POLITECNICO

MILANO 1863

Numerical Analysis for Machine Learning

Lorenzo Bozzoni

December 4, 2023

Contents

1 Basic concepts of linear algebra

The following are the main concepts of linear algebra we are going to face during the starting phase of the course:

1. Linear systems of equations: $A\underline{x} = \underline{b}$

2. Eigenvalues and eigenvectors: $A\underline{x} = \lambda\underline{x}$

3. Singular value decomposition (SVD): $A\underline{v} = \sigma u$

4. Minimization problem

5. Factorization: $PA = LU$

2 Matrix-vector multiplication

$$\underline{c} = \underbrace{\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}}_{A_1} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\underline{x}} = \begin{bmatrix} 1x_1 + 2x_2 \\ 3x_1 + 4x_2 \\ 5x_1 + 6x_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} x_1 + \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} x_2}_{\text{linear combination}}$$

We say that the vector $\underline{c}$ belongs to the **column space** of A_1, i.e. $\underline{c} \in \mathcal{C}(A_1)$.

$$\overbrace{\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}}^{A_2} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$
$$\underline{a_1}\ \underline{a_2}\ \underline{a_3}$$

In this case we can easily notice that $\underline{a_3} = \underline{a_1} + \underline{a_2}$, which means that one column can be expressed as a linear combination of the other two (this means that the matrix A_2 is singular). Because of this, we can say that $\mathcal{C}(A_2) = \mathcal{C}(A_1)$, i.e. the column space of A_2 is the same as the column space of A_1.

Those columns spaces are a plane passing through the origin and spanned by the two vectors $\underline{a_1}$ and $\underline{a_2}$ (they define the slope of that plane).

Let's now consider these matrix:

$$\underbrace{\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 10 \end{bmatrix}}_{A_3} \qquad\qquad \underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}}_{A_4}$$

This left-hand matrix column space is $\mathcal{C}(A_3) = \mathbb{R}^3$, i.e. the entire real space of three dimensions. This is because the three vector columns of A_3 are linearly independet so they span the entire space and not just a plane. While the column space of A_4 is instead: $\mathcal{C}(A_3) = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$ i.e. just a line since the three columns are linearly dependent and so they lie on the same line (they are parallel) just with different magnitude.

Another measure regarding matrices is the **dimension** or **rank**:

- $rank(A_1) = 2$

- $rank(A_2) = 2$

- $rank(A_3) = 3$

- $rank(A_4) = 1$

The rank is the number of linearly independent columns (or rows) of a matrix.

Let's consider again the matrix A_3:

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} x_1 + \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} x_2 + \begin{bmatrix} 7 \\ 8 \\ 10 \end{bmatrix} x_3}_{\in \mathcal{C}(A_3)} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

so $\underline{b}$ must be in $\mathcal{C}(A_3)$ in order to have the system solvable. If $\underline{b} \notin \mathcal{C}(A_3)$, then the system is not solvable. In this particular case we have that the columns of A_3 are linearly independent, so the system is solvable for any $\underline{b}$ because

$\mathcal{C}(A_3) = \mathbb{R}^3$ and so $\underline{b}$ is for sure inside that space.

Given A, find $\mathcal{C}(A)$. How can we solve this problem? Considering $\underline{a_1}, \dots, \underline{a_n}$ as A columns, we can use the following iterative algorithm:

- put $\underline{a_1}$ in $\mathcal{C}(A)$

- if $\underline{a_2} = \alpha\underline{a_1} \to \underline{a_2} \notin \mathcal{C}(A)$, otherwise put $\underline{a_2}$ in $\mathcal{C}(A)$

Until you reach the last column.

2.1 Row-reduced echelon form

Given the matrix A, defined as follow:

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

we can obtain the **row-reduced echelon form** of A by applying the following operations:

$$A = CR = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \end{bmatrix}$$

where C is the matrix containing the columns of A that are linearly independent (i.e. $\mathcal{C}(A)$) and R is the matrix of the coefficients of the linear combination of the columns of A that gives the columns of C.

Let's now consider the following matrix:

$$A_1 = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \qquad\qquad A_1^\mathsf{T} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

What we can say about A_1^T column space? Is there any relationship with the column space of A_1?
In order to compute its column space, we can start noticing that: $\underline{a_3} = 2\underline{a_2} - \underline{a_1}$. So, in general, we can say that:

$$dim(\mathcal{C}(A)) = dim(\mathcal{C}(A^\mathsf{T})) = r \leq n \qquad \text{where } n \text{ is the number of columns of } A$$

3 Matrix-matrix multiplication

$$C = AB = \begin{bmatrix} | & | & | \\ \underline{a_1} & \dots & \underline{a_n} \\ | & | & | \end{bmatrix} \begin{bmatrix} - & \underline{b_1} & - \\ - & \vdots & - \\ - & \underline{b_n} & - \end{bmatrix} = \underset{\underset{\text{col}}{\downarrow}}{\underline{a_1}}\,\overset{\overset{\text{row}}{\downarrow}}{\underline{b_1}} + \dots + \underline{a_n}\underline{b_n}$$

All the products that are summed at the end of the equation are matrices of rank 1.
Example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 2 & 3 \end{bmatrix} = \underbrace{\begin{bmatrix} 2 & 1 \\ 6 & 3 \end{bmatrix}}_{\text{rank } = 1} + \underbrace{\begin{bmatrix} 4 & 6 \\ 8 & 12 \end{bmatrix}}_{\text{rank } = 1} = \begin{bmatrix} 6 & 7 \\ 14 & 15 \end{bmatrix}$$

4 Factorizations

1. $A = LU$ or $PA = LU$

2. $A = QR$ where Q is orthogonal and R is upper triangular This is an improved version of the Row-reduced echelon form because that worked only for square matrices, while this works for any matrix.

3. Eigenvalues and eigenvectors decomposition: when $S = S^\mathsf{T}$ (symmetric matrix) we can factorize it as $S = Q\Lambda Q^\mathsf{T}$ where Λ is a diagonal matrix and Q is an orthogonal matrix (they are all squared matrices)

4. Generalization of the above: $A = X\Lambda X^{-1}$ where X is a non-orthogonal matrix

5. $A = U\Sigma V^\mathsf{T}$ where U and V are orthogonal matrices and Σ is a pseudo-diagonal matrix

A matrix is said to be pseudo-diagonal if it has the following form:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_n \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad m \text{ rows} \times n \text{ columns}$$

So it has diagonal elements for the first n rows then it has all zeros.

4.1 Orthogonal matrices

A matrix Q is orthogonal if $Q^\mathsf{T} Q = I$ (i.e. $Q^\mathsf{T} = Q^{-1}$). This means that the columns of Q are orthonormal, i.e. they are orthogonal and have unit norm.

The determinant of a orthogonal matrix is ± 1.
Properties:

- $||Q\underline{x}|| = ||\underline{x}||$

- $||Q\underline{x}||^2 = (Q\underline{x})^\mathsf{T} Q\underline{x} = \underline{x}^\mathsf{T} \underbrace{Q^\mathsf{T} Q}_{I} \underline{x} = ||\underline{x}||^2$
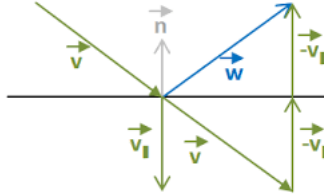
The first property is particularly easy to interpret since it means that when we multiply an orthogonal matrix to a vector, the norm of the vector doesn't change. As a proof of this, we can consider the following examples:

4.1.1 Rotation

A classical rotation matrix is:

$$\begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$$

4.1.2 Reflection

The horizontal line in the figure represent a plane π while $\underline{n}$ is its normal vector of length 1. Given $\underline{v}$ to obtain $\underline{w}$ we can use the following formula:

$$\underline{w} = \underline{v} - 2(\underline{v}^\mathsf{T}\underline{n})\underline{n} = \underbrace{(I - 2\underline{n}\underline{n}^\mathsf{T})}_{\text{reflection matrix}} \underline{v}$$

Moreover, the reflection matrix R is not only orthogonal, but also the inverse of itself, i.e. $R^{-1} = R^\mathsf{T}$. This makes sense because if we apply the reflection matrix twice, we obtain the original vector $\underline{v}$, i.e. the reflection of the reflection is the starting vector.

If we didn't have the 2 in the formula, we would obtain the projection of $\underline{v}$ on the plane π which is called orthogonal projection and the matrix R would be singular.

Let's now dive a bit into the third point of the factorization list. We said that when $S = S^\mathsf{T}$ (symmetric matrix) we can factorize it as $S = Q\Lambda Q^\mathsf{T}$ where Λ is a diagonal matrix and Q is an orthogonal matrix.

$$S = S^\mathsf{T} = \underbrace{(Q\Lambda))}_{\tilde{Q}} Q^\mathsf{T} = \tilde{Q}Q^\mathsf{T}$$

$$\tilde{Q} = \underline{q_1}\underline{\lambda_1} + \cdots + \underline{q_n}\underline{\lambda_n}$$

Where the q vectors are columns and λ vectors are rows. So we can reformulate:

$$S = (\underline{q_1}\underline{\lambda_1} + \cdots + \underline{q_n}\underline{\lambda_n})Q^\intercal = \underline{q_1}\lambda_1\underline{q_1}^\intercal + \cdots + \underline{q_n}\lambda_n\underline{q_n}^\intercal$$

This is called **spectral decomposition** of matrix S and $q_1, \ldots, q_n$ are the eigenvectors of S while $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of S.

$$S\underline{q_1} = \lambda_1\underline{q_1} = (\underline{q_1}\lambda_1\underline{q_1}^\intercal + \cdots + \underline{q_n}\lambda_n\underline{q_n}^\intercal)\underline{q_1} = \underline{\lambda_1}\underline{q_1}(\underline{q_1^\intercal}\underline{q_1})$$

All the other products are null since the vector $\underline{q_1}$ is orthogonal to all the other vectors $\underline{q_i}$ for $i \neq 1$ (recall that they are eigenvectors).

5 Null spaces

Let's consider the starting problem for a linear system of equations:

$$A\underline{x} = \underline{b} \qquad \text{with} \qquad A \in \mathbb{R}^{m \times n}, \text{rank}(A) = r$$

We are going to introduce 2 more spaces other than the column ones. To do so we consider:

$$A\underline{x} = \underline{0} \qquad \rightarrow \qquad N(A) \equiv \ker(A) = \{\underline{x} \in \mathbb{R}^n : A\underline{x} = \underline{0}\}$$

$$A^\intercal\underline{x} = \underline{0} \qquad \rightarrow \qquad N(A^\intercal) \equiv \ker(A^\intercal) = \{\underline{x} \in \mathbb{R}^n : A^\intercal\underline{x} = \underline{0}\}$$

So now, adding the so called **null spaces** we have that:

1. $\mathcal{C}(A) \subset \mathbb{R}^m$ and $dim(\mathcal{C}(A)) = r$

2. $\mathcal{C}(A^\intercal) \subset \mathbb{R}^n$ and $dim(\mathcal{C}(A^\intercal)) = r$

3. $N(A) \subset \mathbb{R}^n$ and $dim(N(A)) =?$

4. $N(A^\intercal) \subset \mathbb{R}^m$ and $dim(N(A^\intercal)) =?$

We still do not know the dimensions of those spaces.

Example

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \implies \begin{cases} x_1 + 4x_2 + 7x_3 = 0 \\ 2x_1 + 5x_2 + 8x_3 = 0 \\ 3x_1 + 6x_2 + 9x_3 = 0 \end{cases}$$

We compute the first equation

$$x_1 = -4x_2 - 7x_3 \implies \begin{cases} -3x_2 - 6x_3 = 0 \\ -6x_2 - 12x_3 = 0 \end{cases}$$

What is important to notice is that A has rank= 2 so we have $3 - 2 = 1$ **degrees of freedom**, i.e. we can choose one variable and the other two are automatically defined. This is visible in the last two equations of the system for example. In general, the degrees of freedom are given by $n - r$ where n is the number of columns of A and r is the rank of A.

If we had 10 instead of 9 in A we would have had $3 - 3 = 0$ degrees of freedom. This would translate in having the matrix A full rank and $N(A) = \{\underline{0}\}$ so the only solution would be the null vector.

6 Null space cardinality

In the first lecture, we defined 4 spaces: $N(A), N(A^\intercal), \mathcal{C}(A), \mathcal{C}(A^\intercal)$. For the last two we defined also their cardinality whilst for the first ones we weren't able to tell yet. In this lecture we are going to find those values and prove them. In order to do so, we start from few useful properties:

1. $\underline{x} = \underline{0} \in N(A)$ for any matrix A

2. if $\underline{x}, \underline{y} \in N(A) \implies A(\underline{x} + \underline{y}) = \underline{0}$

3. if $\underline{x} \in N(A) \implies \alpha \underline{x}$ with $\alpha \in \mathbb{R} \implies A(\alpha \underline{x}) = \underline{0}$

Consider, once again, the matrix $A \in \mathbb{R}^{m \times n}$, rank$(A) = r \le n$. We have seen the decomposition $A = CR$, where C contains the linearly independent columns of A and R contains the coefficients that allow to recover the columns of A starting from its independent columns. So, the matrix A can be rewritten as:

$$A = \begin{bmatrix} A_1 & A_2 \end{bmatrix} \qquad A_1 \in \mathbb{R}^{m \times r} \qquad A_2 \in \mathbb{R}^{m \times (n-r)}$$

Where A_1 contains the independent columns of A and A_2 the dependent ones. Example:

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}}_{A_1} \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \end{bmatrix}}_{B}$$

Since we have the last column of A, that is linearly dependent so it belongs to A_2, we can reformulate it in this way:

$$A = \begin{bmatrix} A_1 & A_2 \end{bmatrix} = \begin{bmatrix} A_1 & A_1 B \end{bmatrix}$$

We build a new matrix K defined as follows:

$$K = \begin{bmatrix} -B \\ I_{n-r} \end{bmatrix} \qquad K \in \mathbb{R}^{n \times (n-r)} \qquad B \in \mathbb{R}^{r \times (n-r)}$$

$$AK = \begin{bmatrix} A_1 & A_1 B \end{bmatrix} \begin{bmatrix} -B \\ I_{n-r} \end{bmatrix} = A_1(-B) + A_1 B = 0$$

Where the last 0 is actually a matrix of zeros of dimension $m \times (n-r)$ because A has size $m \times n$ and K has size $n \times (n-r)$. We have that:

$$AK = 0 \implies A\underline{k_i} = 0 \qquad \forall i \in \{1, \ldots, n-r\}$$

Where k_i is the i-th column of K. This means that: $\underline{k_i} \in N(A) \ \forall i$.

Now, we want to demonstrate that: $K\underline{u} = 0 \implies \underline{u} = \underline{0}$. To do so, we start from expanding K from its definition:

$$K = \begin{bmatrix} -B \\ I \end{bmatrix} \underline{u} = 0 \implies \begin{bmatrix} -B\underline{u} \\ \underline{u} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Where the two zero vectors have dimension r and $n-r$ respectively! Considering the second row of the matrix we get: $\underline{u} = \underline{0}$ so all columns of K are linearly independent.

If we consider the problem $(\star)$ $A\underline{x} = \underline{0}$, we want to prove that each $\underline{x}$ that satisfy $(\star)$ must be a linear combination of the columns of K.

$$A_1 \underline{x} = \underline{0} \in \mathbb{R}^m \implies \underline{x} = \underline{0} \in \mathbb{R}^r$$

Because A_1 has linearly independent columns, i.e. has full rank.

$$A\underline{u} = \underline{0} \in \mathbb{R}^m \implies \begin{bmatrix} A_1 & A_1 B \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} A_1 \underline{u_1} + A_1 B \underline{u_2} \end{bmatrix} = A_1 \begin{bmatrix} \underline{u_1} + B\underline{u_2} \end{bmatrix} = \underline{0}$$

We can notice that the last formulation obtained in the equation has the same form as the one from where we started the prove, so we can say that:

$$\underline{u_1} + B\underline{u_2} = \underline{0} \implies \underline{u_1} = -B\underline{u_2}$$

$$\underline{u} = \begin{bmatrix} -B\underline{u_2} \\ \underline{u_2} \end{bmatrix} = \underbrace{\begin{bmatrix} -B \\ I \end{bmatrix}}_{K} \underline{u_2} = K\underline{u_2} \implies dim(N(A)) = n - r$$

The big picture

- **C(A^T)** dim r — row space, all vectors $A^T y$
- **C(A)** dim r — column space, all vectors Ax
- **N(A)** dimension $n - r$ — nullspace $Ax = 0$
- **N(A^T)** dimension $m - r$ — left nullspace $A^T y = 0$
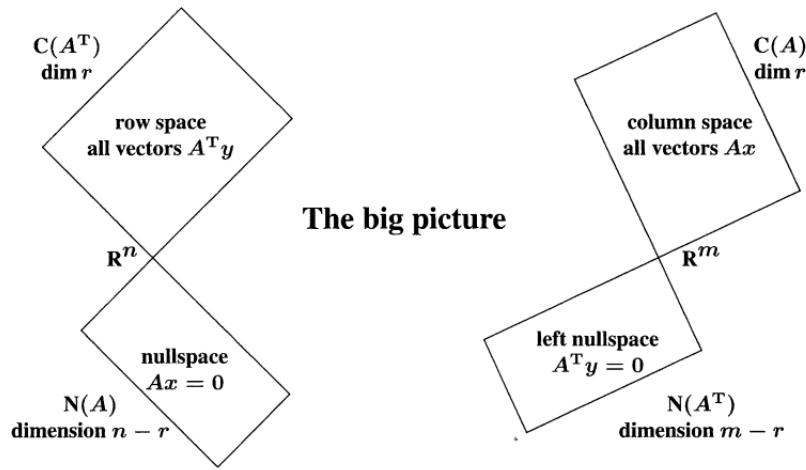
7 Eigenvalues and eigenvectors

Start considering a generic square matrix $n \times n$. We are going later to discuss even the simmetry and positive definite properties. Here below are the vectorial and the matrix form of the eigenvalue problem:

$$A\underline{x_i} = \lambda_i \underline{x_i} \quad i = 1, \dots, n \qquad X^{-1}AX = \Lambda$$

Where in the right-hand side there is a diagonal matrix Λ with the eigenvalues of A on the diagonal while the matrix X with the eigenvectors of A as columns.

7.1 Eigenvectors of matrix power

What can we say about the eigenvectors and eigenvalues of A^2?

$$A^2 \underline{x_i} = A(A\underline{x_i}) = A(\lambda_i \underline{x_i}) = \lambda_i (A\underline{x_i}) = \lambda_i^2 \underline{x_i}$$

So the eigenvalues of A^2 are the eigenvalues of A squared. This is valid for any power of A since this method can be applied recursively and it is very useful when there are problems in which a matrix is iteratively multiplied many times.

Important:
Given $A \in \mathbb{R}^{n \times n}$ full rank, then any vector $\underline{v} \in \mathbb{R}^n$ can be written as a linear combination of the eigenvectors (x_i) of A.

7.2 Power method

In mathematics, power iteration (also known as the power method) is an eigenvalue algorithm: given a diagonalizable matrix A, the algorithm will produce a number λ, which is the greatest (in absolute value) eigenvalue of A, and a nonzero vector v, which is a corresponding eigenvector of λ, that is, $Av = \lambda v$. The algorithm is also known as the Von Mises iteration.

There is also an **inverse PM** which is applied to A^{-1} to find the minimum eigenvalue or also **PM with a shift** applied to $(A - \alpha I)^{-1}, \alpha \in \mathbb{R}$ to find the closest eigenvalue to α.

Can even be used in "deflation method" iteratively:

$$\begin{bmatrix} \lambda_1 & b_1^{\mathsf{T}} \\ 0 & A_1 \end{bmatrix}$$

the original matrix could be reduced in that form and at every iteration the procedure is applied to the A_1 matrix. This works only if we have different eigenvectors (or values) [check].

