# Machine learning

## Lecture 1

> Have an understanding of the goal of machine learning (ML) models.  
> Have a good understanding of basic mathematical concepts used in ML and be able to apply them in the design and implementation of ML methods.

1. Linear algebra  
A tensor is an array of number that may have:  
- a zero dimension and be a scalar,  
- one dimension and be a vector,  
- two dimensions and be a matrix,  
- more dimension…

Transpose: mirror image with regard to the main diagonal.  
(**A**T)I,j = **A**j,I (**AB**)T = **B**T**A**T

Matrix inversion:   
**A**-1**A** = **I**n

Invertibility: a matrix cannot be inverted if   
- the number of rows and columns is not the same, or  
- some rows and columns are “redundant” (linearly dependent)

Norms: functions that measure how “large” a vector is.

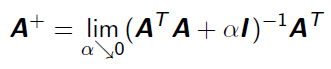
Special vectors and matrices:  
- unit vector ||**x**||n = 1  
- Symmetric matrix **A**  = **A**T  
- Orthogonal matrix **AA**T = **I** = **A**T**A** and **A**T = **A**-1

Eigendecomposition:   
- Eigenvector and eigenvalue **Ax** = λ**x**  
- Eigendecomposition of a matrix **A** = **V**diag(λ)**A**-1

- Every real symmetric matrix has a real orthogonal eigendecomposition **A = Q∧Q**T  
where **Q** is an orthogonal matrix composed of eigenvectors of **A** and **∧** is a diagonal matrix.   
- The eigenvalue ∧ii is associated with the eigenvector in column i of **Q**, denoted as **Q**:,i.

- The eigendecomposition of a real symmetric matrix is used in optimization of quadratic expressions of the form f(**x**) = **x**T**Ax** under the constraint ||**x**||2 = 1.   
- The maximal (minimal) value of f within the constraint region is equal to the maximal (minimal) eigenvalue.

Singular value decomposition:  
- More general than eigenvalue decomposition, matrix need not be square **A = UDV**T  
- **U** and **V** are square matrices and are both orthogonal, **D** is diagonal.  
- The diagonal elements of D are **singular values** of **A;** the column of **U** and **V** are **left-singular** and **right-singular vectors** of **A**, respectively.

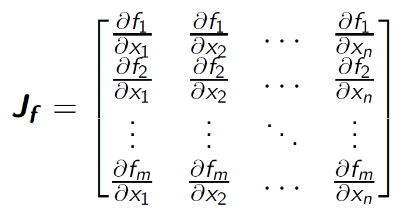
Moore-Penrose pseudoinverse:  
- Matrix inversion not defined on matrices that are not square.   


**x = A**­+**y**

If the equation has   
- Exactly one solution: same as inverse  
- No solution: gives solution with smallest error ||**Ax – y**||2  
- Many solutions: gives solution with smallest norm of **x**.

Computing pseudoinverse  
- Efficient implementations are based on the formula by the singular decomposition **A+** = **VD**+**U**T- **U, D, V,** are from the singular value decomposition of **A**.  
- The pseudoinverse **D**+ of **D** is obtained by taking the reciprocal non-zero elements and after that taking the transpose of the resulting matrix.

Trace:   
Tr(**ABC**) = Tr(**BCA)** = Tr(**CAB**)

2. Gradient-based optimization  
Gradient 🡪 size of gradient of **A** is the same as the size of A.

The Jacobian **Jf** is a generalization of the gradient for vector valued functions. For the special case of a scalar-valued function, the Jacobian is the transpose of the gradient.

Optimization:  
minimizing or maximizing some function f(x). **x**\* = argmin f(**x**)

Gradient-based optimization:  
The derivative tells us how to change x in order to make a small improvement of f(x).

Some notations:  
- Input variable x (scalar) or **x** (vector)  
- Quantitative (numerical) outputs are denoted with y.   
- Qualitative outputs are denoted with g (from group) and take values from a set *G*.

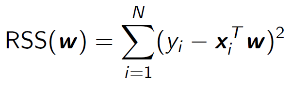
The learning task:  
- Given a value of the input vector **x** make a good prediction of the output y, denoted as ŷ.  
- Both y and ŷ should take values from the same numerical set.   
- Similarly, g and ĝ should both take values from the same set G.   
- Training data: (**X, y**) and/ or (**X**, **g**).   
- Construct a prediction rule based on the training data.

3. Linear model and Nearest-neighbours model  
Linear model fit:  
- Strong assumptions about the structure of the decision boundary.

K-nearest neigbours:  
- Weak assumptions about the structure of the decision boundary.

Linear model fit by least squares   
- Input vector **x**T = (x1, x­2, …, xp­)  
- Output y predicted using the model   
- (0 ≤ I ≤ p) are the parameters of the linear model  
- In vector form

Hyper(space) terminology:  
- Points **x**, form a **hyperplane** in the (p + 1)-dimensional input-output hyperspace.  
- If **x** is extended with constant 1 then the hyperplane includes the origin and it forms a **subspace**.  
- If 1 is not included then the hyperplane is an **affine** set and it cuts the y-axis at the point (**0,** ), where the vector **0** has all xi coordinates equal to 0.   
- Assume that 1 is included in **x** and in  **-** The function f(x) = **w**T**x** defined on the p-dimensional (input) space is a **linear** function (omit the ^ over the w’s since they are now free variables).  
- The gradient is a vector pointing along the direction of maximal change.

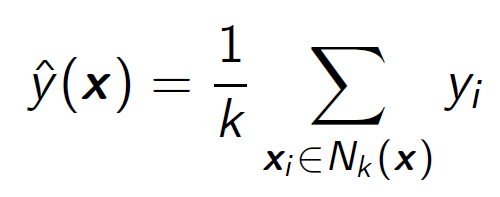
**Least squares** method  
Find coefficients which minimize the error estimated with the **residual sum of squares,** assuming N input –output pairs.

- RSS(**w**) is a quadratic function.   
- A minimum always exists, though not necessarily a unique one.

- If **X**T**X** is non-singular there exists a unique solution given by

**Question:** Why not simply  **y – Xw** = 0 🡪 **y** = **Xw** 🡪 = **X**-1**y**?  
🡪 Because the inverse is not always defined. The matrix is not always square!.

- For each input **x**i there corresponds the fitted output. This is called “making a prediction” for **x**i.   
- The entire fitted surface (hyperplane) is fully characterized by the parameter vector .

**Nearest-neighbours** model  
  
- For the 1-NN method the error on the **training data**  is always 0.   
- The **effective** number of parameters of k-NN is N/k which is in general bigger than *p* (number of weights). N is the size of the training set.

For instance, assume non-overlapping neighbourhoods.   
- There will be N/k neighbourhoods.  
- To each neighbourhood there correspond one parameter (the mean of the elements of the neighbourhood).

4. Probability theory  
A mathematical framework for dealing with uncertainty.

- A **random variable** is a variable that can take values randomly.   
- There are two types of random variables: **discrete** and **continuous**.   
- Consequently: **probability mass functions** vs **probability density functions.**

Probability mass function  
- The domain of a probability mass function P is the set of all possible states of the random variable x.

Probability density function  
- The domain of the probability density function *p* must be the set of all possible states of x.