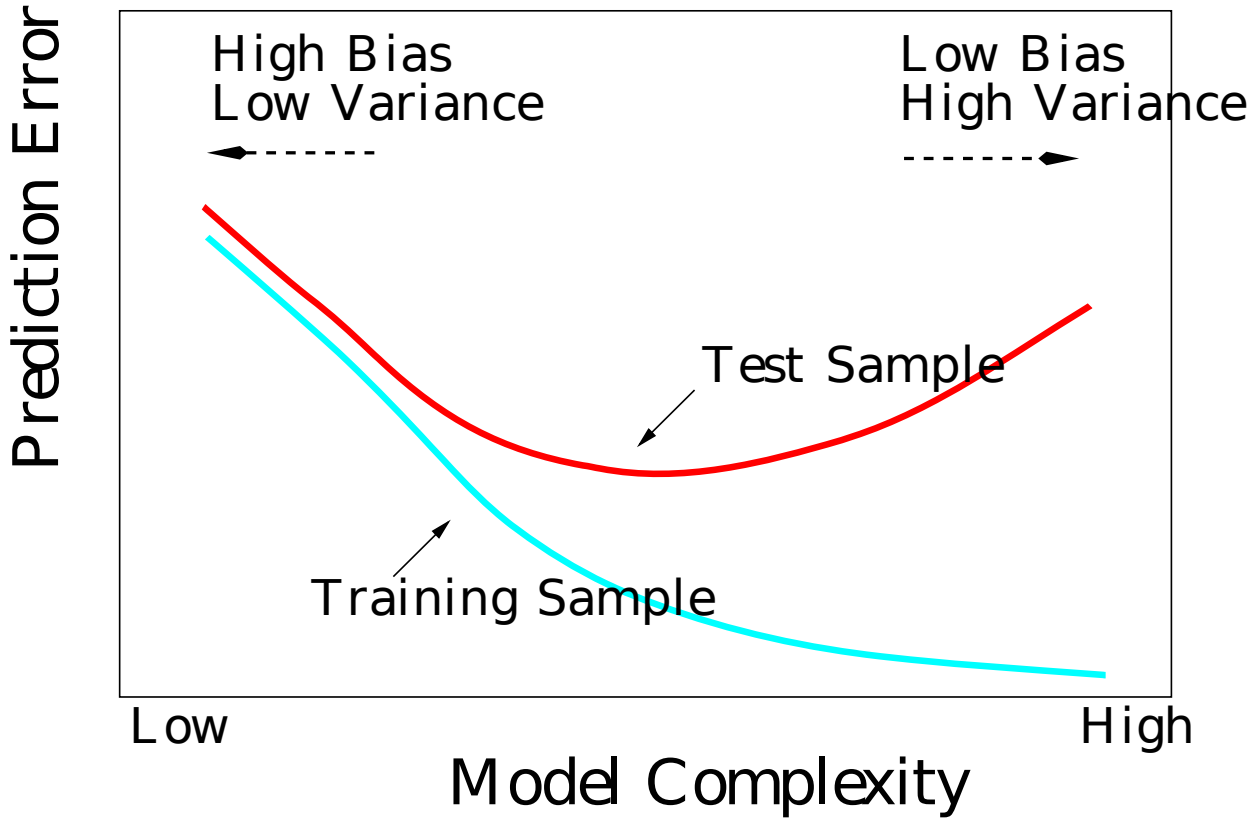
$$Err_\tau = \mathbb{E}\{L(Y, \hat{f}(X))|\tau\},$$

where τ represents the training set, and X and Y are the test set data and labels, drawn randomly from their joint distribution.

If the model is overfitting, then we would expect $Err_\tau > \bar{err}$, where $\bar{err}$ is the average loss of the training sample, **training loss**.

$$\bar{err} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

Prediction error vs model complexity



Bias-variance tradeoff

- Bias - the difference between the average prediction and the correct value
  - Error arising due to wrong model assumptions (e.g. arising due to a model that is not complex enough)
- Variance - the variability of predictions.
  - Models with too high variance, pay too much attention to each training data point and will therefore perform worse on test data.

Tradeoff - making a model more complex leads to lower bias but more variance.

Bias-variance decomposition

Given a model $Y = f(X) + \epsilon$, where the noise has zero mean ($E(\epsilon) = 0$) and variance given by $Var(\epsilon) = \sigma^2$.

Suppose we want to minimize the mean squared error of our prediction $\hat{f}$:

$$\begin{aligned} E((y - \hat{f}(x))^2) &= \sigma^2_{epsilon} + Bias^2(\hat{f}(x)) + Var(\hat{f}(x)) \\ &= \text{Irreducible Error} + \text{Bias}^2 + Variance \end{aligned}$$

Model selection and assessment

How do we choose k in kNN? Or more generally, suppose we have a number of models, some simple some complex, how do we determine which to use?

We cannot optimize k (or more generally model complexity) directly on the training set, as the more complex model (the model with most variance) will always perform better.