7.3 Similar matrices

Given two matrices $A, B \in \mathbb{R}^n$, they are said to be similar if $B = M^{-1}AM$, with M invertible.

$$\underbrace{M^{-1}AM}_{B} \underline{y} = \lambda \underline{y} \implies A \underbrace{M\underline{y}}_{w} = \lambda \underbrace{M\underline{y}}_{w} = A\underline{w} = \lambda\underline{w}$$

Where $\lambda, \underline{y}$ contain respectively the eigenvalues and the eigenvectors of B. What we get from this equation is that **similar matrices share the same eigenvectors with scaled eigenvalues**.

7.4 QR factorization

Here is introduced in the context of eigenvalues. Let's consider a matrix $A \in \mathbb{R}^{m \times n}$ where $m \geq n$ and $rank(A) = n$ (it has all independent columns). We can factorize A in this way:

$$A = QR \qquad Q \in \mathbb{R}^{m \times n} \qquad R \in \mathbb{R}^{n \times n}$$

Where is Q is an orthogonal matrix and R is an upper triangular matrix.Since we are dealing with eigenvalues and eigenvectors, we are now going to consider the matrix A squared with the dimension $n \times n$.

7.4.1 QR iteration

$$A = A^{(0)} = Q^{(0)} R^{(0)}$$

$$A^{(1)} = Q^{(0)\mathsf{T}} A^{(0)} Q^{(0)} = Q^{(1)} R^{(1)}$$

So, iterating this procedure we get:

$$A^{(2)}, \dots, A^{(S)} = \text{is upper triangular}$$

After S iterations you obtain an upper triangular matrix. The matrices $A, A^{(0)}, A^{(1)}, \dots, A^{(S)}$ are similar, so they share the same eigenvalues.

But, how can i compute Q?

With the **Gram-Schmidt** procedure. It works also for non-square matrices.

Let's start from a generic matrix A:

$$A = \begin{bmatrix} | & | & | \\ \underline{a_1} & \dots & \underline{a_n} \\ | & | & | \end{bmatrix}$$

The algorithm is iterative and it is applied to the columns of A in such way:

$$\underline{q_1} = \frac{\underline{a_1}}{||\underline{a_1}||}$$

The vector $\underline{q_1}$ is obtained by normalizing the first column of A, in such manner the new obtained vector will have norm 1.

$$\underline{q_2} = \underline{a_2} - \underline{q_1}(\underline{q_1}^{\mathsf{T}} \underline{a_2}) \implies \underline{q_2} = \frac{\underline{q_2}}{||\underline{q_2}||}$$

The second vector is obtained by subtracting from the second column of A the projection of $\underline{a_2}$ on $\underline{q_1}$, in such manner the new vector will be orthogonal to $\underline{q_1}$ and will have norm 1.

$$\underline{q_3} = \underline{a_3} - \underline{q_1}(\underline{q_1}^{\mathsf{T}} \underline{a_3}) - \underline{q_2}(\underline{q_2}^{\mathsf{T}} \underline{a_3}) \implies \underline{q_3} = \frac{\underline{q_3}}{||\underline{q_3}||}$$

And so on... Recall that the orthogonality is needed since we want to obtain an orthogonal matrix Q useful for the factorization. With Gram-Schmidt the resulting matrix not only will be orthogonal but also orthonormal, this means that its columns will have norm unitary.

Let's now continue with the factorization journey. We have said in the introduction of types of factorizations that, given $A \in \mathbb{R}^{n \times n}$, we have:
$$A = X \Lambda X^{-1}$$

Where X has as columns the eigenvectors of A, while Λ is a diagonal matrix with the eigenvalues of A on the diagonal. Now, let's consider the case where the matrix S is symmetric.
$$S \in \mathbb{R}^{n \times n} \qquad S = S^{\mathsf{T}}$$

We can factorize S as follows:
$$S = Q \Lambda Q^{\mathsf{T}}$$

Where Q is orthogonal (this is true only because S is symmtric) and Λ is diagonal. We can prove that Q is orthogonal by:

1. Consider the two vectors $\underline{x}, \underline{y}$ such as: $S\underline{x} = \lambda \underline{x}$ and $S\underline{y} = 0\underline{y}$. So, we are saying that both vectors are eigenvectors.
$$\left. \begin{array}{l} \underline{y} \in N(S) \\ \underline{x} \in \mathcal{C}(S) = \mathcal{C}(S^{\mathsf{T}}) \end{array} \right\} \implies \underline{x} \perp \underline{y}$$

   This is confirmed also by the scheme done during lecture with the 4 blocks. Notice that we have not specified or made any assumption on the value of λ.

2. Similar to point 1, we consider the two vectors $\underline{x}, y$ such as: $S\underline{x} = \lambda \underline{x}$ and $S\underline{y} = \alpha \underline{y}$. Now, consider the matrix $(S - \alpha I)$, we can write:
$$(S - \alpha I)\underline{y} = 0\underline{y} \implies \underline{y} \in N(S - \alpha I)$$
$$(S - \alpha I)\underline{x} = (\lambda - \alpha)\underline{x} \implies \underline{x} \in \mathcal{C}(S - \alpha I) = \mathcal{C}((S - \alpha I)^{\mathsf{T}})$$

   So, again we obtain: $\underline{x} \perp \underline{y}$.

There is another property: $\lambda_i \in \mathbb{R}$, so the eigenvalues on the diagonal of Λ are real. Proof:
$$S\underline{x} = \lambda \underline{x} \implies \overline{\underline{x}}^{\mathsf{T}} S\underline{x} = \lambda \overline{\underline{x}}^{\mathsf{T}} \underline{x}$$

The $\overline{\underline{x}}$ represent the conjugate of the vector $\underline{x}$. If that vector has complex components, those elements are conjugated, otherwise, i.e. they are all real, they remain inalterated. In particular, once a complex number is conjugated, the result is a real number, as shown here:
$$(a + ib)(a - ib) = (a^2 + b^2) \in \mathbb{R}$$

From the previos equation, we obtain:
$$\lambda = \frac{\overline{\underline{x}}^{\mathsf{T}} S\underline{x}}{\overline{\underline{x}}^{\mathsf{T}} \underline{x}} \in \mathbb{R}$$

7.5 Positive-definite symmetric matrices (SPD)

Characterizations:

i $\lambda_i > 0 \quad \forall i = 1, \dots, n$

ii $\underline{v}^{\mathsf{T}} S\underline{v} \geq 0 \quad \forall \underline{v} \in \mathbb{R}^n$, with equality if and only if $\underline{v} = 0$

iii Leading determinants are positive.
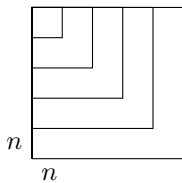
This means that the determinant of the matrix obtained by taking the first k rows and columns of S is positive, $\forall k = 1, \dots, n$.

iv Cholesky decomposition: $S = B^{\mathsf{T}} B$, with B upper triangular

v All pivot elements are positive in the Gaussian elimination process

Let's consider $\lambda > 0$ being a certain eigenvalue.

$$S\underline{x} = \lambda \underline{x}$$

We multiply both sides by $\underline{x}^\intercal$:

$$\underline{x}^\intercal S\underline{x} = \lambda \underline{x}^\intercal \underline{x} = \lambda \|\|\underline{x}\|\|^2 \geq 0$$

Recall that $\underline{x}$ is an eigenvector while the before considered vector $\underline{v}$ is a generic vector. With $\underline{v}$, instead, we have:

$$\underline{v} = (c_1\underline{x}_1 + c_2\underline{x}_2 + \cdots + c_n\underline{x}_n)$$

So we are expressing $\underline{v}$ as a linear combination of the eigenvectors of S.

$$(c_1\underline{x}_1 + c_2\underline{x}_2 + \cdots + c_n\underline{x}_n)^\intercal S(c_1\underline{x}_1 + c_2\underline{x}_2 + \cdots + c_n\underline{x}_n)$$

$$\left. \begin{aligned} c_1^2\underline{x}_1^\intercal S\underline{x}_1 = c_1^2\lambda_1\underline{x}_1^\intercal\underline{x}_1 = c_1^2\lambda_1\|\|\underline{x}_1\|\|^2 \\ c_1c_2\underline{x}_1^\intercal S\underline{x}_2 = c_1c_2\lambda_2\underline{x}_1^\intercal\underline{x}_2 = 0 \end{aligned} \right\} \text{there are two types of components}$$

The first components is given by the eigenvectors with the same direction, while the second no so their scalar product is null (they are orthogonal).

From iv):

$$S = B^\intercal B \implies \underline{v}^\intercal(B^\intercal B)\underline{v} = (\underline{v}^\intercal B^\intercal)(B\underline{v}) = (B\underline{v})^\intercal(B\underline{v}) = \|\|B\underline{v}\|\|^2 \geq 0$$

8 Singular Value Decomposition (SVD)

We are going to use it for:

- Least-squares approximation by introducing the pseudo-inverse of a matrix (Moore-Penrose inverse)

- Low-rank approximation with the Eckart-Young theorem

We start from:

$$A \in \mathbb{R}^{m \times n} \qquad \begin{cases} m = \# \text{ of samples} \\ n = \# \text{ of features} \end{cases}$$

We can write:

$$A = U\Sigma V^\intercal$$

With:

- U with dimensions $m \times m$ and orthogonal

- $V^\intercal$ with dimensions $n \times n$ and orthogonal

- Σ with dimensions $m \times n$ *almost* diagonal

If $m > n$, we can represent the matrices like this:

What is the idea of SVD? Try to change features so variances are maximized and covariances are minimized. We don't want columns to be correlated.

In general: $\text{rank}(A) = r < n$.

$$AV = U\Sigma \impliedby V^\intercal V = I \impliedby V \text{ is orthogonal}$$

The component wise notation is:

$$A\underline{v}_i = \sigma_i\underline{u}_i$$

Given that the rank of A is r:

$$\begin{bmatrix} & & \\ & & \\ & & \\ & & \end{bmatrix} \left[\begin{array}{ccccc|c} \sigma_1 & & & & & \\ & \sigma_2 & & & & \\ & & \ddots & & & \\ & & & \sigma_r & & \\ & & & & 0 & \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right] \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \qquad \begin{cases} \sigma_1, \ldots, \sigma_r > 0 \\ \sigma_{r+1}, \ldots, \sigma_n = 0 \end{cases}$$

Typically: $\sigma_1 > \sigma_2 > \cdots > \sigma_r > \sigma_{r+1} = 0$. We have:

$$\begin{cases} A\underline{v_1} = \sigma_1\underline{u_1} \\ \quad \vdots \\ A\underline{v_r} = \sigma_r\underline{u_r} \end{cases} r \\ \begin{cases} A\underline{v_{r+1}} = \sigma_{r+1}\underline{u_{r+1}} \\ \quad \vdots \\ A\underline{v_n} = \sigma_n\underline{u_n} \end{cases} n - r$$

So the first r vectors span the column space of A while for the last $n - r$ means that $\underline{v_i} \in N(A)$ for $i = r + 1, \ldots, n$. If we have $A^\intercal$, the decomposition is $A^\intercal = (U\Sigma V^\intercal)^\intercal = V\Sigma^\intercal U^\intercal$.

8.1 Economy SVD

What we've seen so far is the full SVD, but it can be optimized. Here is following the compact (reduced) representation, where once again we consider $m > n$:

$$\underbrace{\begin{bmatrix} & \\ & \\ & \\ & \end{bmatrix}}_{m \times n} \underbrace{\begin{bmatrix} & \\ & \end{bmatrix}}_{n \times n} \underbrace{\begin{bmatrix} & \\ & \end{bmatrix}}_{n \times n}$$

This is caused by the fact that the last $m - n$ rows in the central matrix are all 0 so multiply them for the last $m - n$ columns of the left matrix is useless. This can be furthermore optimized by having matrix dimensions: $(m \times r)(r \times r)(r \times n)$ because not all σ might be different than 0 (i.e. the rank of A is r), so, in that case is useless even to multiply the last $m - r$ rows of the central matrix.

The SVD works for any matrix A.

Let's suppose A is full rank $n \times n$:

$$A = U\Sigma V^\intercal = \sum_{i=1}^{n} \sigma_i \underbrace{\underline{u_i}\underline{v_i}^\intercal}_{\text{rank} = 1}$$

View matrix-matrix multiplication for the rank 1 concept. If A is not full rank but instead has rank$(A)=r$, the same sum is nomore computed until n, but instead r.

$$A = \sum_{i=1}^{r} \sigma_i \underline{u_i}\underline{v_i}^\intercal$$

What happens now if we pick a certain value $\tilde{r} < r$?

$$A = U\Sigma V^\intercal \cong \sum_{i=1}^{\tilde{r}} \sigma_i \underline{u_i}\underline{v_i}^\intercal$$

We obtain a **rank $\tilde{r}$ approximation of the matrix A**. The rank of the matrix is know because is the sum of $\tilde{r}$ matrices of rank 1. Moreover, that one, is the best approximation of rank $\tilde{r}$ possible, i.e.:

$$||A - \tilde{A}|| \leq ||A - B|| \qquad \forall B \text{ of rank} = \tilde{r}$$

8.2 Proof of the existence of SVD

Once again, we start from matrix $A \in \mathbb{R} \times \mathbb{R}$ an rank= r. We consider the new matrix $A^\mathsf{T}A$ which is:

- symmetric: $(A^\mathsf{T}A)^\mathsf{T} = A^\mathsf{T}A$

- positive definite: $\underline{x}^\mathsf{T}(A^\mathsf{T}A)\underline{x} = (\underline{x}^\mathsf{T}A^\mathsf{T})(A\underline{x}) = (A\underline{x})^\mathsf{T}(A\underline{x}) = ||A\underline{x}||^2 \geq 0$

We can use the following decomposition:

$$A^\mathsf{T}A = V\Lambda V^\mathsf{T} = \sum_{i=1}^{n} \lambda_i \underline{v_i}\underline{v_i}^\mathsf{T}$$

Recall that V contains the eigenvectors while Λ contains the eigenvalues. We rename $\lambda_i = \sigma_i^2$. The rank of $A^\mathsf{T}A$ is r.

We want to prove that if $\underline{x} \in N(A)$ then $\underline{x} \in N(A^\mathsf{T}A)$, to do so we proceed in both directions:

1. If we have $A\underline{x} = 0 \implies \underline{x} \in N(A)$. Is it possible to multiply both terms:

$$A^\mathsf{T}(A\underline{x}) = A^\mathsf{T}\underline{0} = \underline{0} \qquad \text{so} \qquad \underline{x} \in N(A) \implies \underline{x} \in N(A^\mathsf{T}A)$$

2. We start from $(A^\mathsf{T}A)\underline{x} = 0 \implies \underline{x} \in N(A^\mathsf{T}A)$. Again, we multiply:

$$\underline{x}^\mathsf{T}A^\mathsf{T}A\underline{x} = ||A\underline{x}||^2 = 0 \qquad \text{so} \qquad \underline{x} \in N(A^\mathsf{T}A) \implies \underline{x} \in N(A)$$

Let's consider the couple of (eigenvalues, eigenvectors) $= (\sigma_i^2, \underline{v_i})$:

$$A^\mathsf{T}A\underline{v_i} = \sigma_i^2\underline{v_i} \quad \overset{\text{component-wise}}{\longrightarrow} \quad A^\mathsf{T}A\underline{v_i} = \sigma_i^2\underline{v_i} \qquad (\dagger)$$

We introduce the quantity $\underline{u_i} = \dfrac{A\underline{v_i}}{\sigma_i}$ which has some characteristics:

i $\underline{u_i}$ are unitary vectors:

$$\underline{u_i}^\mathsf{T}\underline{u_i} = \left(\frac{A\underline{v_i}}{\sigma_i}\right)^\mathsf{T}\left(\frac{A\underline{v_i}}{\sigma_i}\right) = \frac{\underline{v_i}^\mathsf{T}A^\mathsf{T}A\underline{v_i}}{\sigma^2} \overset{\dagger}{=} \frac{\sigma_i^2\underline{v_i}^\mathsf{T}\underline{v_i}}{\sigma_i^2} = 1$$

The last passage of the equation is true because $\underline{v_i}$ vectors are orthonormal.

ii $\underline{u_i} \perp \underline{u_j}$:

$$\underline{u_i}^\mathsf{T}\underline{u_j} = \left(\frac{A\underline{v_i}}{\sigma_i}\right)^\mathsf{T}\left(\frac{A\underline{v_j}}{\sigma_j}\right) = \frac{\underline{v_i}^\mathsf{T}A^\mathsf{T}A\underline{v_j}}{\sigma_i\sigma_j} \overset{\dagger}{=} \frac{\sigma_j^2\underline{v_i}^\mathsf{T}\underline{v_j}}{\sigma_i\sigma_j} = 0$$

iii $\underline{u_i}$ are eigenvectors of AA^T with eigenvalues σ_i^2:

$$(AA^\mathsf{T}\underline{u_i}) = AA^\mathsf{T}\left(\frac{A\underline{v_i}}{\sigma_i}\right) = A\frac{A^\mathsf{T}A\underline{v_i}}{\sigma_i} \overset{\dagger}{=} A\frac{\sigma_i^2\underline{v_i}}{\sigma_i} = \sigma_i^2\left(\frac{A\underline{v_i}}{\sigma_i}\right) = \sigma_i^2\underline{u_i}$$

We have demonstrated that $A\underline{u_i} = \sigma_i\underline{u_i}$ and $\underline{u_i}$ are orthonormal as well.

We have seen that $u_i = \frac{Av_i}{\sigma_i}$ but, what happen if $\sigma_i = 0$?

Until now we have assumed that could not happen. Let's 2 examples, in the first we have the standard case while in the second is explained how to manage the case $\sigma_i = 0$.

Example 1

$$A = \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix} \qquad \text{rank}(A) = 2$$

We build two new matrices X and Y:

$$X = A^\mathsf{T} A = \begin{bmatrix} 25 & 7 \\ 7 & 25 \end{bmatrix} \qquad \text{eigs}(X) = \begin{cases} \lambda_1 = 18 & v_1 = \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix} \\ \lambda_2 = 32 & v_2 = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix} \end{cases}$$

$$Y = AA^\mathsf{T} = \begin{bmatrix} 32 & 0 \\ 0 & 18 \end{bmatrix} \qquad \text{eigs}(Y) = \begin{cases} \lambda_1 = 18 & u_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ \lambda_2 = 32 & u_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{cases}$$

So we have all elements for contructing the SVD:

$$U = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sqrt{18} & 0 \\ 0 & \sqrt{32} \end{bmatrix} \qquad V = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

It is easy to verify that $A = U\Sigma V^\mathsf{T}$.

Example 2

$$A = \begin{bmatrix} 4 & 3 \\ 8 & 6 \end{bmatrix} \qquad \text{rank}(A) = 1$$

As in the example before, we start building our matrices X and Y:

$$X = A^\mathsf{T} A = \begin{bmatrix} 80 & 60 \\ 60 & 45 \end{bmatrix} \qquad \text{eigs}(X) = \begin{cases} \lambda_1 = 125 & v_1 = \begin{bmatrix} \frac{-4}{5} \\ \frac{-3}{5} \end{bmatrix} \\ \lambda_2 = 0 & v_2 = ? \end{cases}$$

How can we compute the value of v_2? The idea is to bild a vector v_2 such that it is orthogonal to v_1. A possible solution is: $v_2 = \begin{bmatrix} \frac{3}{5} \\ \frac{-4}{5} \end{bmatrix}$.

The equation at the beginning of this page (the one regarding u_i) can still be computed for the vector v_1 in this case:

$$u_1 = \frac{Av_1}{\sigma_1} = \frac{1}{\sqrt{125}} \begin{bmatrix} 4 & 3 \\ 8 & 6 \end{bmatrix} \begin{bmatrix} \frac{-4}{5} \\ \frac{-3}{5} \end{bmatrix} = \frac{1}{\sqrt{125}} \begin{bmatrix} -5 \\ -10 \end{bmatrix}$$

Now, the problem rises again because $\sigma_2 = 0$ so we cannot compute u_2. In reality, we do not need to compute u_2 because u_1 and v_1 are sufficient to recover the original matrix through (reduced) SVD, indeed:

$$\underbrace{\frac{1}{\sqrt{125}} \begin{bmatrix} -5 \\ -10 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \sqrt{125} \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} \frac{-4}{5} & \frac{-3}{5} \end{bmatrix}}_{V} = \begin{bmatrix} 4 & 3 \\ 8 & 6 \end{bmatrix}$$

We could have used also the full SVD with the following matrices:

$$U = \frac{1}{\sqrt{125}} \begin{bmatrix} -5 & -10 \\ -10 & 5 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sqrt{125} & 0 \\ 0 & 0 \end{bmatrix} \qquad V = \begin{bmatrix} \frac{-4}{5} & \frac{-3}{5} \\ \frac{-3}{5} & \frac{4}{5} \end{bmatrix}$$

Where u_2 is the second column of the matrix U and is a vector orthogonal to u_1. All elements written here and not in the previous formulation are a waste of memory. A concise recap of all versions of SVD:

1. full SVD: $\underset{(m\times n)}{A} = \underset{(m\times m)}{U} \underset{(m\times n)}{\Sigma} \underset{(n\times n)}{V^\mathsf{T}}$

2. economy SVD: $\underset{(m \times n)}{A} = \underset{(m \times n)}{U} \underset{(n \times n)}{\Sigma} \underset{(n \times n)}{V^\mathsf{T}}$

3. reduced SVD: $\underset{(m \times n)}{A} = \underset{(m \times r)}{U} \underset{(r \times r)}{\Sigma} \underset{(r \times n)}{V^\mathsf{T}}$

4. truncated SVD approximation: $\sum_{i=1}^{\tilde{r}} \sigma_i \underline{u_i} \underline{v_i^\mathsf{T}}$

For the first two cases the rank of A is n, for the third is r while in the first the approximation rank is decided.

8.3   Geometrical interpretation of SVD

Matrices U, V^T are orthonormal so, as mentioned in the section above, they represent a rotation (or reflection) while Σ, being a diagonal matrix, correspond to a scaling transformation.
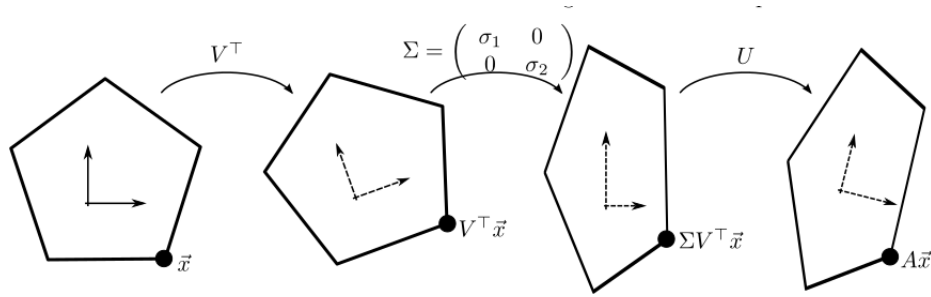
Since for SVD no assumptions are made on the starting matrix, this means that <u>any</u> matrix can be obtained from 2 rotations and 1 scaling.

When A is squared and symmetric ($A \in \mathbb{R}^{n \times n}, A = A^\mathsf{T}$)?
We can apply **polar decomposition**:
$$A = QR$$

Where Q is orthogonal and S is symmetric positive semi-definite. Why? From SVD.

$$A = U\Sigma V^\mathsf{T} = \underbrace{(UV^\mathsf{T})}_{Q}\underbrace{(V\Sigma V^\mathsf{T})}_{S}$$

In particular, the product of the two orthogonal matrices in the first parenthesis is always another orthogonal matrix. In this case of decomposition, matrix A is obtained just by one rotation and one scaling (missing one more rotation with respect to classic SVD). Indeed, the second parenthesis results just in a stretch because V and $V^i ntercal$ are two rotations identical and opposite, i.e. they cancels out.

8.3.1   Properties of SVD

i If A is orthogonal, then $\sigma_i = 1$ because if A orthogonal then $A^\mathsf{T} A = I$

ii All eigenvalues of a square matrix are $\leq \sigma_1$. Proof:

$$||A\underline{x}|| = ||U\Sigma V^\mathsf{T}\underline{x}|| = ||\Sigma V^\mathsf{T}\underline{x}|| \leq \sigma_1||V^\mathsf{T}\underline{x}|| = \sigma_1||\underline{x}|| \implies ||A\underline{x_i}|| \leq \sigma_1||\underline{x_i}||$$

The matrix U disappear becuase an orthogonal matrix, when multiplied, does not change the magnitude of a vector. If we consider
$$||A\underline{x_i}|| = ||\lambda_i \underline{x_i}|| = |\lambda_i| \cdot ||\underline{x_i}||$$
$$|\lambda_i| \cdot ||\underline{x_i}|| \leq \sigma_1||\underline{x_i}|| \implies |\lambda_i| \leq \sigma_1 \quad \forall i$$

8.4   Snapshots method

During real case scenarios, we will have a certain matrix $A \in \succ \times \ltimes$ and it might happen that $m \gg n$ i.e. the number of samples is much greater than the number of features. In these cases, we can use the following trick to be more efficient.

For the SVD we need to compute both AA^T and $A^\mathsf{T} A$. Which one is better to start with? And why? We would have $AA^\mathsf{T} : (m \times m)$ and $A^\mathsf{T} A : (n \times n)$. Given $m \gg n$ it is clear that the second one is better to start with becuase, being much smaller, it will be easier to computer its eigenvalues and eigenvectors.

8.5 Matrix norms

This concept is an extension of the vector norm. Given a matrix $A \in \mathbb{R}^{m \times n}$, we define the matrix norm as:

- **Frobenius norm**: $||A||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{\text{Tr}(A^\intercal A)} = \sqrt{\text{Tr}(AA^\intercal)}$

  Recall that $\text{Tr}(AB) = \text{Tr}(BA)$.

  Example:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \qquad ||A||_F = \sqrt{1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2} = \sqrt{91}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 1^2 + 2^2 & \cdots & \cdots \\ \cdots & 3^2 + 4^2 & \cdots \\ \cdots & \cdots & 5^2 + 6^2 \end{bmatrix} \implies \sqrt{\text{Tr}(AA^\intercal)} = \sqrt{91}$$

  If U is orthogonal, what happen to $||AU||_F^2$?

$$||AU||_F^2 = \text{Tr}((AU)^\intercal AU) = \text{Tr}(U^\intercal A^\intercal AU) = \text{Tr}(A^\intercal A \underbrace{UU^\intercal}_{I}) = \text{Tr}(A^\intercal A) = ||A||_F^2$$

  This means that the Frobenius norm is invariant with respect to orthogonal transformations (†).

  The Frobenius norm is also equal to $\left(\sqrt{\sum_{i=1}^{r} \sigma_i^2} \right)$ where r is the rank of A. Proof:

$$||A||_F = ||U\Sigma V^\intercal||_F \overset{\dagger}{=} ||\Sigma||_F \triangleq \text{Tr} \sqrt{(\Sigma \Sigma^\intercal)} = \sqrt{\sum_{i=1}^{r} \sigma_i^2}$$

- **P-norms**: Recall the p-norm for vectors:

$$\underline{p} \in \mathbb{R}^n \implies ||\underline{p}||_p = \left(\sum_{i=1}^{n} |p_i|^p \right)^{\frac{1}{p}}$$

  Given $A \in \mathbb{R}^{m \times n}$ we can define the p-norm for that matrix as:

$$||A||_p = \sup_{\underline{x} \in \mathbb{R}^n} \frac{||A\underline{x}||_p}{||\underline{x}||_p} = \sup_{\underline{x} \in \mathbb{R}^n, \, ||\underline{x}||_p} ||A\underline{x}||_p$$

  If you choose $p = 2$ you get the operator norm and $||A||_2 = \sigma_1$.

8.6 Eckart-Young theorem

Having defined the norms, we can now proceed with the proof of the **Eckart-Young theorem** which states:

For either $|| \cdot ||_F$, $|| \cdot ||_2$ we have:

$$||A - A_k|| \leq ||A - B|| \qquad \forall B \text{ of rank } k$$

where $A_k = \sum_{i=1}^{k} \sigma_i \underline{u_i v_i}^\intercal$ i.e. is the SVD approximation of rank k of the matrix A. Depending on the chosen norm you get:

$$||A - A_k|| = \begin{cases} \sigma_{k+1} \text{ if } || \cdot ||_2 \text{considered} \\ \left(\sum_{i=1}^{r} \sigma_i^2 \right)^{\frac{1}{2}} \text{ if } || \cdot ||_F \text{ considered} \end{cases}$$

There will be 2 proofs, one for each type of norm.

8.6.1 Proof considering $||\cdot||_F$

We start from the **Weyl inequality**:

$$\sigma_{i+j-1}(X+Y) = \sigma_i(X) + \sigma_j(Y)$$

Where a generic $\sigma_k(E)$ is the k-eiths singular value of the matrix E. We define

$$X = A_k - B \qquad\qquad Y = B$$

So we have

$$\sigma_{i+k}(A) \leq \sigma_i(A-B) + \underbrace{\sigma_{k+1}(B)}_{0}$$

The last component has value of 0 because the matrix B has rank equal to k.

$$||A-A_k||_F^2 = \left(\sum_{i=k+1}^{r}\sigma_i^2(A)\right) \overset{\text{shift}}{\underset{\downarrow}{=}} \left(\sum_{i=1}^{r-k}\sigma_{i+k}^2(A)\right) \leq \left(\sum_{i=1}^{r-k}\sigma_{i+k}^2(A-B)\right) \leq \left(\sum_{i=1}^{\min(m,n)}\sigma_{i+k}^2(A-B)\right) = ||A-B||_F^2$$

In particular, recall that $A = \sum\limits_{i=1}^{r}\sigma_i \underline{u_i}\underline{v_i}^\intercal$ and $A_k = \sum\limits_{i=1}^{k}\sigma_i\underline{u_i}\underline{v_i}^\intercal$. The theorem is proved just by picking the first and last components of the inequality written above.

8.6.2 Proof considering $||\cdot||_2$

We are now going to prove the theorem with the last norm we did not use yet. Let's consider the matrix B with dimensions $n \times d$ and rank(B)=k. This means that:

$$N(B) \subset \mathbb{R}^d \qquad dim(N(B)) = d - k$$

If we consider matrix V of the SVD decomposition of A:

$$V_{k+1} = \underbrace{\begin{bmatrix} | & | & & | \\ \underline{v_1} & \cdots & \underline{v_{k+1}} \\ | & | & & | \end{bmatrix}}_{k+1\,\text{cols}}$$
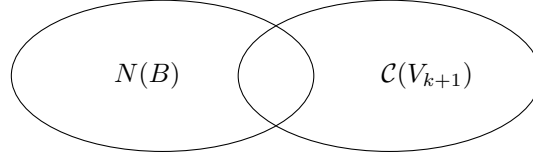
Those are the $k + 1$ columns of V.

$$\mathcal{C}(V_{k+1}) \subset \mathbb{R}^d \qquad dim(\mathcal{C}(V_{k+1})) = k + 1$$

By adding the previous information, we have:

$$dim(N(B)) + dim(\mathcal{C}(V_{k+1})) = d - k + k + 1 = d + 1$$

Since both spaces are subset of $\mathbb{R}^d$ and their summed dimensions are $d+1$ this means that the intersection of those spaces is not empty.

Consider $\underline{w} \in N(B) \bigcap \mathcal{C}(V_{k+1})$, we suppose for easyness that $||\underline{w}||_2 = 1$.

$$\underline{w} = \sum_{i=1}^{k+1} c_i \underline{v_i} \overset{(\star)}{=} V_{k+1}\underline{c} \qquad \sum_{i=1}^{k+1} c_i^2 = 1$$

We want to measure the following quantity:

$$||A - B||_2^2 = \underbrace{\sup_{||\underline{w}||}||(A - B)\underline{w}||_2^2}_{\text{particular } ||\underline{w}||} \geq \underbrace{||(A - B)\underline{w}||_2^2}_{\text{generic } ||\underline{w}||} \implies$$

Recall that $\underline{w} \in N(B) \implies B\underline{w} = 0$.

$$\implies ||A - B||_2^2 \geq ||A\underline{w}||_2^2 = \underline{w}^{\mathsf{T}} A^{\mathsf{T}} A \underline{w} \overset{\text{SVD}}{=} \underline{w} V \Sigma^{\mathsf{T}} \underbrace{U^{\mathsf{T}} U}_{I} \Sigma V^{\mathsf{T}} \underline{w} =$$

$$?\underline{w}^{\mathsf{T}} V \Sigma^{\mathsf{T}} \Sigma V^{\mathsf{T}} \underline{w} \overset{(\star)}{=} \underline{c}^{\mathsf{T}} \underbrace{V_{k+1}^{\mathsf{T}} V_{k+1}}_{I} \Sigma^{\mathsf{T}} \Sigma \underbrace{V_{k+1}^{\mathsf{T}} V_{k+1}}_{I} \underline{c} = \underline{c}^{\mathsf{T}} \Sigma^{\mathsf{T}} \Sigma \underline{c} = \sum_{i=1}^{k+1} c_i^2 \sigma_i^2 \geq$$

Since singular values are ordered

$$\geq \sigma_{k+1}^2 \underbrace{\sum_{i=1}^{k+1} c_i^2}_{} = \sigma_{k+1}^2$$

So

$$||A - B||_2^2 \geq \sigma_{k+1}^2 = ||A - A_k||_2^2$$

And, erasing the squares:

$$||A - B||_2^2 \geq \sigma_{k+1} = ||A - A_k||_2$$

Where A_k is the rank k truncated SVD approximation, therefore

$$||A - A_k||_2 \leq ||A - B||_2 \qquad \forall B \text{ of rank } k$$

9 PCA

Here is a first real-world application of the SVD. PCA has the same aim as SVD i.e. find a way of projecting the dataset in a new space where variances are maximized and covariances are minimized.

We start from $A \in \mathbb{R}^{n \times d}$ and we follow these points:

i **Center the matrix A**

$\bar{A}$ is the mean centered with respect to columns while H is called the centering matrix and is obtained as follows:

$$H = I_n - \frac{1}{n} \mathbb{1}_n \mathbb{1}_n^{\intercal}$$

Where $\mathbb{1}_n$ is the vector of dimension n containing all ones. The centered matrix is obtained:

$$\bar{A} = HA$$

ii Build the covariance matrix

$$S = \frac{\bar{A}^{\intercal} \bar{A}}{n-1}$$

Where the denominator is $n-1$ is because we want an unbiased estimator and it's not n because we have already taken 1 degree of freedom by centering the matrix. The covariance matrix is semidefinite positive so we can use eigenvalues and eigenvectors decomposition.

$$SV = VD \implies VDV^{\intercal}, D = V^{\intercal}SV$$

If you order the eigenvalues in decreasing order the corresponding eigenvectors are called principal components. To notice the relationship between SVD and PCA we can write:

$$S = \frac{1}{n-1}\bar{A}^{\intercal}\bar{A} = \frac{1}{n-1}V\Sigma^{\intercal}U^{\intercal}U\Sigma V^{\intercal} = \frac{1}{n-1}V\Sigma^2 V^{\intercal}$$

$$D = \frac{1}{n-1}\Sigma^2 \implies \lambda_k \frac{\sigma_k^2}{n-1}$$

PCA is SVD applied to a particular matrix.

9.1 Choose rank of truncated SVD

When using truncated SVD, how to choose the rank k? One possibility is to use k such that a predefined percentage of the variance is retained. Another idea starts from:

$$A = A_{true} + \gamma A_{noise}$$

Where:

- A: is out dataset

- A_{true}: is the underlying low-rank representation of our data

- γ: magnitude of the noise

- A_{noise}: is a gaussian noise with 0 mean and unitary variance

By defining τ as threshold we have that if $sigma_i > \tau$ we are picking A_{true}. There are two cases:

- γ is known, i.e. we know the magnitude of the noise:

  - if $A \in \mathbb{R}^{n \times n}$ (square) then $\tau = \frac{4}{\sqrt{3}} \gamma \sqrt{n}$

  - if $A \in \mathbb{R}^{m \times n}$ we have two more cases:
    * if $n \ll m \implies \tau = \lambda(\beta)$ where $\beta = \frac{n}{m}$ and λ is the following function:

$$\lambda(\beta) = \sqrt{2(\beta+1) + \frac{8\beta}{(\beta+1) + \sqrt{(\beta^2 + 14\beta + 1))}}}$$

* if $m \ll n \implies \tau = \lambda(\beta)$ where $\beta = \dfrac{m}{n}$ so it's equal as before but in this case the numerator and denumerator of β are swapped.

- γ is unknown. We define τ as follows:

$$\tau = \omega(\beta)\sigma_{med} \qquad \omega(\beta) = \frac{\lambda(\beta)}{\mu_\beta}$$

In which σ_{med} is the median of the singular values and μ_β is the median of the Marcenko-Pastur distribution. $\lambda(\beta)$ is the same function as before. In particular:

$$\mu_\beta = \int_{(1-\beta)^2}^{\mu_\beta} \frac{\sqrt{((1-\sqrt{\beta})^2 - t)(t - (1 - \sqrt{\beta})^2)}}{2\pi t} dt = \frac{1}{2}$$

9.2   Randomize SVD

10   Least squares approximation

Consider, with $n > p$:

$$X \in \mathbb{R}^{n \times p} \qquad n = \# \text{ samples}, \; p = \# \text{ features}, \; \text{rank}(X) = p$$

$$\underline{y} \in \mathbb{R}^n \qquad \text{labels of each sample}$$

We have:

$$\underbrace{\begin{bmatrix} - & x_1^\mathsf{T} & - \\ - & x_2^\mathsf{T} & - \\ & \vdots & \\ - & x_n^\mathsf{T} & - \end{bmatrix}}_{p} \qquad \underbrace{x_j}_{} : \text{j-th column of } X$$

If we provide a new sample $\underline{\tilde{x}} \to \tilde{y}$ we want to predict $\tilde{y}$, i.e. the label, using the information we have from the training set.

We can use a linear model:

$$\tilde{y} = \underline{\tilde{x}}^\mathsf{T} \underline{w} \qquad \underline{w} \in \mathbb{R}^p$$

Typically $\underline{y} \neq X\underline{w}$ so $\underline{y}$ won't be precisely obtained with that multiplication. We will have to the approximation: $\underline{\hat{y}} = X\underline{w}$. So we can say that $\underline{\hat{y}} \in \mathcal{C}(X)$ while in general $\underline{y} \notin \mathcal{C}(X)$. The error of the prediction is given by:
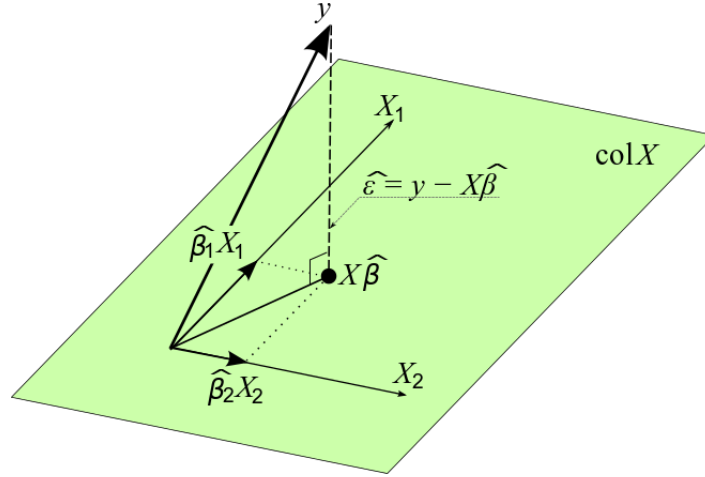
$$r_i(\underline{w}) = \underline{y}_i - \underline{\hat{y}}_i$$

We define the **residual vector** as $\underline{r}(\underline{w})$ and our goal is to minimize its l2-norm squared: $||\underline{r}(\underline{w})||_2^2$.