Model selection and assessment

How do we choose k in kNN? Or more generally, suppose we have a number of models, some simple some complex, how do we determine which to use?

We cannot optimize k (or more generally model complexity) directly on the training set, as the more complex model (the model with most variance) will always perform better.

We can split the dataset in three parts

- Train (50% of data)- Fit the models
- Validation (25% of data)- Estimate prediction error and select the best model
- Test (25% of data)- Assess generalization error of the best model

Parameters that are not inferred on the training set, such as k in kNN, are usually known as hyperparameters.

Estimating in-sample prediction error

What do we do if do not have enough data for a three-way split?

Estimating in-sample prediction error

What do we do if do not have enough data for a three-way split?

We can try to estimate the **in-sample prediction error** (e.g. by in some sense adjusting for how overly optimistic our prediction is on the training set, by considering how complex the model is):

- Akaike information criterion
- Bayesian information criterion
- Minimum description length
- Vapnik-Chervonenkis dimension

The most commonly used practical solution in ML

- Cross-validation

K-fold cross-validation

- Split the dataset evenly in K parts
- Fit and select model on $K-1$ parts of the data
- Assess the model on the remaining part
- Estimate the prediction error as the average error in each assessed fold

Table: 5-fold cross-validation

	Part 1	Part 2	Part 3	Part 4	Part 5
Fold 1:	Train	Train	Train	Validation	Test
Fold 2:	Train	Train	Validation	Test	Train
Fold 3:	Train	Validation	Test	Train	Train
Fold 4:	Validation	Test	Train	Train	Train
Fold 5:	Test	Train	Train	Train	Validation

K-fold cross-validation

Note: $K = N$ is also called leave-one-out cross-validation.

- (+) Unbiased estimator of the true prediction error
- (-) Can have high variance because of overlap in training sets (Increases with K)
- (-) Can have high computational burden

Parametric and non-parametric methods

A ML method can be described as parametric or non-parametric, but what do these term mean?

- Parametric - a model that summarizes data with parameters of fixed size.
- Non-parametric - a model that does not ...

Parametric and non-parametric methods

A ML method can be described as parametric or non-parametric, but what do these term mean?

- Parametric - a model that summarizes data with parameters of fixed size. They are typically
  - simple and interpretable
  - fast learners and require less data to learn from
  - constrained by their underlying functional form
- Non-parametric - a model that does not ...

Parametric and non-parametric methods

A ML method can be described as parametric or non-parametric, but what do these term mean?

- Parametric - a model that summarizes data with parameters of fixed size. They are typically
  - simple and interpretable
  - fast learners and require less data to learn from
  - constrained by their underlying functional form
- Non-parametric - a model that does not ...These are often
  - flexible and uninterpretable
  - slow learners and require more data to learn from
  - unconstrained and use no assumptions of an underlying functional form

Parametric and non-parametric methods

A random list of methods (parametric or non-parametric?)

- linear and logistic regression
- linear discriminant analysis
- k-Nearest Neighbours
- decision trees
- support vector machines

Parametric and non-parametric methods

A random list of methods (parametric or non-parametric?)

- linear and logistic regression (parametric)
- linear discriminant analysis (parametric)
- k-Nearest Neighbours (non-parametric)
- decision trees (non-parametric)
- support vector machines (non-parametric)

Parametric and non-parametric methods

A random list of methods (parametric or non-parametric?)

- linear and logistic regression (parametric)
- linear discriminant analysis (parametric)
- k-Nearest Neighbours (non-parametric)
- decision trees (non-parametric)
- support vector machines (non-parametric)

Some thoughts: it is not always clear what makes a method parametric or non-parametric, and perhaps this is okay.

Example: small, fixed size neural networks are considered parametric, whereas, if the neural network architecture is adapting based on the training set then I think it would fall into the non-parametric category.

The curse of dimensionality

Machine learning models often have to learn how to make decisions in high dimensional spaces.

Example: kNN classifier operating in a high dimensional space of many input features. [1]

- Lets assume this space has D-dimensions and we have n points uniformly distributed in the range of $[0, 1]$ along each axes to build our classifier on.
- We want a certain number of points in our neighbourhood to base our prediction of the class label on, so we choose $k = 0.01n$.

A hyper-cube around a sample point x would need to be $0.01^{1/D}$ large on average to contain these points.

- In one dimension the size of the cube along each dimension would thus be 0.01
- In two, 0.1, in three 0.22, in four 0.32 and in ten 0.63

[1] Example from Introduction to Statistical Learning, Hastie et al. 2009

The curse of dimensionality

Machine learning models often have to learn how to make decisions in high dimensional spaces.

Example: kNN classifier operating in a high dimensional space of many input features. [1]

- Lets assume this space has D-dimensions and we have n points uniformly distributed in the range of $[0, 1]$ along each axes to build our classifier on.
- We want a certain number of points in our neighbourhood to base our prediction of the class label on, so we choose $k = 0.01n$.