Mathematically we can describe the problem as finding: $\underline{\hat{w}} = \arg\min_{\underline{w}} ||\underline{r}(\underline{w})||_2^2$. There are two approaches possible:

1. Geometrical interpretation

2. By means of optimization procedures

10.1   Geometrical interpretation

So, a brief description of the figure above:

- the plane is the space of the predictions $\mathcal{C}(X)$ indeed it is spanned by the column vectors of X

- the vector $\underline{y}$ is the vector of the label to be predicted

- the vector $X\hat{\beta}$ in the figure is our prediction $\underline{\hat{y}}$ and correspond to the projection of $\underline{y}$ on the plane $\mathcal{C}(X)$

- ϵ is the residual vector $\underline{r}(\underline{w})$ and represent the error between the predicted value and the actual label

If we pick $\overline{y} \in \mathcal{C}(X)$ different from $\underline{\hat{y}}$ and their respective residuals: $\overline{r} = \underline{y} - \overline{y}$ and $\hat{r} = \underline{y} - \hat{y}$ we have that:

$$||\hat{r}||_2 \leq ||\overline{r}||_2$$

Indeed, $\hat{\underline{r}}$ is the orthogonal projection of $\underline{y}$ on $\mathcal{C}(X)$ and the orthogonal projection is the closest point to the vector $\underline{y}$ in the subspace $\mathcal{C}(X)$.

Because of this, we can also say that:

$$\underline{x}_j^\mathsf{T}\hat{\underline{r}} = 0 \quad j = 1, \dots, p$$

in matrix form:

$$X^\mathsf{T}\hat{\underline{r}} = \underline{0} \implies X^\mathsf{T}(\underline{y} - \hat{\underline{y}}) = \underline{0} \implies X^\mathsf{T}(\underline{y} - X\hat{\underline{w}}) = \underline{0} \implies X^\mathsf{T}\underline{y} = X^\mathsf{T}X\hat{\underline{w}}$$

In general the rank of the matrix $X^\mathsf{T}X$ is the same as the rank of X so if X is full rank then also $X^\mathsf{T}X$ is full rank and this imply that is invertible and we can find $\hat{\underline{w}}$ like this:

$$\hat{\underline{w}} = (X^\mathsf{T}X)^{-1}X^\mathsf{T}\underline{y}$$

If we are using this linear model for a binary classification we have to apply to the predicted value $\hat{\underline{y}}$ a sign function in order to have the output restrained to {-1,1}.

10.2 Optimization

Here we exploit mathematics for solving the initial problem:

$$\hat{\underline{w}} = \arg\min_{\underline{w}}||\underline{r}(\underline{w})||_2^2 = \arg\min_{\underline{w}}||\underline{y} - X\underline{w}||_2^2 = \arg\min_{\underline{w}}(\underline{y} - X\underline{w})^\mathsf{T}(\underline{y} - X\underline{w}) =$$

$$= \arg\min_{\underline{w}}\left[\underline{y}^\mathsf{T}\underline{y} - (X\underline{w})^\mathsf{T}\underline{y} - \underline{y}^\mathsf{T}(X\underline{w}) + (X\underline{w})^\mathsf{T}X\underline{w}\right] = \arg\min_{\underline{w}}\underbrace{\left[\underline{y}^\mathsf{T}\underline{y} - 2\underline{y}^\mathsf{T}X\underline{w} + \underline{w}^\mathsf{T}X^\mathsf{T}X\underline{w}\right]}_{F(\underline{w})}$$

$F(\underline{w})$ is a **quadratic functional**. X is full rank, $X^\mathsf{T}X$ is positive definite so this means that the functional is strictly convex and has a unique minimum. So we can compute the gradient and set it to zero:

$$\nabla_{\underline{w}}F(\underline{w}) = -2X^\mathsf{T}\underline{y} + 2X^\mathsf{T}X\underline{w} = \underline{0}$$

Example 1 - Derivation

$$F(\underline{w}) = \underline{w}^\mathsf{T}\underline{c} = w_1c_1 + w_2c_2 + w_3c_3 = \underline{c}\underline{w}^\mathsf{T} \qquad \underline{w}, \underline{c} \in \mathbb{R}^3$$

The gradient is:

$$\nabla F = \begin{bmatrix} \dfrac{\partial F}{\partial w_1} \\ \dfrac{\partial F}{\partial w_2} \\ \dfrac{\partial F}{\partial w_3} \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \underline{c}$$

Example 2 - Derivation

$$F(\underline{w}) = \underline{w}^\mathsf{T}A\underline{w} = \sum_{i=1}^{n}\sum_{j=1}^{n}w_j a_{ij} w_i \qquad A \in \mathbb{R}^{n \times n}$$

$$\frac{\partial}{\partial w_k}(w_j a_{ij} w_i) = \begin{cases} a_{ij}w_i & k = j \neq i \\ w_j a_{ij} & k = i \neq j \\ 0 & k \neq i, k \neq j \\ 2w_k a_{ij} & k = i = j \end{cases}$$

In the last lecture we have introduced the least squares method. In particular we have mentioned the linear model for which:

$$\hat{\underline{y}} = X\hat{\underline{w}} \qquad X \in \mathbb{R}^{n \times p} \quad \hat{\underline{y}} \in \mathbb{R}^n \quad n \geq p \quad X \text{ full rank}$$

We have obtained:

$$\hat{\underline{w}} = (X^\intercal X)^{-1} X^\intercal \underline{y} \implies \hat{\underline{y}} = \underbrace{X(X^\intercal X)^{-1} X^\intercal}_{P_x} \underline{y}$$

The matrix P_x dimension is given by the product of: $(n \times p)(p \times p)(p \times n) = (n \times n)$ and has this properties:

- $P_x = P_x^2$

- P_x is a projection matrix

Let's consider U an orthogonal ($U^\intercal U = I$) matrix that contains the basis for $\mathcal{C}(X)$ this means that $\mathcal{C}(X) = \mathcal{C}(U)$. We can write:

$$\hat{\underline{y}} = X\hat{\underline{w}} = U\tilde{\underline{w}}$$

So this basically means that the predicted value of y still a projection on a plane but this time the plane is spanned by the columns of U and not by the columns of X. By substituting last equation in the minimization method for least squares we have:

$$\tilde{\underline{w}} = \arg\min_{w} ||\underline{y} - U\underline{w}||_2^2 \implies \hat{\underline{y}} = U\tilde{\underline{w}} = U(U^\intercal U)^{-1} U^\intercal \underline{y} = UU^\intercal \underline{y}$$

This formulation is possible because this time in the parenthesis we have an orthogonal matrix and this means that $(U^\intercal U)^{-1} = U^\intercal U = I$. In general $UU^\intercal \neq I$ because it might be rectangular (while $U^\intercal U$ is always square).

Example of usage of U We start from X:

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 0 & 0 \end{bmatrix} \qquad \underline{x_1} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \qquad \underline{x_2} = \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}$$

How do we build the orthogonal matrix U? We can use the Gram-Schmidt procedure:

$$\underline{u_1} = \frac{\underline{x_1}}{||\underline{x_1}||} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$$

$$\underline{x'_2} = \underline{x_2} - (\underline{x_2}^\intercal \underline{u_1})\underline{u_1} = \underline{x_2} - (\underline{u_1}^\intercal \underline{u_1})\underline{x_2} \implies \underline{u_2} = \frac{\underline{x'_2}}{||\underline{x'_2}||} = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$$

So, the overall matrix U is:

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 \end{bmatrix} \qquad U^\intercal U = I \text{ and } UU^\intercal \neq I$$

A drawback of Gram Schmidt is that, depending on the order chosen for the columns of X, the matrix U can be different. Moreover, the order of vector columns of U is meaningless.

Now, we want to exploit the SVD for computing the orthogonal matrix U. We start from the SVD of X:

$$X = U\Sigma V^\intercal$$

So

Recall that:

$$\hat{\underline{w}} = (X^\intercal X)^{-1} X^\intercal y$$
$$= (V\Sigma^\intercal U^\intercal U\Sigma V^\intercal)^{-1} V\Sigma^\intercal U^\intercal \underline{y}$$
$$= (V\Sigma^\intercal \Sigma V^\intercal)^{-1} V\Sigma^\intercal U^\intercal \underline{y}$$
$$= V(V\Sigma^\intercal \Sigma V^\intercal)^{-1} V\Sigma^\intercal U^\intercal \underline{y}$$
$$= V(\Sigma^\intercal \Sigma)^{-1} \underbrace{V^\intercal V}_{I} \Sigma^\intercal U^\intercal \underline{y}$$
$$= V \underbrace{(\Sigma^\intercal \Sigma)^{-1} \Sigma^\intercal}_{\Sigma^+} U^\intercal \underline{y}$$
$$= V\Sigma^+ U^\intercal \underline{y}$$

$$(AB)^{-1} = B^{-1} A^{-1}$$
$$(V^\intercal)^{-1} = V$$

Because V is orthogonal.
Σ^+ il called the pseudo-inverse of Σ.

Eventually, we have:
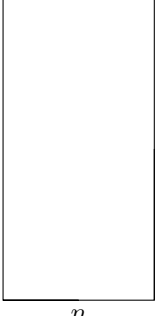
$$\hat{\underline{y}} = X(X^\intercal X)^{-1} X^\intercal \underline{y} = XX^+ \underline{y}$$
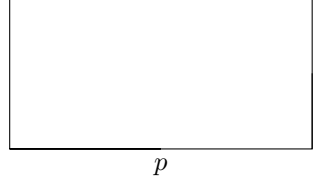$$= U(U^\intercal U)^{-1} U^\intercal \underline{y} = UU^+ \underline{y}$$

Recalling that we are considering the case in which $n \geq p$, the matrices have these dimensions: $U = (n \times n), \Sigma = (n \times p), V^\intercal = (p \times p)$ and:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_p \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix}$$

$$\Sigma^\intercal \Sigma = \begin{bmatrix} \sigma_1 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_p & 0 & \ldots & 0 \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_p \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & 0 & \ldots & 0 \\ 0 & \sigma_2^2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_p^2 \end{bmatrix}$$

$$(\Sigma^\intercal \Sigma)^{-1} = \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 & \ldots & 0 \\ 0 & \frac{1}{\sigma_2^2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \frac{1}{\sigma_p^2} \end{bmatrix}$$

$$\Sigma^+ = (\Sigma^\intercal \Sigma)^{-1} \Sigma^\intercal = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & \frac{1}{\sigma_2} & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \frac{1}{\sigma_p} & 0 & \ldots & 0 \end{bmatrix}$$

Let's consider now the case in which $p \geq n$ and X has n linearly independent rows (before we had p linearly independent columns). This means that we have more unknowns than equations and we would find infinite solutions for $\hat{\underline{w}}$ such that $\hat{\underline{y}} = X\hat{\underline{w}}$.

The solution found before $\hat{\underline{w}} = V\Sigma^+ U^\intercal \underline{y}$ it's still valid but now $\Sigma^+ = \Sigma^\intercal(\Sigma\Sigma^\intercal)^{-1}$ so it has the same shape as before but transposed. Summary:

$$n > p$$

$$p > n$$

$$n$$

$$p$$

$$\underline{\hat{w}} = V\Sigma^+ U^\intercal \underline{y}$$

$$\Sigma^+ = \Sigma^\intercal (\Sigma\Sigma^\intercal)^{-1}$$

$$\underline{\hat{w}} = V\Sigma^+ U^\intercal y$$

$$\Sigma^+ = (\Sigma^\intercal \Sigma)^{-1}\Sigma^\intercal$$

Let's consider two vectors: $\underline{w}$ and $\underline{\hat{w}}$:

$$\underline{y} = x\underline{w} = X\underline{\hat{w}}$$

So they both are solutions of the system of equations.

$$
\begin{aligned}
\star &= (\underline{w} - \underline{\hat{w}})^\intercal X^\intercal (XX^\intercal)\underline{y} \\
&= (X(\underline{w} - \underline{\hat{w}}))^\intercal (XX^\intercal)^{-1}\underline{y} \\
&= (X\underline{w} - X\underline{\hat{w}})^\intercal (XX^\intercal)^{-1}\underline{y} \\
&= (0)^\intercal (XX^\intercal)^{-1}\underline{y} \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
||\underline{w}||_2^2 &= ||(\underline{w} - \underline{\hat{w}}) + \underline{\hat{w}}||_2^2 \\
&= ||\underline{w} - \underline{\hat{w}}||_2^2 - 2\underbrace{(\underline{w} - \underline{\hat{w}})^\intercal \underline{\hat{w}}}_{\star} + ||\underline{\hat{w}}||_2^2 \\
&= \underbrace{||\underline{w} - \underline{\hat{w}}||_2^2}_{\geq 0} + ||\underline{\hat{w}}||_2^2
\end{aligned}
$$

So

$$\implies ||\underline{w}||_2^2 \geq ||\underline{\hat{w}}||_2^2$$

This means that $\underline{\hat{w}}$ has minimum norm solution, i.e. among all possible vectors w that satisfy the initial equation $\underline{y} = x\underline{w} = X\underline{\hat{w}}$, $\underline{\hat{w}}$ is the one with the minimum norm. In order to understand why that property is important, consider the following example.

Suppose to have:

$$X = \begin{bmatrix} 1 & 0 & 0,01 \\ 0 & 1 & 0 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We could choose:

$$\underline{w_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \underline{w_2} = \begin{bmatrix} 0 \\ 0 \\ 100 \end{bmatrix}$$

It is easily understandable that, if would choose the second vector as a solution instead of the first one, we would have an important amplification of the error related to the measure of the value of the column feature. The same cannot be said for the first vector.

Notice that, it is not that improbable to have more features than samples, consider for example images, for which each pixel is a feature.

11   Matrix completion

We start from

$$X \in \mathbb{R}^{n \times p} \qquad rank(X) = r \ll \min(n, p)$$

We know only partially X, we know $X_{i,j}$ for $(i, j) \in \Omega$.

Matrix completion is a method for filling missing values. If we had full rank matrix we would have very independent columns so we would not be able to retrieve/obtain missing values. Being low rank, in this sense, helps us accomplishing this goal.

11.1   Ideal estimator

$$\begin{cases} \hat{X} = \underset{z \in \mathbb{R}^{n \times p}}{\arg \min} [\ \text{rank}(z)] \\ \text{subject to } \hat{X}_{ij} = X_{ij}(i,j) \in \Omega \end{cases}$$

This formulation is computationally unfeasible and the object function is non-convex. What is in practical terms saying is that, amongs all possible solutions, we want the one with minimal rank.

How can we solve it?

11.2   Practical estimator

We are going to use the **Nuclear norm**: $|| \cdot ||_* = \sum_{i=1}^{\min(n,p)} \sigma_i$.

The idea is that, instead of minimizing the rank, we minimize the norm since, when performing the SVD, the number of non-zero singular values is equal to the rank of the matrix.

$$\begin{cases} \hat{X} = \underset{z \in \mathbb{R}^{n \times p}}{\arg \min} ||z||_* \\ \text{subject to } \hat{X}_{ij} = X_{ij}(i,j) \in \Omega \end{cases}$$

This new formulation is convex-optimal. How are we going to solve this problem? With the **SVT (Singular Value Threshold)**. Here is the algorithm:

- Initialize $\hat{X} = \texttt{zeros}(n,p)$

- Set $\hat{X}_{ij} = X_{ij} \ \texttt{for} \ (i,j) \in \Omega$

- For $k = 1, 2, \dots, N$

  - $\hat{X}_{old} = \hat{X}$
  - $[U, \Sigma, V^\intercal] = \texttt{svd}(\hat{X})$
  - $\Sigma \to \hat{\Sigma} \begin{cases} \hat{\sigma}_i = \sigma_i & \text{if } \sigma_i > \tau \\ \hat{\sigma}_i = 0 & \text{otherwise} \end{cases}$
  - $\hat{X} = U\hat{\Sigma}V^\intercal$
  - $\hat{X}_{ij} = X_{ij} \ \texttt{for} \ (i,j) \in \Omega$

- $||\hat{X} - X_{old}|| < \epsilon$

The constant τ is imposing some sort of thresholding on the acceptance of singular values. This is an example of non-monothone algorithm because after the reduction of rank with the condition on sigmas, we override some value of the matrix to the one contained in Ω. Is it proved that for a large enough index k the algorithm converge to the solution.

Let's recap what we have seen so far regarding the Least Squares method.

$$(x_i, y_i) \;\; i = 1, \ldots, n \qquad \underline{x_i} \in \mathbb{R}^p \text{ and } y_i \in \mathbb{R}$$

In matrix form:

$$X \in \mathbb{R}^{n \times p} \text{ and } \underline{y} \in \mathbb{R}^n$$

We have made the hypothesis, until now, of having a full rank matrix X. The linear model is defined as:

$$\hat{\underline{y}} = X\hat{\underline{w}}_{LS} \qquad \text{with} \qquad \hat{\underline{w}}_{LS} = (X^\intercal X)^{-1} X^\intercal \underline{y}$$

Moreover, we said that the actual value of y is not exactly given by this linear model but it will be an approximation:

$$y \approx X\underline{w}_{LS}$$

And this can be written by expliciting an error:

$$\underline{y} = X\underline{w}^* + \underline{\epsilon}$$

where $\underline{\epsilon}$ is the error or noise vector. If we replace this last equation in the one with $\hat{\underline{w}}_{LS}$ we get:

$$\hat{\underline{w}}_{LS} = (X^\intercal X)^{-1} X^\intercal (X\underline{w}^* + \underline{\epsilon}) = \underline{w}^* + (X^\intercal X)^{-1} X^\intercal \underline{\epsilon} =$$

Now we can apply the SVD to X so, if we call $X = U\Sigma V^\intercal$ we get, from previous lectures, that: $(X^\intercal X)^{-1} X^\intercal = V\Sigma^+ U^\intercal$. We have said before that X is full rank so we will have all singular components different than 0 and $\sigma_1 > \sigma_2 > \cdots > \sigma_p > 0$. The matrix Σ^+ is a psuedo-diagonal matrix with the inverse of the singular values on the diagonal (check few pages above).

$$= \underline{w}^* + V\Sigma^+ U^\intercal \underline{\epsilon}$$

- $U^\intercal \underline{\epsilon}$: we multiply a vector with an orthogonal matrix so its norm will not change.

- $V\Sigma^+ U^\intercal \underline{\epsilon}$: we multiply a vector with a diagonal matrix so we will have a scaling of the vector. But **if a singular value is very very small, its inverse in the matrix SigmaPlus will be huge and will scale the error vector a lot! So the error will be amplified and the original model vector w* will be negligible.**

11.3 Ridge regression (regularization)

This method will help us in preventing the problem mentioned just before. We start from the definition of the weight vector for linear model explicited for the optimization method of the Least Squares:

$$\hat{\underline{w}}_{LS} = \arg\min_{\underline{w}} \underbrace{||\underline{y} - X\underline{w}||_2^2 + \lambda ||\underline{w}||_2^2}_{f(\underline{w})}$$

In particular we have added a term.

$$f(\underline{w}) = \underline{y}^\intercal \underline{y} - 2\underline{w}^\intercal X\underline{y} + \underline{w}^\intercal X^\intercal X\underline{w} + \lambda \underline{w}^\intercal \underline{w}$$

We can now compute the gradient of this function:

$$\nabla_w(f(\underline{w})) = -2X^\intercal \underline{y} + 2X^\intercal X\underline{w} + 2\lambda \underline{w} = 0$$

$$X^\intercal \underline{y} = (X^\intercal X + \lambda I)\underline{w}$$

$$\hat{\underline{w}}_R = (X^\intercal X + \lambda I)^{-1} X^\intercal \underline{y}$$

It's easy to notice that if $\lambda = 0$ we get the Least Squares solution. If $\lambda > 0$ we will have a different solution. We can now compute the SVD of X:

$$\hat{\underline{w}}_R = (V\Sigma^\intercal \underbrace{U^\intercal U}_{I} \Sigma V^\intercal + \lambda \underbrace{V^\intercal V}_{I})^{-1} V\Sigma^\intercal U^\intercal \underline{y}$$

$$= [V(\Sigma^\intercal \Sigma + \lambda I)V^\intercal]^{-1} V\Sigma^\intercal U^\intercal \underline{y}$$

$$= V \underbrace{(\Sigma^\intercal \Sigma + \lambda I)\Sigma^\intercal}_{M} U^\intercal \underline{y}$$

Where

$$M = \begin{bmatrix} \dfrac{\sigma_1}{\sigma_1^2 + \lambda} & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & \dfrac{\sigma_2}{\sigma_2^2 + \lambda} & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \dfrac{\sigma_p}{\sigma_p^2 + \lambda} & 0 & \ldots & 0 \end{bmatrix}$$

- if σ_i is big compared to λ then $\dfrac{\sigma_i}{\sigma_i^2 + \lambda} \approx \dfrac{1}{\sigma_i}$

- if σ_i is close to 0 then $\dfrac{\sigma_i}{\sigma_i^2 + \lambda} \approx 0$

If we now consider again the problem of having a singular value close to 0 we can see that it is now solved because we would reconduct to the second case just above and the pseudo inverse would be almost 0. If λ is very small the matrix of Σ's is almost equal to Σ^+.

$$
\begin{aligned}
\hat{\underline{w}}_R &= (X^\intercal X + \lambda I)^{-1} X^\intercal \underline{y} \\
&= (X^\intercal X + \lambda I)^{-1} X^\intercal (X \underline{w}^* + \underline{\epsilon}) \\
&= (X^\intercal X + \lambda I)^{-1} X^\intercal X \underline{w}^* + (X^\intercal X + \lambda I)^{-1} X^\intercal \underline{\epsilon}
\end{aligned}
$$

If $\underline{\epsilon} = \underline{0}$ then there is only the first term and, since we are not finding the perfect projection on the plane, we make an higher error with respect to Least Squares.

12 Page Rank

Consider 4 websites, the arrows represents the link from one site to another.
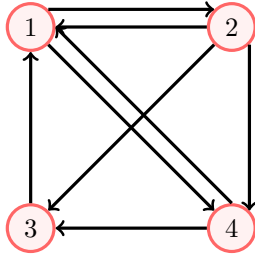
The idea is to surf the web randomly (random walks on the graph) and if you do that long enough you will reach a **Steady state** where π_i will be the probability of being on the i-th website.
In this case the vector $\underline{\pi} \in \mathbb{R}^4$ because there are 4 websites.

We are assuming there are not separated sites. How can we represent the matrix? It an **Adjacency matrix**

$$
\tilde{A} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \rightarrow \tilde{A}_{ij} = \begin{cases} 1 & \text{if there is a link from i to j} \\ 0 & \text{otherwise} \end{cases} \xrightarrow{\text{normalized version}} A = \begin{bmatrix} 0 & 1 & 1/3 & 1/2 \\ 0 & 0 & 1/3 & 1/2 \\ 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/3 & 0 \end{bmatrix}
$$

In the normalized version all columns sum up to 1. What happen if we multiply the matrix A times a canonical basis vector $\underline{e}_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^\intercal$?

$$
A \underline{e}_3 = \begin{bmatrix} 1/3 \\ 1/3 \\ 0 \\ 1/3 \end{bmatrix}
$$

As it was trivial to figure, we obtain, in this case, the third column of the adjacency matrix. In a probability perspective, the vector $\underline{e}_3$ represents the probability of starting from the third website. The vector we obtain is the probability of reaching the other websites starting from the third one. This means that in this case we are certain that we start from the third website and for sure we won't be in that site in the following iteration.

Considering

$$
\underline{\pi} = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{bmatrix}
$$

The steady state is defined as: $A \underline{\pi} = \underline{\pi}$. This means that, the steady state is the eigenvector of the matrix A with eigenvalue 1. The matrix A is positive (not positive-definite) i.e. it has all non-negative coheffients. A positive matrix is denoted with $A > 0$. From Perron-Frobenius theory we know that $\lambda_1 = 1$ is the largest eigenvalue.

$$
\lambda_1 = 1 > \lambda_2 > \lambda_3 > \cdots > \lambda_n \qquad \lambda_i \neq 0
$$

As mentioned in a previous lecture, we can use the power method in order to retrieve the largest eigenvalue. In particular, we start from:

$$\underline{\pi}^{(0)} \quad \text{with} \quad ||\underline{\pi}^{(0)}|| = 1$$

Then, for $k = 1, 2, \ldots$

$$\underline{\pi}^{(k)} = \frac{A\underline{\pi}^{(k-1)}}{||A\underline{\pi}^{(k-1)}||}$$

$$\text{if } ||\underline{\pi}^{(k)} - \underline{\pi}^{(k-1)}|| < \epsilon \text{ then stop}$$

Obviously, ϵ represents a tolerance value. As we can see, there is an recursive definition in a sense that, at each iteration, the same operation is made on the same variable. For example:

$$\underline{\pi}^{(1)} = \frac{A\underline{\pi}^{(0)}}{||A\underline{\pi}^{(0)}||} \qquad\qquad \underline{\pi}^{(2)} = \frac{A\underline{\pi}^{(1)}}{||A\underline{\pi}^{(1)}||} = \frac{A^2\underline{\pi}^{()}}{||A^2\underline{\pi}^{(k-1)}||}$$

So, iterating k times:

$$\underline{\pi}^{(k)} = \frac{A^k\underline{\pi}^{(0)}}{||A^k\underline{\pi}^{(0)}||}$$

Since in A there are no columns that are completely equal to 0, the matrix can be diagonalized. This implies that there are some $\underline{v}_i, i = 1, \ldots, n$ that can used as a basis for the space. In particular we can write:

$$\underline{\pi}^{(0)} = \sum_{i=1}^{n} \alpha_i \underline{v}_i$$

We want to plug this expression in the previous equation (power method). Before, notice that, $A\underline{v}_i = \lambda \underline{v}_i$ because $\underline{v}_i$ are the vectors which diagonalize A so the ones such that $A = V\Lambda V^{\intercal}$. The numerator of $\underline{\pi}^{(k)}$ can be written as:

$$A^k\underline{\pi}^{(0)} = \alpha_1 \lambda_1^k \left(\underline{v}_1 + \sum_{i=2}^{n} \frac{\alpha_i}{\alpha_1} \left(\frac{\lambda_i}{\lambda_1} \right)^k \underline{v}_i \right)$$

Proof:

$$
\begin{aligned}
A^k\underline{\pi}^{(0)} &= V\Lambda^k V^{-1} \left(\alpha_1 \underline{v}_1 + \cdots + \alpha_n \underline{v}_n \right) \\
&\stackrel{\circledast}{=} V\Lambda^k \left(\alpha_1 \underline{e}_1 + \cdots + \alpha_n \underline{e}_n \right) \\
&= V \left(\alpha_1 \lambda_1^k \underline{e}_1 + \cdots + \alpha_n \lambda_n^k \underline{e}_n \right) \\
&\stackrel{\diamond}{=} \alpha_1 \lambda_1^k \underline{v}_1 + \cdots + \alpha_n \lambda_n^k \underline{v}_n \\
&= \alpha_1 \lambda_1^k \left(\underline{v}_1 + \frac{\alpha_2}{\alpha_1} \left(\frac{\lambda_2}{\lambda_1} \right)^k \underline{v}_2 + \cdots + \frac{\alpha_n}{\alpha_1} \left(\frac{\lambda_n}{\lambda_1} \right)^k \underline{v}_n \right)
\end{aligned}
$$

$$\circledast \rightarrow V^{-1}\underline{v}_i = V^{-1}\underline{v}_i = \begin{bmatrix} \text{---} & \underline{v}_1^{\intercal} & \text{---} \\ \text{---} & \underline{v}_2^{\intercal} & \text{---} \\ & \vdots & \\ \text{---} & \underline{v}_n^{\intercal} & \text{---} \end{bmatrix} \underline{v}_i = \underline{e}_i$$

$$\diamond \rightarrow A\underline{v} = \lambda\underline{v} \implies AA\underline{v} = \lambda A\underline{v} \implies A^2\underline{v} = \lambda^2\underline{v}$$

If $k \to \infty$ then $\left(\frac{\lambda_i}{\lambda_1} \right)^k \to 0$ because $\lambda_1 > \lambda_i$ for $i = 2, \ldots, n$. So, the only remaining term is $\underline{v}_1$ which is the eigenvector with the largest eigenvalue. So what we have just done is the proof of the convergence of the power method. The closer are the eigenvalues to λ_1 the slower is the convergence.

So far we have considered:

$$y = X\underline{w} \qquad X \in \mathbb{R}^{n \times p} \text{ and } \left\{ \begin{array}{l} \text{Least Squares}: \hat{\underline{w}}_L S \\ \text{Ridge Regression}: \hat{\underline{w}}_{RR} \end{array} \right\} \underline{w} \in \mathbb{R}^p \text{ dense vector, i.e. not many zeros}$$

We are going to see a method which obtain a vector $\hat{\underline{w}}$ with many zeros, as sparse as possible. We want to consider this model:

$$y = X\underline{w} \qquad X \in \mathbb{R}^{n \times p} \qquad p > n$$

We have said that the system is undetermined since it has infinite solutions. Suppose to have 2 features and 1 sample.

$$X = \begin{bmatrix} 2 & 3 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \end{bmatrix}$$

And so:

$$1 = \begin{bmatrix} 2 & 3 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \implies 1 = 2w_1 + 3w_2 \quad \leftarrow \quad \text{line}$$

As mentioned in a previous lecture, we want to find the minimum length solution so we can plot the line found before and the circles that represents the l2-norm distance.

(a) $p = 2$ (b) $p = 1$
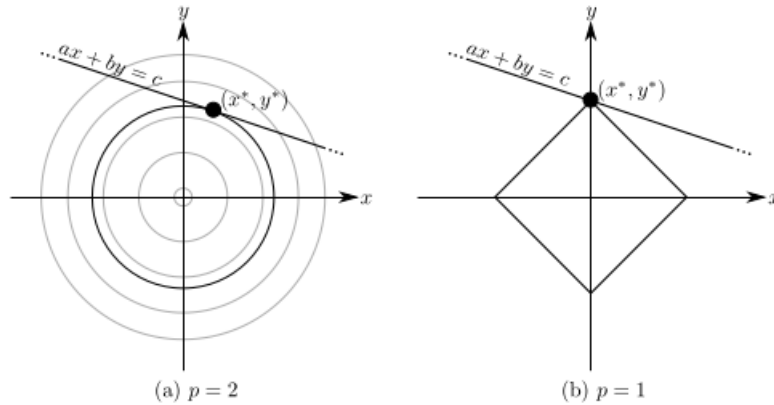
On the right-hand side it is represented the equivalent plot but with the l1-norm. We can see that the solution on the right has one coordinate (are features!) that is equal to zero. This is true in general, i.e. we will obtain sparser solution using the l1-norm rather than the l2-norm.

With L1-norm it's still a convex optimization problem and

$$F(\underline{w}) = ||X\underline{w} - \underline{y}||_2^2 + \lambda ||\underline{w}||_1$$

This model is implemented by **Lasso** (Least Absolute Shrinkage and Selection Operator) and achieve both the shortest distance solution and the selection of some features.

This is important because by reducing the number of features, we increase the interpretability of the model.

There is also the **Elastic Net** which combines both Lasso and Ridge:

$$F(\underline{w}) = ||X\underline{w} - \underline{y}||_2^2 + \lambda_1 ||\underline{w}||_1 + \lambda_2 ||\underline{w}||_2^2$$

Once again, we start considering:

$$\underline{y} \in \mathbb{R}^n \qquad X \in \mathbb{R}^{n \times p} \text{ with } p \text{ lin. ind. cols } (\implies \sigma_i > 0, \ i = 1, \dots, p)$$

Now we write the formulation of the weights vector w we found for Ridge Regression.

$$\begin{aligned} \hat{\underline{w}}_R &= V \underbrace{(\Sigma^\mathsf{T}\Sigma + \lambda I)^{-1}\Sigma^\mathsf{T}}_{\Sigma^\mathsf{T}(\Sigma\Sigma^\mathsf{T} + \lambda I)^{-1}} U^\mathsf{T}\underline{y} \\ &= \underbrace{V\Sigma^\mathsf{T}U^\mathsf{T}}_{X^\mathsf{T}} \underbrace{U(\Sigma\Sigma^\mathsf{T} + \lambda I)^{-1}U^\mathsf{T}\underline{y}}_{\underline{\alpha} \in \mathbb{R}^n} \\ &\overset{\square}{=} X^\mathsf{T}\underline{\alpha} \\ &= \sum_{i=1}^{n} \alpha_i \underline{x}_i \end{aligned}$$

In the second passage we have added the matrices $U^\mathsf{T}U$ because its the identity matrix. We obtain a weighted sum of the column vectors of X^T, where X is:

$$X = \begin{bmatrix} — & \underline{x}_1^\mathsf{T} & — \\ — & \underline{x}_2^\mathsf{T} & — \\ & \vdots & \\ — & \underline{x}_n^\mathsf{T} & — \end{bmatrix}$$

So, until now, we have discussed about linear models. A generic form would be:

$$\hat{y}_i = w_1 x_{i1} + w_2 x_{i2} \qquad \text{if} \qquad \underline{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix}$$

Now a new model:

$$\hat{y}_i = w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i1}^2 + w_4 x_{i2}^2 + w_5 x_{i1} x_{i2}$$

So, the original feature vector $\underline{x}$ is transformed into a new feature vector by means of function called feature map $\phi(x)$:

$$\phi(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ x_1 x_2 \end{bmatrix} \in \mathbb{R}^d \qquad d > p \text{ typically}$$

And

$$\hat{y}_i = \phi(x_i)^\mathsf{T} \underline{w}$$

In general d can be huge.

13 Kernel Methods

The aim of this methods is to avoid the necessity of computing huge vectors.

$$\Phi = \begin{bmatrix} — & \phi(\underline{x}_1)^\mathsf{T} & — \\ — & \phi(\underline{x}_2)^\mathsf{T} & — \\ & \vdots & \\ — & \phi(\underline{x}_n)^\mathsf{T} & — \end{bmatrix} \in \mathbb{R}^{n \times d} \qquad \hat{\underline{y}} = \Phi \underline{w}$$

The objective is still the same: i.e. finding $\underline{w}$. We are now going to consider ridge regression in order to achieve that. Instead of $\hat{\underline{w}}_R = X^\mathsf{T} \underline{\alpha}$ we can now write:

$$\hat{\underline{w}}_R = \Phi^\mathsf{T} \underline{\alpha}$$

Where

$$\underline{\alpha} = U(\Sigma\Sigma^\mathsf{T} + \lambda I)^{-1} U^\mathsf{T} \underline{y} = (XX^\mathsf{T} + \lambda I)^{-1} \underline{y}$$

Here is the proof:

$$(U\Sigma V^\mathsf{T} V \Sigma^\mathsf{T} U^\mathsf{T} + \lambda UU^\mathsf{T})^{-1} \underline{y}$$

So, by similarity with the expression obtained in the previous page ($\hat{\underline{w}}_R \overset{\square}{=} X^\mathsf{T} \underline{\alpha}$) we can write:

$$\underline{\alpha} = (\Phi\Phi^\mathsf{T} + \lambda I)^{-1} \underline{y}$$

The computation of $\underline{\alpha}$ is practically impossible since Φ is huge. How can we reduce the cost of that computation? Introducing the **Kernel Trick of Kernel Function**:

$$K(\underline{x}_i, \underline{x}_j) = \underline{x}_i^\mathsf{T} \underline{x}_j \qquad \leftarrow \text{ scalar product between 2 samples}$$

Consider:

$$\underline{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix} \quad \rightarrow \quad \phi(\underline{x}_i) = \begin{bmatrix} x_{i1}^2 \\ \sqrt{2} x_{i1} x_{i2} \\ x_{i2}^2 \end{bmatrix}$$

$$\phi(\underline{x}_i)^\mathsf{T} \phi(\underline{x}_j) = x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{i2} x_{j1} x_{j2} + x_{i2}^2 x_{j2}^2 = (\star)$$

In order to compute $(\star)$ we have to apply the feature map twice and the scalar product between the two vectors. We can avoid this by using the kernel function:

$$K(\underline{x}_i, \underline{x}_j) = (\underline{x}_i^\mathsf{T}\underline{x}_j)^2 = (x_{i1}x_{j1} + x_{i2}x_{j2})^2 = (\star)$$

But here i'm computing this product immediately, without applying the feature map.WE have computed the scalar product in a cheaper way.

Typical kernel functions family:

i Polynomials of degree q: $K(\underline{x}_i, \underline{x}_j) = (\underline{x}_i^\mathsf{T}\underline{x}_j)^q$

ii Polynomials of degree less or equal to q: $K(\underline{x}_i, \underline{x}_j) = (\underline{x}_i^\mathsf{T}\underline{x}_j + 1)^q$

iii Gaussian RBF: $K(\underline{x}_i, \underline{x}_j) = \exp(-\frac{||\underline{x}_i - \underline{x}_j||_2^2}{2\sigma^2})$

Consider again the value of alpha:

$$\underline{\alpha} = (\ \underbrace{\Phi\Phi^\mathsf{T}}_{K \in \mathbb{R}^{n \times n}} + \lambda I)^{-1}\underline{y}$$
$$= (K + \lambda I)^{-1}\underline{y}$$

$\Phi \in \mathbb{R}^{n \times d}$

$K_{ij} = \phi(\underline{x}_i)^\mathsf{T}\phi(\underline{x}_j) = K(\underline{x}_i, \underline{x}_j)$

How to predict a label from a new sample?

$$\begin{aligned}
\hat{y} &= \phi(\underline{x})^\mathsf{T}\hat{\underline{w}}_R \\
&= \hat{\underline{w}}_R^\mathsf{T}\phi(\underline{x}) \\
&= (\Phi^\mathsf{T}\underline{\alpha})\phi(\underline{x}) \\
&= \underline{\alpha}^\mathsf{T}\Phi\phi(\underline{x}) \\
&= \sum_{i=1}^{n} \alpha_i \phi(\underline{x}_i)^\mathsf{T}\phi(\underline{x}) \\
&= \sum_{i=1}^{n} \alpha_i K(\underline{x}_i, \underline{x})
\end{aligned}$$

We are doing a scalar product between the already present samples in the dataset and the new one. All weighted by the value of α which represent how much each sample contribute to the definition of the decision boundary. Recall that the scalar product is a similarity measure so there will be samples more or less similar to the one to predict.

Small recap considering both cases in which we consider both following cases: the features are on the horizontal axes and the vertical one.

$X \in \mathbb{R}^{n \times p}$ centered and $\begin{cases} n & \text{\# of samples} \\ p & \text{\# of features} \end{cases}$

$$C = \frac{1}{n-1} X^\mathsf{T} X \, (spd)$$

can be factorized:

$$C = V \Lambda V^\mathsf{T}$$

The columns of V are the eigenvectors of C and Λ is a diagonal matrix with the eigenvalues of C. The eigenvectors are also called **principal directions**. The **principal components** instead, are obtained as:

$$XV$$

And they represent the coordinates of the original samples in the new basis. If we write $X = U \Sigma V^\mathsf{T}$ then

$$C = \frac{1}{n-1} V \Sigma^\mathsf{T} U^\mathsf{T} U \Sigma V^\mathsf{T}$$
$$= \frac{1}{n-1} V \Sigma^2 V^\mathsf{T}$$

And we have also that:

$$\lambda_i = \frac{1}{n-1} \sigma_i^2$$

Moreover, with SVD:

$$XV = U \Sigma V^\mathsf{T} V = U \Sigma$$

Are still principal components.