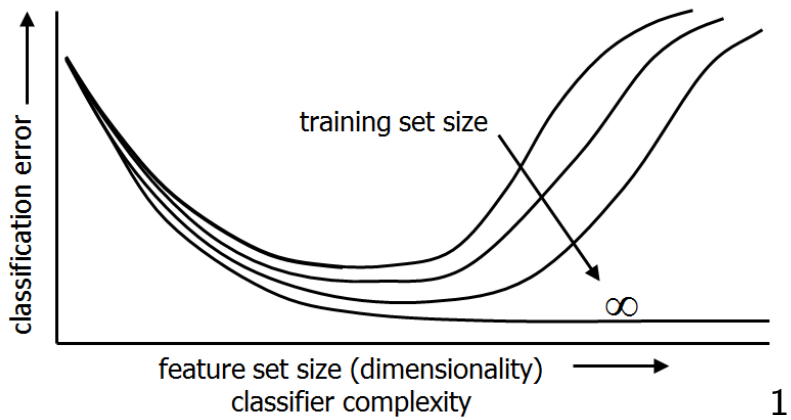
As we increase the dimensions the method stops being local, that is, decisions of the classifier, will be based on points that are further and further away. To keep distance fixed, we will need to grow our sample size expontially with D.

[1] Example from Introduction to Statistical Learning, Hastie et al. 2009

Curse of dimensionality

This is known as **the curse of dimensionality** and comes up in many different areas of machine learning and other fields.

Typically, features help in the prediction task as well, so one often observes a peak with an optimal tradeoff between disadvantages of extra dimensions and advantages the features might bring given the amount of data.



[1]

[1] Figure from `http://37steps.com/2279/the-peaking-paradox/`

No free lunch theorem

> ***The no free lunch theorem*** *for machine learning states that, averaged over all possible data-generating disitributions, every classification algorithm has the same error rate when classifying previously unobserved points.*

Deep Learning, Goodfellow et al. (2016), theorem due to Wolpert, 1996

So why does machine learning work? (I guess some would argue it does not...)

No free lunch theorem

> ***The no free lunch theorem*** *for machine learning states that, averaged over all possible data-generating disitributions, every classification algorithm has the same error rate when classifying previously unobserved points.*

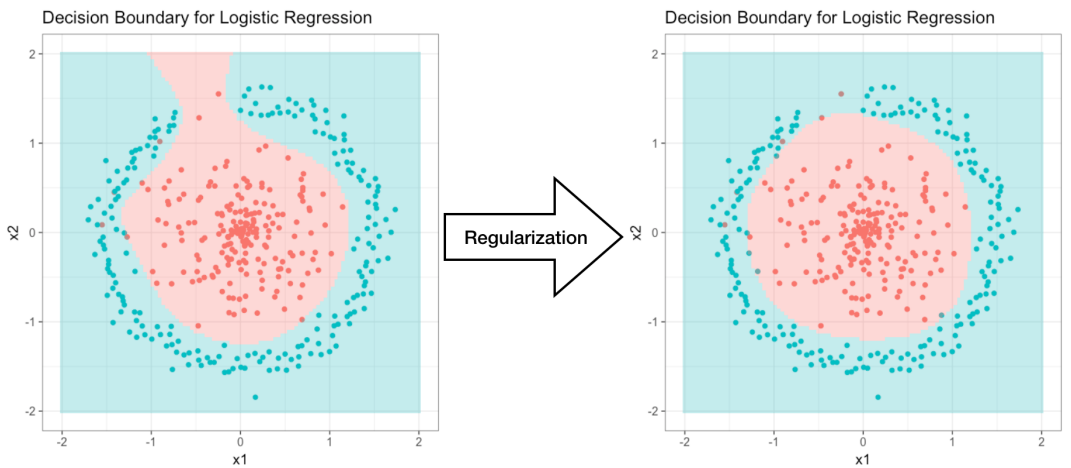
Deep Learning, Goodfellow et al. (2016), theorem due to Wolpert, 1996

- We do not apply our methods to all data-generating distributions.
- We make assumptions about the data we apply them to and these assumptions may hold enough so that our ML algorithm also works.
- We may for instance assume that the unobserved points are drawn from the same distribution as the points in the training set.

Regularization

"**Regularization** is any modifications we make to a learning algorithm that is intended to reduce its generalization error but not its training error"

Deep Learning, Goodfellow et al. (2016)



[1]

[1] Illustration from
https://towardsdatascience.com/understanding-regularization-in-machine-learning-5a0369ac73b9

Regularization

"**Regularization** is any modifications we make to a learning algorithm that is intended to reduce its generalization error but not its training error"

Deep Learning, Goodfellow et al. (2016)

Unintended effects may also be included. That is,

- we can have **explicit regularization**, by for instance choosing to add terms to our loss function that will penalize solutions with many non-zero parameters.
- However, we can also have **implicit regularization**, in which our choice of model and training method will tend to find solutions that are more regularized compared to other models and training methods.

Ideally, regularization should reduce the variance of the model with less substantial increases in bias.

Thanks, questions?