$X \in \mathbb{R}^{p \times n}$ centered and $\begin{cases} n & \text{\# of samples} \\ p & \text{\# of features} \end{cases}$

Now, in this case we simply need to interchange the roles for U and V.

$$C = \frac{1}{n-1} X X^\mathsf{T} = \frac{1}{n-1} U \Sigma^2 V^\mathsf{T}$$

The principal directions are in U (and indeed not in V this time). To get the principal components we have to:

$$X^\mathsf{T} U$$

In Ridge we have found that:

$$\hat{\underline{w}}_R = \arg\min_{\underline{w}} ||\underline{y} - \Phi \underline{w}||_2^2 + \lambda ||\underline{w}||_2^2$$

Let's consider a binary classification problem so we have $(\underline{x}_i, y_i)$ and $y_i \in \{-1, 1\}, \underline{x}_i \in \mathbb{R}^p$. Now consider Least squares for this problem:

$$\hat{\underline{w}}_{LS} = \arg\min_{\underline{w}} \sum_{i=1}^n (y_i - \underline{x}_i^\mathsf{T} \underline{w})^2 = \arg\min_{\underline{w}} \sum_{i=1}^n (1 - y_i \underline{x}_i^\mathsf{T} \underline{w})^2$$

You can easily check that the last equality holds in both values of y_i because the parenthesis is squared. There is an example on the notes that shows how the least squares is not a good idea for classification problems. Here are different loss functions we could use:

1. Square loss: $L_i(y_i, \underline{X}_i^\mathsf{T}, \underline{w}) = (1 - y_i \underline{X}_i^\mathsf{T} \underline{w})^2$

2. Ideal loss: $L_i(y_i, \underline{X}_i^\mathsf{T}, \underline{w}) = \begin{cases} 0 & y_i \\ 1 & y_i \underline{X}_i^\mathsf{T} \underline{w} < 0 \end{cases}$

3. Hinge loss: $L_i(y_i \underline{X}_i^\mathsf{T} \underline{w}) = (1 - y_i \underline{X}_i^\mathsf{T} \underline{w})_+$

Where $(a)_+ = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{if } a \le 0 \end{cases}$

Here is the plot of those functions:
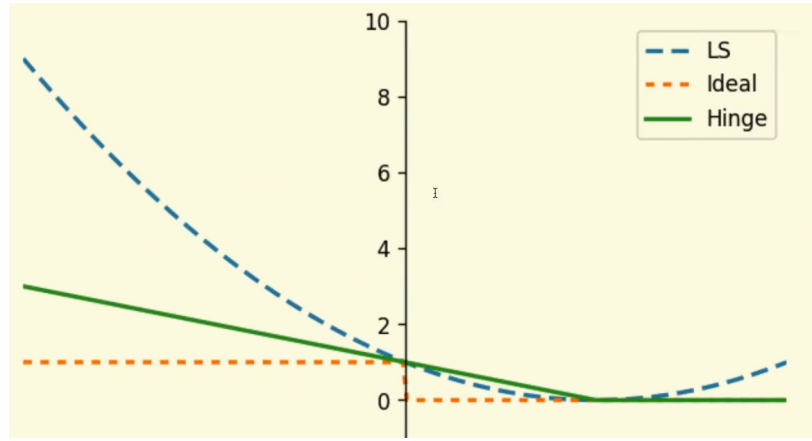
On the x axis you have $y_i \underline{X}_i^{\mathsf{T}} \underline{w}$. This picture can be divided in 3 points: if we are in the negative part (left part of the plot) this means that the predicted label is wrong and it has opposite sign with respect to the true label. In the middle part, the predicted label is correct but the margin is small. In the right part, the predicted label is correct and the margin is large. The square loss, differently from the other two, continue to grow even if the margin is large. The ideal loss is not differentiable and this is a problem for the optimization. The hinge loss is differentiable and it is the most used in practice. The ideal loss gives the same weight in the loss for a misclassified entry independently on our far is from the boundary. In the hinge instead the loss grows linearly with the distance and in the square grows squared.

14 Computing derivatives

Now we are going to consider different ways to compute the derivative of a function:

Method	PRO	CONS
Manual computation	Exact, good for the proofs	Prone to error, expensive for complex functions
Numerical differentiation	Easy to program	Floating point precision*, computational cost
Symbolic differentiation	Exact, good for proofs	Expression swelling**, not able to differentiate if conditions or loops
Automatic differentiation (AD)	Exact, fast	Implementation

*: view jupyter notebook "Round_off&Truncation.ipynb" for more details.
**: big expressions of the derivative.

Consider this function:
$$y = f(x_1, x_2) = \sin((x_1 + x_2)x_2^2)$$

There are two possibilities for representing this function:

1. **Wengert list** for a generic function $f : \mathbb{R}^n \to \mathbb{R}^m$ is defined as:

$$\begin{cases} \text{Input variables: } v_{i-n} = x_i & i = 1, \ldots, n \\ \text{Intermediate variables: } v_i & i = 1, \ldots, l \\ \text{Output variables: } v_{n-i} = v_{l-i} & i = m-1, \ldots, 0 \end{cases}$$

Now, let's go back to out example function, in this case it's Wengert list formulation is the following:

$$v_{-1} = x_1 = 1$$
$$v_0 = x_2 = 2$$
$$\overline{v_1 = x_1 + x_2 = 3}$$
$$v_2 = x_2^2 = 4$$
$$v_3 = v_1 v_2 = 12$$
$$\overline{v_4 = \sin(v_3) = 0.536}$$
$$y = v_4$$

Notice that i've separated with the horizontal lines the different variables of the list. The input variables values have been chosen randomly just to compute the example with numbers.

How can we use the Wengert list for computing the derivative of a function? Let's consider the derivative of y with respect to x_1:

$$\dot{v}_{-1} = \dot{x}_1 = 1$$
$$\dot{v}_0 = \dot{x}_2 = 0$$
$$\overline{\dot{v}_1 = \dot{x}_1 + \dot{x}_2 = 1}$$
$$\dot{v}_2 = 0$$
$$\dot{v}_3 = \dot{v}_1 v_2 + v_1 \dot{v}_2 = 4$$
$$\overline{\dot{v}_4 = \cos(v_3)\dot{v}_3 = 3.37}$$
$$\overline{\dot{y} = \dot{v}_4 = 3.37}$$

In this way we have splitted and computed only easy operations. This is called the **Forward Mode of Automatic Differentiation (AD)**. AD gives you the derivative computed for a certain popint. If we change points the opereations have to be done again.

2. **Direct Acyclic Graph (DAG)**
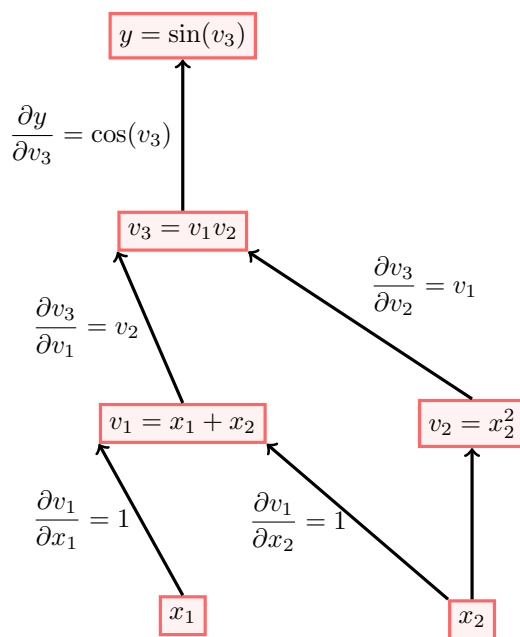
As you can see on the nodes there are the values of the variables and on the arrows there are the derivatives. The arrows are the partial derivatives of the function with respect to the variable of the node. **To compute the derivative of the function wrt a certain variable, you need to consider all paths that connect the top of the graph with the desider variable**. From y to x_1 there is a single path so it's easy but if you consider y and x_2 there are 2 possible paths, in this case you need to add them.

This is the **Reverse or Backward Mode of Automatic Differentiation (AD)**. It is more efficient than the forward mode because it computes the derivative of all the variables at the same time. It is used in neural networks because the number of variables is much higher than the number of samples.

Consider $y = f : \mathbb{R}^n \to \mathbb{R}^m$ and the two cases: $\begin{cases} m \gg n \\ m \ll n \end{cases}$ In particular, the bottom case is the most interesting one to us since we might have many features in our model, especially for images and neural networks. For this case, the forward mode (FM) seen during last lecture is not very suitable because, considering NN, you need to compute a lot of derivatives wrt the weights.

Recall that, the Jacobian matrix is defined as:

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial y_1}{\partial x_1} & \dfrac{\partial y_1}{\partial x_2} & \cdots & \dfrac{\partial y_1}{\partial x_n} \\ \dfrac{\partial y_2}{\partial x_1} & \dfrac{\partial y_2}{\partial x_2} & & \dfrac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial y_m}{\partial x_1} & \dfrac{\partial y_m}{\partial x_2} & \cdots & \dfrac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Suppose to have $\underline{\dot{X}} = \underline{e}_i$ as initialization value for the Wengert list forward pass method. We would have:

$$\dot{y}_j = \frac{\partial y_j}{\partial x_i} \quad j = 1, \ldots, n$$

You obtain a column of the jacobian. In general we are interested in the product of the jacobian times a vector which could be anything but often is the evaluation of the function at the previous iteration. It means that in practical computation, we are not interested in having the jacobian and then to perform the matrix-vector multiplication, but we want the result of it.

$$\mathbf{J}\underline{r}$$

The FM of AD allows to have this result in a matrix free approach. Any algorithm that is "matrix-free" means that in practise, even if the computation involves the presence of a matrix, you are able to avoid the construction of the complete matrix. You can resort to some tricks or algorithms that allow you to come up to the desired result without the need to explicitly computing the matrix. This is really good especially when dealing with big matrices.

Instead of $\underline{\dot{X}} = \underline{e}_i$, we can start from $\underline{\dot{X}} = \underline{r}$ where $\underline{r}$ is the vector we want to multiply the jacobian with.

14.1 Dual-numbers

They are similar to complex numbers.

$$a + b\epsilon \quad \text{with} \quad a, b \in \mathbb{R} \quad \text{and} \quad \epsilon \neq 0, \epsilon^2 = 0$$

Where a is the real part and b is the dual part. Basic operations with dual numbers:

$$(a + b\epsilon) + (c + d\epsilon) = (a + c) + (b + d)\epsilon$$

$$(a + b\epsilon)(c + d\epsilon) = ac + (ad + bc)\epsilon$$

Why this algebra is useful to our purposes? Consider the generic function f, we want to evaluate it with a dual number:

$$f(a + b\epsilon) = f(a) + f'(a)b\epsilon + \underbrace{\frac{f''(a)}{2}b^2\epsilon^2 + \ldots}_{0}$$

We have used the Taylor expansion, the last terms are zero because of ϵ^2. Notice that, if we have unitary dual part ($b = 1$), we obtain the derivative of f in a. This is the key point of the dual numbers:

$$b = 1 \implies a + b\epsilon = a + \epsilon \implies f(a + \epsilon) = f(a) + f'(a)\epsilon$$

The last term is again a dual number in which the real part is the value of the function in a and the dual part is the derivative of the function in a.

Let's check if the derivative properties are actually respected, suppose to have $f(x) = g(x)h(x)$:

$$\begin{aligned} f(a + \epsilon) &= g(a + \epsilon)h(a + \epsilon) \\ &= [g(a) + g'(a)\epsilon][h(a) + h'(a)\epsilon] \\ &= g(a)h(a) + g(a)h'(a)\epsilon + g'(a)h(a)\epsilon + g'(a)h'(a)\epsilon^2 \\ &= g(a)h(a) + \underbrace{[g(a)h'(a) + g'(a)h(a)]}_{\text{der. of the product}}\epsilon \end{aligned}$$

Or $f(x) = g(h(x))$:

$$\begin{aligned}
f(a + \epsilon) &= g(h(a + \epsilon)) \\
&= g(h(a) + h'(a)\epsilon) \\
&= \underbrace{g(h(a))}_{f(a)} + \underbrace{g'(h(a))h'(a)\epsilon}_{f'(a)\epsilon}
\end{aligned}$$

Numerical example:

$$f(x) = \frac{1}{x} \implies f(x + \epsilon) = \frac{1}{x + \epsilon} = \frac{x - \epsilon}{(x + \epsilon)(x - \epsilon)} = \frac{x - \epsilon}{x^2 - \epsilon^2} = \frac{x - \epsilon}{x^2} = \underbrace{\frac{1}{x}}_{f(a)} - \underbrace{\frac{1}{x^2}\epsilon}_{f'(a)\epsilon}$$

Normally, we won't use the definition like this, but we will use real values, i.e. $x = 2$ for example. There is formalism for which ϵ are represented as matrices:

$$\epsilon = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \implies \epsilon^2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

But this is not so important because at the end we are interested in its cohefficients.

Let's now analyze the **Backward or reverse mode**.

It's related with the sensitivity of the output wrt the input. We introduce a new quantity:

$$\bar{v}_i = \frac{\partial y}{\partial v_i}$$

Where v_i is one of the variables we have written considering the forward mode and which might not be the input. This is what happen in backpropagation, a partial derivative of the output wrt to a certain weight of the NN. We start from the output:
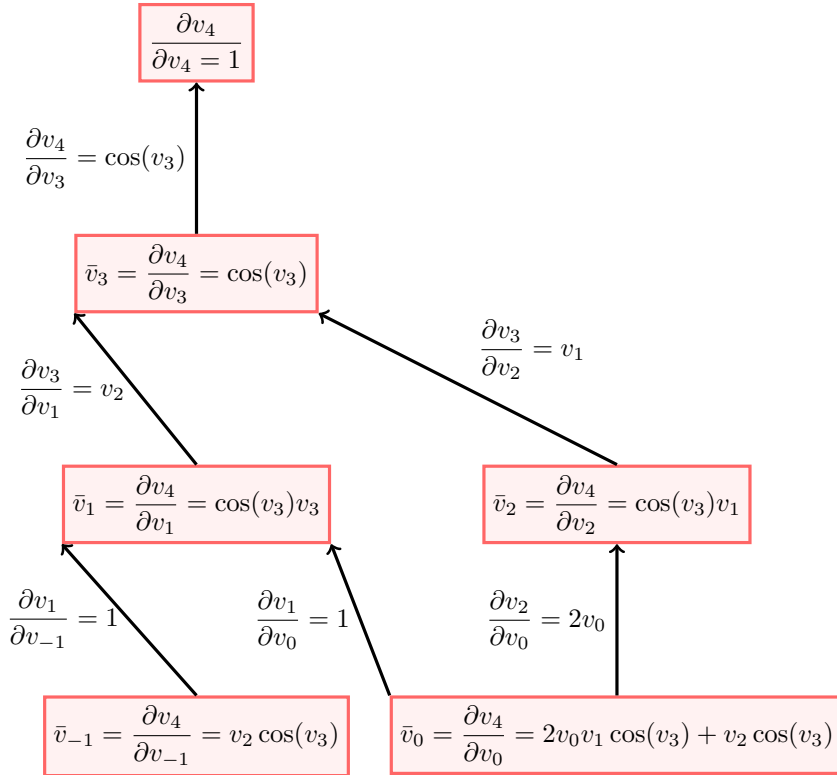
Where the bottom left box is $\frac{\partial y}{\partial x_1}$ and the bottom right box is $\frac{\partial y}{\partial x_2}$ We have been able to compute all the intermediate sensitivities not just wrt of the imput. If we have $\mathbb{R}^n \to \mathbb{R}^m$ and $n \gg m$ this method is very efficient because with just one sweep you compute the derivative of the output wrt all the input parameters. In the forward mode i would have had to perform n forward steps, and obviously this would be more costly. The same reasons but inversly are valid in case of $n \ll m$.

If you have a function $f : \mathbb{R}^n \to \mathbb{R}$, for 1 evaluation of $ops(f)$ (floating point operations), then is possible to find that, the computation of the Jacobian requires:

- FM $\approx n \times c \times ops(f)$ operations

- BM $\approx m \times c \times ops(f)$ operations

Where $c \approx 2.5$. Theoretical results that have been proven. So, obviously, depending on the size of m and n, one of the two methods is more convenient.

If in FM we initialize $\dot{\underline{x}} = \underline{r}$ then we will end up with $\mathbf{J}\underline{r}$. While in BM if we initialize with $\bar{y} = \underline{r}$ then we will obtain $\mathbf{J}^{\intercal}\underline{r}$. In both cases we have a matrix-free method that allows to come up with this problem. What is one drawback of the backward mode BM? In order to compute the backward you need to store all the values of the forward step.

In the **Newton method** we need to compute the **Hessian matrix H**. As before, in numerical methods, we won't be interested in having the Hessian matrix by its own but always it's important to compute the product of the Hessian times a vector $\underline{v}$. So we have the same situation we had for the jacobian.

Suppose to have $f : \mathbb{R}^n \to \mathbb{R}$ and the vector $\underline{x}$ that we want to multiply to the Hessian. There is the method called: **Reverse-on-forward** in which essentially you need to perform two steps:

i $\nabla f \times \underline{x} \equiv \dfrac{\partial f}{\partial \underline{x}}$ and this can be done in a forward sweep. In this particular function case this operation is the same as $\mathbf{J}\underline{r}$ and correspond to the directional derivative of f in the direction of $\underline{x}$. This can be computed with the forward mode by initializing $\dot{\underline{x}} = \underline{v}$.

ii Apply the backward mode to the result of i and obtain $\nabla^2 f \times \underline{v} = H\underline{v}$.

To compute in general the Jacobian requires n^2 operations while in this case just n. we don't have the complete matrix **H** and this is obviously the tradeoff but in this case we are interested in the product of the Hessian times a vector, so depending on what you are interested in, you can use this efficient method.

15 Convolution

In this lecture we are going to deal with the topic of convolution. The general definition is given by:

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(t)g(x - t)dt$$

The operation can be applied also to vector with:

$$(\underline{c} * \underline{d})_K = \sum_{i+j=k} c_i d_j = \sum_i c_i d_{k-i}$$

The pedix K is used to indicate the k-th element of the vector to which is the convolution is applied. The vectors can be expressed also as polynomials:

$$c(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1}$$

$$d(x) = d_0 + d_1 x + d_2 x^2 + \cdots + d_{n-1} x^{n-1}$$

The convolution is the product of these polynomials. [to finish]

15.1 Cyclic convolution

The cyclic convolution is a particular case of convolution in which the vectors are cyclic. This means that the last element of the vector is followed by the first one. The cyclic convolution is defined as:

$$(\underline{c} \circledast \underline{d})_K = \sum_{i+j=k \mod (n)} c_i d_j$$

How can this be written in matrix form?

<div style="display:flex">

CONVOLUTION

In this case are used the **Toeplitz matrices** (or also called Time-Invariant Linear Systems), the elements are given by $(2n - 1)$-length sequence:

$$\{t_K : -(n - 1) \leq K \leq (n - 1)\}$$

The element in position (i, j) is given by $T(i, j) = t_i - t_j$ and the generic Toeplitz matrix as this shape:

$$T = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \cdots & t_{-(n-1)} \\ t_1 & t_0 & t_{-1} & \cdots & t_{-(n-2)} \\ t_2 & t_1 & t_0 & \cdots & t_{-(n-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{n-1} & t_{n-2} & t_{n-3} & \cdots & t_0 \end{bmatrix}$$

So, it's easy to notice that on the diagonal there is always a constant vector.

CYCLIC CONVOLUTION

In this case are used the **Circulant matrices**, a particular case of a Toeplitz matrix. For a given $n \times n$ matrix, the elements are given by (n)-length sequence:

$$\{c_K : 0 \leq K \leq (n - 1)\}$$

The element in position (i, j) is given by $C(i, j) = c_{i-j \mod (n)}$ and the generic Circulant matrix as this shape:

$$C = \begin{bmatrix} c_0 & c_{n-1} & c_{n-2} & \cdots & c_1 \\ c_1 & c_0 & c_{n-1} & \cdots & c_2 \\ c_2 & c_1 & c_0 & \cdots & c_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n-1} & c_{n-2} & c_{n-3} & \cdots & c_0 \end{bmatrix}$$

As you can see, the last element of a column vector becomes the first element of the next column vector, for this reason is called circulant.

</div>

Example of circulant matrix:

$$C = \begin{bmatrix} 1 & 8 & 5 & 3 \\ 3 & 1 & 8 & 5 \\ 5 & 3 & 1 & 8 \\ 8 & 5 & 3 & 1 \end{bmatrix}$$

Now we introduce a **permutation matrix** P defined as follows:

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

If we multiply the permutation matrix with a vector, this happen:

$$P\underline{c} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_0 \end{bmatrix}$$

All elements are shifted of 1 position. This means that the circulant matrix C can be built as follows:

$$C = c_0 I + c_1 P + c_2 P^2 + c_3 P^3$$

This is true for any circulant matrix so we define also D:

$$D = d_0 I + d_1 P + d_2 P^2 + d_3 P^3$$

What happen when we multiply CD, i.e two circulant matrices?

$$CD = (c_0 I + c_1 P + c_2 P^2 + c_3 P^3)(d_0 I + d_1 P + d_2 P^2 + d_3 P^3)$$

But this means that we end up with elements with P^4 and P^5 and so on, but $P^4 = I$, $P^5 = P$ and $P^6 = P^2$. This makes sense even considering that we are dealing with circulant matrices. In general we can say:

$$P_n \text{ of } n \times n \implies P^n = I$$

Example:
We want to multiply $(1,2,1)(3,1,2)$. We transform the two vectors in two polynomials:

$$(1 + 2x + x^2)(3 + x + 2x^2) = 3 + 7x + 7x^2 + 5x^3 + 2x^4$$

The terms at the third and forth power must be shifted.

$$(3 + 5) + (7 + 2)x + 7x^2 = 8 + 9x + 7x^2$$

The final result is (8,9,7) and, to check its correctness we can consider this sort of property:

$$\{(1, 2, 1) \implies 1 + 2 + 1 = 4 \times 6 = 3 + 1 + 2 \impliedby (3, 1, 2)\} \implies 6 \times 4 = 8 + 9 + 7 \impliedby (8, 9, 7)$$

If we now rename the vectors $\underline{c} = (1, 2, 1)$ and $\underline{d} = (3, 1, 2)$, we can compute their convolution by:

$$\underline{c} * \underline{d} = C\underline{d} = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 + 1 + 4 \\ 6 + 1 + 2 \\ 3 + 2 + 2 \end{bmatrix} = \begin{bmatrix} 8 \\ 9 \\ 7 \end{bmatrix}$$

We have built the circulant matrix starting from the vector $\underline{c}$ and then we have multiplied it with the vector $\underline{d}$. This is the same as multiplying the two polynomials. I did not understand wheter you can either build C from using the initial vector as its row or column.

The matrix CD is circulant because it's the product of two circulant matrices.

15.2   Eigenvectors and eigenvalues of a circuland matrix

Let's start considering again the permutation matrix P. We can compute its eigenvalues by using the definition method:

$$P - \lambda I = \begin{bmatrix} -\lambda & 1 & 0 & 0 \\ 0 & -\lambda & 1 & 0 \\ 0 & 0 & -\lambda & 1 \\ 1 & 0 & 0 & -\lambda \end{bmatrix} \implies \det(P - \lambda I) = \lambda^4 - 1 = 0$$

So the eigenvalues of P are the fourth roots of 1, which correspond to:

$$\lambda^4 - 1 = 0$$
$$\lambda_1 = 1$$
$$\lambda_2 = i$$
$$\lambda_3 = -1$$
$$\lambda_4 = -i$$
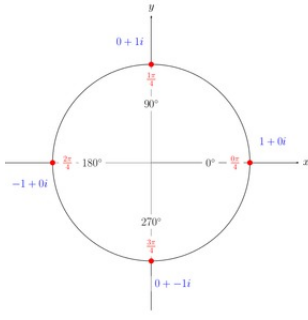
We introduce now the complex number w given by:

$$w = e^{\frac{2\pi i}{n}} \xrightarrow{\text{in this case}} w = e^{\frac{2\pi i}{4}} \implies \begin{cases} \lambda_1 = w^0 \\ \lambda_2 = w^1 \\ \lambda_3 = w^2 \\ \lambda_4 = w^3 \end{cases}$$

In general we have:

$$P_n \implies \lambda^n - 1 = 0 \implies w^0, w^1, \ldots, w^{n-1}$$

Another property of P: it's orthogonal indeed $P^\intercal P = I$. What about the eigenvectors of P? Consider the generic matrix C written in terms of P:

$$C = c_0 I + c_1 P + c_2 P^2 + c_3 P^3$$

We want to find the eigenvectors of C. If $\lambda_k, \underline{v}_k$ is the couple of eigenvalue and eigenvector of C, then:

$$C\underline{v}_k = \lambda_k \underline{v}_k$$
$$(c_0 I + c_1 P + c_2 P^2 + c_3 P^3)\underline{v}_k = (c_0 + c_1 \lambda_k + c_2 \lambda_k^2 + c_3 \lambda_k^3)\underline{v}_k$$

$\underline{v}_k$ is an eigenvector of P.

Example:

$$\lambda_1 = 1 \qquad \underline{v}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\lambda_2 = i \qquad \underline{v}_2 = \begin{bmatrix} 1 \\ i \\ i^2 \\ i^3 \end{bmatrix}$$

$$\lambda_3 = -1 \qquad \underline{v}_3 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

$$\lambda_4 = -i \qquad \underline{v}_4 = \begin{bmatrix} 1 \\ (-i) \\ (-i)^2 \\ (-i)^3 \end{bmatrix}$$

Example of computation of the third eigenvector:

$$\underline{v}_3 = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}}_{\begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}} = -1 \underbrace{\begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}}_{\begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}}$$

We define

$$\mathbf{F} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 \\ 1 & w^2 & w^4 & w^6 \\ 1 & w^3 & w^6 & w^9 \end{bmatrix}}_{\text{eigenvectors}} \underbrace{\frac{1}{\sqrt{n}}}_{\text{normalization}} \qquad \text{where} \quad w = e^{\frac{2\pi i}{4}}$$

43

If you multiply a vector for $\mathbf{F}$ you get its **Discrete Fourier Transform (DFT)**.

$$\underline{\lambda}_c = \begin{bmatrix} \lambda_0(c) \\ \vdots \\ \lambda_{K-1}(c) \end{bmatrix} = \begin{bmatrix} c_0 + c_1 + \cdots + c_{n-1} \\ \vdots \\ c_0 + c_1 w^{n-1} + \cdots + c_{n-1} w^{(n-1)(n-1)} \end{bmatrix} = \mathbf{F} \begin{bmatrix} \underline{c}_0 \\ \vdots \\ \underline{c}_{n-1} \end{bmatrix} = \mathbf{F}\underline{c}$$

Where $\underline{\lambda}_c$ is the vector containing the eigenvalues of the circulant matrix C and $\underline{c}$ is the vector containing the coefficients of the polynomial $c(x)$.

Consider two matrices A and B, with some eigenvalues and eigenvectors:

$$A\underline{v} = \lambda\underline{v} \qquad B\underline{v} = \gamma\underline{v}$$

then

$$AB\underline{v} = \gamma A\underline{v} = \gamma\lambda\underline{v}$$

So the eigenvalues of the product is the product of the eigenvalues. Now we are going to exploit this property: consider two circulant matrices C and D built from the vectors $\underline{c}$ and $\underline{d}$:

- CD in the first row you have the cyclic convolution of $\underline{c}$ and $\underline{d}$ ($\underline{c} \circledast \underline{d}$)

- $\mathbf{F}(\underline{c} \circledast \underline{d})$ is the vector of eigenvalues of CD

Eigenvalues of C: $\lambda(c) = \mathbf{F}\underline{c}$ and eigenvalues of D: $\lambda(d) = \mathbf{F}\underline{d}$
So the eigenvalues of CD are:

$$\lambda(CD) = \lambda(C) \cdot^* \lambda(D) = \mathbf{F}\underline{c} \cdot^* \mathbf{F}\underline{d} = \underline{\lambda}_{CD}$$

Two ways of computing the eigenvalues:

$$\mathbf{F}(\underline{C} \circledast \underline{d}) = \mathbf{F}\underline{c} \cdot^* \mathbf{F}\underline{d}$$

This is called **Convolution rule**. Which is the fastest method? The FFT (Fast Fourier Transform) takes $n\log(n)$ operations for computing the matrix multiplication.

- $\mathbf{F}(\underline{C} \circledast \underline{d}) = n^2 n\log(n)$

- $\mathbf{F}\underline{c} \cdot^* \mathbf{F}\underline{d} = 2n\log(n) + n$ so it's better this one!

So far we have seen that our goal is to find:

$$\underline{w}^* = \arg\min_{\underline{w}} J(\underline{w})$$

Let's now see gradually, how to solve this minimization problem by means of different methods.

16   Gradient descent (GD)

Consider the following first order approximation:

$$\Delta J \approx \nabla J \underbrace{\Delta \underline{w}}_{\Delta \underline{w} = -\eta \nabla J} = -\eta \|\nabla J\|^2$$

The actual method is defined by the formula:

$$\underline{w}^{(k+1)} = w^{(k)} - \eta \nabla J(\underline{w}^{(k)})$$

Let's see some properties using now x as w, so the formula is: $\underline{x}^{(k+1)} = x^{(k)} - \eta \nabla f(\underline{x}^{(k)})$ where f is the function we want to minimize and η the learning rate.

If η is to small, the convergence is slow. If η is too large, the convergence is not guaranteed since you have overshooting. For now, we are considering a fixed η (not dependent on k).

16.1   Convergence

Definition: $f : \operatorname{dom}(f) \to \mathbb{R}$ is **convex** if:

  i $\operatorname{dom}(f)$ is convex

 ii $\forall \underline{x}, \underline{y} \in \operatorname{dom}(f), \lambda \in [0, 1] : f(\lambda \underline{x} + (1 - \lambda)\underline{y}) \leq \lambda f(\underline{x}) + (1 - \lambda)f(\underline{y})$

Lemma: if $\operatorname{dom}(f)$ is open and f is differentiable, then f is complex iff the domain of f is convex and

$$f(\underline{y}) \geq f(\underline{x}) + \nabla f(\underline{x})^T(\underline{y} - \underline{x}) \quad \forall \underline{x}, \underline{y} \in \operatorname{dom}(f)$$

This last formulation states that the function must be above the tangent line or, more generally, above the tangent hyperplane built starting from point x.

Set $\underline{x} = \underline{x}^{(k)}$ and $\underline{y} = \underline{x}^*$, where $\underline{x}^*$ is the minimum point. Then we can substitute the inequality in the lemma and obtain:

$$f(\underline{x}^*) \geq f(\underline{x}^{(k)}) + \nabla f(\underline{x}^{(k)})^T(\underline{x}^* - \underline{x}^{(k)})$$
$$f(\underline{x}^{(k)}) - f(\underline{x}^*) \leq \nabla f(\underline{x}^{(k)})^T(\underline{x}^* - \underline{x}^{(k)})$$

We define now the value $\underline{g}^{(k)} = \nabla f(x^{(k)})$. Thanks to this, we can rewrite the original gradient descent formula as:

$$\underline{x}^{(k+1)} - \underline{x}^{(k)} = -\eta \underline{g}^{(k)}$$

And so we can reformulate as follows:

$$\underline{g}^{(k)} = \frac{\underline{x}^{(k+1)} - \underline{x}^{(k)}}{-\eta} = \frac{\underline{x}^{(k)} - \underline{x}^{(k+1)}}{\eta}$$

Thanks to this we have:

$$f(\underline{x}^{(k)}) - f(\underline{x}^*) \leq \nabla f(\underline{x}^{(k)})^T(\underline{x}^* - \underline{x}^{(k)}) = \frac{1}{\eta}(\underline{x}^{(k)} - \underline{x}^*)(\underline{x}^{(k)} - \underline{x}^{(k+1)}) = (\star)$$

We are now going to exploit a property seen also when dealing with least squares, for which:

$$\|\underline{v} - \underline{w}\|^2 = (\underline{v} - \underline{w})^\mathsf{T}(\underline{v} - \underline{w}) = \underline{v}^\mathsf{T}\underline{v} - 2\underline{v}^\mathsf{T}\underline{w} + \underline{w}^\mathsf{T}\underline{w} = \|\underline{v}\|^2 - 2\underline{v}^\mathsf{T}\underline{w} + \|\underline{w}\|^2$$

$$\implies 2\underline{v}^\mathsf{T}\underline{w} = \|\underline{v}\|^2 + \|\underline{w}\|^2 - \|\underline{v} - \underline{w}\|^2$$

We reconduct to this form by adding a 2 both at the numerator and denumerator and considering the two parenthesis as the vectors v and w:

$$(\star) = \frac{1}{2\eta} 2 \underbrace{(\underline{x}^{(k)} - \underline{x}^*)}_{v^\intercal} \underbrace{(\underline{x}^{(k)} - \underline{x}^{(k+1)})}_{w}$$

$$= \frac{1}{2\eta} \left(\|\underline{x}^{(k)} - \underline{x}^{(k+1)}\|^2 + \|\underline{x}^{(k)} - \underline{x}^*\|^2 - \|\underline{x}^{(k+1)} - \underline{x}^*\|^2 \right)$$

$$= \frac{1}{2\eta} \left(\eta^2 \|\underline{g}^{(k)}\|^2 + \|\underline{x}^{(k)} - \underline{x}^*\|^2 - \|\underline{x}^{(k+1)} - \underline{x}^*\|^2 \right)$$

$$= \frac{\eta}{2} \|g^{(k)}\| + \frac{1}{2\eta} \left(\|\underline{x}^{(k)} - \underline{x}^*\|^2 - \|\underline{x}^{(k+1)} - \underline{x}^*\|^2 \right)$$

In the second row, in the last norm, you should have $x^{(k)}$ also (notice the property) but it's not there because it's summed to it's opposite so they cancel out.

If N is the last iteration we want to perform we can write:

$$\sum_{k=0}^{N-1} \left(f(\underline{x}^{(k)}) - f(\underline{x}^*) \right) \leq \frac{\eta}{2} \sum_{k=0}^{N-1} \|\underline{g}^{(k)}\|^2 + \frac{1}{2\eta} \|\underline{x}^{(0)} - \underline{x}^*\|^2 = (\divideontimes)$$

There are just two terms of $\underline{x}$ because it is a telescopic sum and the intermediate contributions cancel out (see previous equations in which both termns $x^{(k)}$ and $x^{(k+1)}$ are present). To be even more precise, there should be also another term $x^{(N)}$ (its norm with x star)but it is not there because it is summed with the negative sign so, the inequality holds even without it.

Recall that $x^{(0)}$ is the initial guess, given. The problem is the norm of the gradient, which is not known.

16.2 Lipschitz convex functions

We are assuming that all the gradients are bounded ($\|g^{(k)} \leq L\|, \forall k$) which implies that the function is Lipschitz continuous. We want to prove this theorem:

Theorem: $f : \mathbb{R}^d \to \mathbb{R}$ convex, differentiable with minimum $\underline{x}^*$, suppose that $\|x^{(0)} - x^*\| < R$ and $\nabla f(x) \leq L$. Then chosing $\eta = \frac{R}{L\sqrt{N}}$ we have

$$\frac{1}{N} \sum_{k=0}^{N-1} \left(f(\underline{x}^{(k)}) - f(\underline{x}^*) \right) \leq \frac{RL}{\sqrt{N}}$$

For example the function $f(x) = x^2$ does not satisfy the hypothesis of this theorem because the gradient is not bounded. This hypothesis $\|x^{(0)} - x^*\| < R$ states that the initial guess is not too far from the optimum. Let's now consider the quantity $(\divideontimes)$, it can be rewritten, considering the hypothesis of the theorem, as:

$$(\divideontimes) = \underbrace{\frac{\eta}{2} NL^2 + \frac{1}{2\eta} R^2}_{l(\eta)} \implies l'(\eta) = \frac{NL^2}{2} - \frac{R^2}{2\eta^2}$$

We set the derivative to zero to find the minimum of the function $l(\eta)$:

$$NL^2 - \frac{R^2}{\eta^2} = 0 \implies \eta = \frac{R}{L\sqrt{N}}$$

Which corresponds to the value we have stated in the hypothesis of the theorem. We now substitute the value of η in $l(\eta)$:

$$l(\eta) = \frac{\eta}{2} NL^2 + \frac{1}{2\eta} R^2 = \frac{R}{2L\sqrt{N}} NL^2 + \frac{1}{2} \frac{R^2 L\sqrt{N}}{R} = RL\sqrt{N}$$

So, we recap what we have done so far:

$$\sum_{k=0}^{N-1} \left(f(\underline{x}^{(k)}) - f(\underline{x}^*) \right) \leq \frac{\eta}{2} \sum_{k=0}^{N-1} \|\underline{g}^{(k)}\|^2 + \frac{1}{2\eta} \|\underline{x}^{(0)} - \underline{x}^*\|^2 = \frac{\eta}{2} NL^2 + \frac{1}{2\eta} R^2$$

$$\leq \frac{\eta}{2} NL^2 + \frac{1}{2\eta} R^2$$

$$\leq RL\sqrt{N}$$

We can divide both terms for N and obtain the result:

$$\frac{1}{N} \sum_{k=0}^{N-1} \left(f(\underline{x}^{(k)}) - f(\underline{x}^*) \right) \leq \frac{RL}{\sqrt{N}}$$

The left part is a sort of average of the error, we can bound it by working on the right part.

$$\frac{RL}{\sqrt{N}} \leq \epsilon \implies \frac{R^2 L^2}{N} \leq \epsilon^2 \implies N \geq \frac{R^2 L^2}{\epsilon^2}$$

And so $N = O(\frac{1}{\epsilon^2})$. This is the number of iterations of the gradient descent method that we need in order to achieve an average error ϵ. R, L are not quantity easily computable but R in general, in real scenarios, could be big (look how it is defined in the theorem). L also can be big. An **important** thing to notice regarding the last result is that we have found a bound which does not depend on the dimension of the feature vector d (indeed we defined $f : \mathbb{R}^d \to \mathbb{R}$).

16.3   Smooth convex functions

Definition: $f : \text{dom}(f) \to \mathbb{R}$ differentiable, $X \subseteq \text{dom}(f)$ is convex, $L \in \mathbb{R}^+$ (is a positive constant) function f is smooth with parameter L over X if:

$$\forall \underline{x}, \underline{y} \in X : f(\underline{y}) \leq f(\underline{x}) + \nabla f(\underline{x})^T (\underline{y} - \underline{x}) + \frac{L}{2} \|\underline{y} - \underline{x}\|^2$$

This means that the function is below a quadratic approximation.

Lemma: $\text{dom}(f)$ is open and convex and $f : \text{dom}(f) \to \mathbb{R}$ differentiable. Let $L \in \mathbb{R}^+$ the following statements are equivalent:

1. f is smooth with paramenter L over $\text{dom}(f)$

2. g defined by $\underline{g}(\underline{x}) = \frac{L}{2} \underline{x}^\mathsf{T} \underline{x} - f(x)$ is convex over $\text{dom}(\underline{g}) = \text{dom}(f)$

Considering the definition given above we have these cases:

- if $L = 0$ we can rewrite the inequality as:

$$f(\underline{y}) = f(\underline{x}) + \nabla f(\underline{x})^T (\underline{y} - \underline{x})$$

  Which is the definition of affine function.

- if $f(x) = x^2$, i.e. a function which does not satisfy the hypothesis of the theorem:

$$f(y) = y^2 = x^2 + 2x(y - x) + (y - x)^2$$

  Where there should be a $\frac{L}{2}$ in front of the last term but it is not there because it is 1 which implies that $L = 2$. So, the function is convex and smooth with parameter $L = 2$.

- if $f(\underline{x}) = \underline{x}^\mathsf{T} Q \underline{x} + \underline{b}^\mathsf{T} \underline{x} + c$, typical exmaple of quadratic function. Q is a $d \times d$ matrix , symmetric, $\underline{b} \in \mathbb{R}^d$ and $c \in \mathbb{R}$. The function is smooth with $L = 2\|Q\|$ (twice the spectral norm). Where:

$$\|Q\| = \max_{\substack{\underline{x} \in \mathbb{R}^d \\ \underline{x} \neq 0}} \frac{\|A\underline{x}\|}{\|\underline{x}\|}$$

- if $f(x) = x^4$ then f is not smooth but is convex. **You cannot expect functions that have an asymptotic behaviour higher the quadratic one which are smooth. The highest degree for which you can expect to have smoothness is 2. This does not imply that the functions with degree lower than 2 are smooth.** Consider for example $f(x) = |x|^{3/2}$.

Lemma: $f : \mathbb{R}^d \to \mathbb{R}$ convex, differentiable. The following statements are equivalent:

1. f is smooth with parameter L over $\text{dom}(f)$

2. $\forall \underline{x}, \underline{y} \in \mathbb{R}^d : \|\nabla f(\underline{x}) - \nabla f(\underline{y})\| \leq L\|\underline{x} - \underline{y}\|$

So if the gradient is Lipschitz continuous, then the function is smooth and viceversa.

Let $f : \mathbb{R}^d \to \mathbb{R}$ smooth L, differentiable. Picking $\eta = \frac{1}{L}$, Gradient descent satisfies:

$$f(\underline{x}^{(k+1)}) \le f(\underline{x}^{(k)}) - \frac{1}{2L}\|\nabla f(\underline{x}^{(k)})\|^2 \quad \forall k \ge 0q$$

Proof: we start writing the gradient descent with the η value defined:

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} - \frac{1}{L}\nabla f(\underline{x}^{(k)})$$

We use now the inequality obatined in the smooth convex function definition:

$$
\begin{aligned}
f(\underline{x}^{(k+1)}) &\le f(\underline{x}^{(k)}) + \nabla f(\underline{x}^{(k)})^T(\underline{x}^{(k+1)} - \underline{x}^{(k)}) + \frac{L}{2}\|\underline{x}^{(k+1)} - \underline{x}^{(k)}\|^2 \\
&= f(\underline{x}^{(k)}) + \nabla f(\underline{x}^{(k)})^T\left(\underline{x}^{(k)} - \frac{1}{L}\nabla f(\underline{x}^{(k)}) - \underline{x}^{(k)}\right) + \frac{L}{2}\|\underline{x}^{(k)} - \frac{1}{L}\nabla f(\underline{x}^{(k)}) - \underline{x}^{(k)}\|^2 \\
&= f(\underline{x}^{(k)}) - \frac{1}{L}\|\nabla f(x^{(k)})\|^2 + \frac{1}{2L}\|\nabla f(\underline{x}^k)\|^2 \\
&= f(\underline{x}^{(k)}) - \frac{1}{2L}\|\nabla f(\underline{x}^k)\|^2
\end{aligned}
$$

At each iteration we are reducing the value of f because we are subtracting a positive value everytime. So, by choosing the correct value of the learning rate we have this guarantee. These are called sufficient decrease conditions.

So we have

- smoothness L

- $\eta = \frac{1}{L}$

Then, you can prove that:

$$f(\underline{x}^{(N)}) - f(x^*) \le \frac{L}{2N}\|\underline{x}^{(0)} - \underline{x}^*\|^2 \quad N > 0$$

Proof:

$$\frac{1}{2L}\sum_{k=0}^{N-1}\|\nabla f(\underline{x}^{(k)})\|^2 \le \sum_{k=0}^{N-1}\left(f(\underline{x}^k) - f(\underline{x}^{k+1})\right) = f(\underline{x}^{(0)}) - f(\underline{x}^{(N)})$$

If we assume to have $\|\underline{x}^{(0)} - \underline{x}^*\| \le R$ then

$$\frac{LR^2}{2N} < \epsilon \implies N > \frac{LR^2}{2\epsilon} \implies N = O\left(\frac{1}{\epsilon}\right)$$

This means that the gradient descent applied to a smooth function, with learning rate equal to $1/L$, then we can expect to achieve a tolerance of the order of ϵ in a number of iteration in the order of $1/\epsilon$ and not $1/\epsilon^2$ as in the case of Lipschitz continuous functions. This is way better even if apparantly the starting condition is weaker. This smooth function case is the practical one.

16.4 Convergence results for Gradient Descent

- Lipschitz-convex functions: $O\left(\frac{1}{\epsilon^2}\right)$

- Smooth functions: $O\left(\frac{1}{\epsilon}\right)$

- Smooth and strongy convex functions: $O\left(\frac{1}{\log(\epsilon)}\right)$

- Accelerated gradient descent: $O\left(\frac{1}{\sqrt{\epsilon}}\right)$

16.5 Accelerated Gradient Descent

Aim: minimizing convex function $f : \mathbb{R}^d \to \mathbb{R}$ (with gradient ∇f). "First order method" because it only uses the function and its gradient. What is the best first order method?

Nemirovski and Yudin (1979): every first order method needs in the worst case $O\left(\frac{1}{\sqrt{\epsilon}}\right)$ steps.

Nesterov (1983): accelerated gradient descent (AGD). Let $f : \mathbb{R}^d \to \mathbb{R}$ convex, differentiable and smooth with parameter L. ADG reads:

- choose $\underline{z}^{(0)} = \underline{y}^{(0)} = \underline{x}^{(0)}$

- for $k \geq 0$ set:

$$- \quad \underline{y}^{(k+1)} = \underline{x}^{(k)} - \frac{1}{L}\nabla f(\underline{x}^{(k)}) \qquad \text{Normal step}$$

$$- \quad \underline{z}^{(k+1)} = \underline{z}^{(k)} - \frac{k+1}{2L}\nabla f(\underline{x}^{(k)}) \qquad \text{Aggressive step}$$

$$- \quad \underline{x}^{(k+1)} = \frac{k+1}{k+3}\underline{y}^{(k+1)} + \frac{2}{k+3}\underline{z}^{(k+1)} \qquad \text{Average}$$

Theorem of convergence of AGD: lef $f : \mathbb{R}^d \to \mathbb{R}$ convex, differentiable with a global minimum $\underline{x}^*$ and smooth with parameter L. AGD yields:

$$f(\underline{y}^{(N)}) - f(\underline{x}^*) \leq \frac{2L\|\underline{z}^{(0)} - \underline{x}^*\|^2}{N(N+1)} \qquad N > 0$$

Definition of smooth and strongly convex functions: Let $f : \mathrm{dom}(f) \to \mathbb{R}$ be a convex and differentiable function, $X \subseteq \mathrm{dom}(f)$ convex and $\mu \in \mathbb{R}^+$. Function f is called strongly convex with parameter μ over X if:

$$f(\underline{y}) \geq f(\underline{x}) + \nabla f(\underline{x})^T(\underline{y} - \underline{x}) + \frac{\mu}{2}\|\underline{y} - \underline{x}\|^2 \qquad \forall \underline{x}, \underline{y} \in X$$

Remark

- **smoothness**: $\forall \underline{x} \in X$ the graph of f is below a not-too-steep tangent paraboloid.

- **Strongly-convex**: $\forall \underline{x} \in X$ the graph of f is above a not-too-flat tangent paraboloid.

Theorem of convergence: strongly convex case: let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex, differentiable. Suppose that f is smooth with parameter L and strongly convex with parameter $\mu > 0$. Choosing:

$$\nu = \frac{1}{L}$$

Gradient Descent with arbitrary initial point $\underline{x}^{(0)}$ satisfies:

1. Squared distances to $\underline{x}^*$ are geometrically decreasing: $\|\underline{x}^{(k+1)} - \underline{x}^*\|^2 \leq \left(1 - \frac{\mu}{L}\right)\|\underline{x}^{(k)} - \underline{x}^*\|$ $\qquad k \geq 0$

2. Absolute error after N iterations is exponentially small in N: $f(\underline{x}^{(N)}) - f(\underline{x}^*) \leq \frac{L}{2}\left(1 - \frac{\mu}{L}\right)^N \|\underline{x}^{(0)} - \underline{x}^*\|^2$ $N > 0$

Remember: recalling that $\ln(1+x) \leq x$ we have:

$$N \geq \frac{L}{\mu}\ln\left(\frac{R^2 L}{2\epsilon}\right)$$

17 How a neural network works

Let's define some quantities:

- w_{jk}^l = weight from k-th neuron in layer $(l-1)$-th to j-th neuron in layer l-th

- b_j^l = bias of j-th neuron in layer l-th

- a_j^l = activation of j-th neuron in layer l-th

In particular:

$$a_j^l = \sigma \left(\sum_{k=1}^{n_{l-1}} w_{jk}^l a_k^{l-1} + b_j^l \right)$$

Or in matrix form:

$$\underline{a}^l = \underbrace{\sigma(W^l \underline{a}^{l-1} + \underline{b}^l)}_{\underline{z}^l}$$

where $\underline{W}^l$ is the matrix of weights (the elements are given by $W^l(j,k) = w_{jk}^l$), $\underline{b}^l$ is the vector of biases and $\underline{z}^l$ is the vector of activations.
The sum in the first equation above refers to the neurons in the previous layer (so $l-1$).

17.1 Cost function J

The aim is to compute $\frac{\partial J}{\partial w}$ and $\frac{\partial J}{\partial b}$ with backpropagation for each w and b in the network.

1. The cost function can be written as an average over the cost functions for each sample:

$$J = \frac{1}{N} \sum_n J_n$$

For example $J_n = \frac{1}{2}\|y_n - \bar{y}_n\|^2, \bar{y}_n = a_n^L$.

2. The function can be written in terms of the outputs of the neural network:

$$J = J(\underline{a}^L)$$

17.2 Fundamental relations

We define a new quantity:

$$\delta_j^l = \frac{\partial J}{\partial z_j^l}$$

The term δ_j^l is the error in the j-th neuron in the l-th layer so with the notation $\underline{\delta}^l$ we mean the vector of errors in the l-th layer.

Now we can write the fundamental relations:

1. Error in output layer δ^L

$$\delta_j^L = \frac{\partial J}{\partial a_j^L} \sigma'(z_j^L) \qquad \underline{\delta}^L = \nabla_a J \odot \sigma'(\underline{z}^L)$$

Check footnote about the mathematical symbol $\odot$[1]. Here is the proof:

$$\delta_j^L = \frac{\partial J}{\partial z_j^L} = \sum_k \frac{\partial J}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L}$$

The sum is over all the neurons in the output layer (layer L). a_k^L of the k-th neuron depends only on z_j^L for the j-th neuron when $k = j$, so:

$$\frac{\partial a_k^L}{\partial z_j^L} = 0 \text{ for } k \neq j \implies \delta_j^L = \frac{\partial J}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial J}{\partial a_j^L} \sigma'(z_j^L) \quad (a_j^L = \sigma(z_j^L))$$

---

[1] The symbol $\odot$ means element-wise multiplication (Hadamard product) and from now on we will be using it instead of $\cdot^*$ that was used in the previous notes.

2. Error $\underline{\delta}^l$ in terms of error in next layer $\underline{\delta}^{l+1}$

$$\underline{\delta}^l = \left[(W^{l+1})^T \underline{\delta}^{l+1}\right] \odot \sigma'(\underline{z}^l)$$

Proof:

$$\delta_j^l = \frac{\partial J}{\partial z_j^l} = \sum_k \frac{\partial J}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_k^l}$$

But

$$z_k^{l+1} = \sum_j w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1} \sigma(z_j^l) + b_k^{l+1}$$

$$\Downarrow$$

$$\frac{\partial z_k^{l+1}}{\partial z_k^l} = w_{kj}^{l+1} \sigma'(z_j^l) \implies \delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(z_j^l)$$

3. Variation of cost function J with respect to bias

$$\frac{\partial J}{\partial b_j^l} = \delta_j^l$$

Proof:

$$\frac{\partial J}{\partial b_j^l} = \frac{\partial J}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l}$$

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l \implies \frac{\partial z_j^l}{\partial b_j^l} = 1 \implies \frac{\partial J}{\partial b_j^l} = \delta_j^l$$

4. Variation of cost function J with respect to weight

$$\frac{\partial J}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$

Proof:

$$\frac{\partial J}{\partial w_{jk}^l} = \frac{\partial J}{\partial z_j^l} \underbrace{\frac{\partial z_j^l}{\partial w_{jk}^l}}_{a_k^{l-1}} = \delta_j^l a_k^{l-1}$$
$$\underbrace{\phantom{\frac{\partial J}{\partial z_j^l}}}_{\delta_j^l}$$

Summary of the fundamental relations:

1. $\delta_j^L = \dfrac{\partial J}{\partial a_j^L} \sigma'(z_j^L) \qquad \underline{\delta}^L = \nabla_a J \odot \sigma'(\underline{z}^L)$

2. $\underline{\delta}^l = \left[(W^{l+1})^T \underline{\delta}^{l+1}\right] \odot \sigma'(\underline{z}^l)$

3. $\dfrac{\partial J}{\partial b_j^l} = \delta_j^l$

4. $\dfrac{\partial J}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$

Remark: in 2, the matrix $(W^{l+1})^T$ is the transpose of the matrix of weights of the next layer and backpropagates the error, With 1 and 2 we can compute $\underline{\delta}^l$ for all layers.

Remark: the $\underline{z}^l$ and $\underline{z}^L$ are computed in the forward procedure.

Remark: if a_k^{l-1} is small in 4, then $\dfrac{\partial J}{\partial w_{jk}^l}$ is small. This means that the weight w_{jk}^l is not updated much and the neural network will learn slowly.

Remark: in 1 we have $\sigma'(\underline{z}^L)$. If we use the sigmoid then for $z \to \pm\infty$ the function tends to 1 or 0 respectively and it is flat, hence in those situations we have $\sigma'(\underline{z}^L) \approx 0$. So in the last layer if the activation is high (≈ 1) or low (≈ 0) then the neuron will learn slowly (the neuron has **saturated**). Similarly for the biases.

17.3   Algorithm

1. $\underline{x}$ input vector $\rightarrow$ compute $\underline{a}^1$ (input layer)

2. Feedforward: for each $l = 2, 3, ..., L$ compute $\underline{z}^l$ and $\underline{a}^l$:

$$\underline{z}^l = W^l \underline{a}^{l-1} + \underline{b}^l \qquad \underline{a}^l = \sigma(\underline{z}^l)$$

3. Output error $\underline{\delta}^L$, compute it as:

$$\underline{\delta}^L = \nabla_a J \odot \sigma'(\underline{z}^L)$$

4. Backpropagate the error: for each $l = L-1, L-2, ..., 2$ compute $\underline{\delta}^l$:

$$\underline{\delta}^l = \left[(W^{l+1})^T \underline{\delta}^{l+1} \right] \odot \sigma'(\underline{z}^l)$$

5. Output values of the derivatives. Compute:

$$\frac{\partial J}{\partial b_j^l} = \delta_j^l \qquad \frac{\partial J}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$

17.4   Stochastic Gradient Descent (SGD)

Find $w^* = \arg\min_{\underline{w}} J(\underline{w})$. Until now we have considered the standard gradient descent for which the iterations where given by $\underline{w}^{(k+1)} = \underline{w}^{(k)} - \eta \nabla J(\underline{w}^{(k)})$. We can define the finite sum of J as $J(\underline{w}) = \frac{1}{N} \sum_{i=1}^{N} J_i(\underline{w})$ where N is the number of samples.

For example: $J(\underline{w}) = \frac{1}{N} \|\underline{y} - X\underline{w}\|^2 = \frac{1}{N} \sum_{i=1}^{N} \underbrace{(y_i - \underline{x}_i^\mathsf{T} \underline{w})^2}_{J_i}$ (LS).

The stochastic gradient descent method at iteration k, select a random sample $i_k \in \{1, 2, \ldots, N\}$ and update the weights as:

$$\underline{w}^{(k+1)} = \underline{w}^{(k)} - \eta \nabla J_{i_k}(\underline{w}^{(k)})$$

Where J_{i_k} is the cost function for the i_k-th sample. The important thing is that the gradient descent is computer **for only one sample at a time so this results in being N times faster than the standard GD!**.

Remark: in ML we have training data $\{(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)\}$ where $\underline{x}_i \in \mathbb{R}^d$, both N and d are large.

1D example.
We have:

$$J(w) = \frac{1}{2} \sum_{i=1}^{N} (x_i w - b_i)^2$$

Each $(x_i w - b_i)^2$ is a quadratic function. I want to minimize each J_i so i set the derivative to zero:

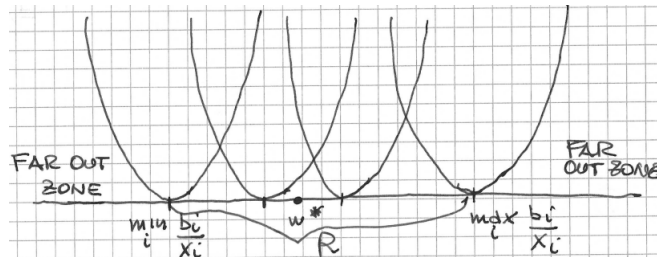$$\frac{\partial J_i}{\partial w} = 0 \implies \frac{1}{2} 2(x_i w - b_i) x_i = 0 \implies w = \frac{b_i}{x_i}$$

For the whole function we have:

$$\frac{\partial J}{\partial w} = \sum_{i=1}^{N} \frac{\partial J_i}{\partial w} = \sum_{i=1}^{N} (x_i w - b_i) x_i = 0 \implies \underline{w}^* = \frac{\sum_{i=1}^{N} b_i x_i}{\sum_{i=1}^{N} x_i^2}$$

$$w^* \in \left[\min_i \frac{b_i}{x_i}, \max_i \frac{b_i}{x_i} \right]$$

We can plot it as:

Where R is the Region of confusion. In the Far Out Zone (FOZ) ∇J_i has the same sign as ∇J. Stochastic gradient descent uses a randomized version g of ∇J and we have:

$$\mathbb{E}[g] = \nabla J \qquad \text{unbiased estimator}$$

It is important to control the variance and speed. How to choose i_k:

1. Randomly pick i_k with replacement (good theoretically)

2. Randomly pick i_k without replacement (good in practice)

17.4.1 Minibatch

It uses the following update rule:
$$\underline{w}^{(k+1)} = \underline{w}^{(k)} - \frac{\eta}{|I_k|} \sum_{j \in I_k} \nabla J_j(\underline{w}^{(k)})$$

So, basically, we are taking the average of the gradients of the samples in the minibatch. This helps reducing the variance. Very large mini-batch shrinks confusion region (reduced noises) and can create overfitting. It helps also for parallelism.

Let's continue the discussion on stochastic gradient descent method. We recall the cost function:

$$J(\underline{w}) = \frac{1}{N} \sum_{i=1}^{N} J_i(\underline{w})$$

And the algorithm is the following:

- sample i_k randomly from $\{1, \ldots, N\}$

- $\underline{w}_{k+1} = \underline{w}_k - \eta_k \underbrace{\nabla J_{i_k}(\underline{w}^{(k)})}_{g_k = \text{ stochastic gradient}}$

Theorem: the following statements are equivalent to the condition that differentiable function is μ-strongly convex:

i $f(y) \geq f(x) + \nabla f(x)^\intercal (y - x) + \frac{\mu}{2} \|y - x\|^2$

ii $g(x) = f(x) - \frac{\mu}{2} \|x\|^2$ is convex

iii $(\nabla f(x) - \nabla f(y))^\intercal (y - x) \geq \mu \|x - y\|^2$

It is trivial to check that $i \implies ii$ and $ii \implies iii$.

17.4.2 Simple convergence results for SGD

Consider two quantities:

- $J(\underline{w}^{(k+1)}) - J(\underline{w}^*)$

- $\mathbb{E}[J(\underline{w}^{(k+1)}) - J(\underline{w}^*)]$

These are two measures to evaluate the convergence of the algorithm. We can consider convergence in expectation.
Theorem: assume $\underline{w}^*$ is the minimizer of J and:

i ∇J is L-Lipschitz $\implies J(\underline{y}) \leq J(\underline{x}) + \nabla J(\underline{x})^\intercal (\underline{y} - \underline{x}) + \frac{L}{2} \|\underline{y} - \underline{x}\|^2$

ii J is μ-strongly convex $\implies J(\underline{y}) \geq J(\underline{x}) + \nabla J(\underline{x})^\intercal (\underline{y} - \underline{x}) + \frac{\mu}{2} \|\underline{y} - \underline{x}\|^2$

iii $\|\nabla J_i(\underline{x})\| \leq C \quad \forall \underline{w}, i$

iv $0 < \eta < \frac{1}{2\mu}$

v $\mathbb{E}[g_k] = \nabla J(\underline{w}^{(k)})$

Then, we have two results:

1. $\mathbb{E}[\|\underline{w}^{(k)} - \underline{w}^*\|^2] \leq (1 - 2\mu\eta)^k \|\underline{w}^{(0)} - \underline{w}^*\|^2 + \frac{\eta C^2}{2\mu}$

2. $\mathbb{E}[J(\underline{w}^{(k)}) - J(\underline{w}^*)] \leq (1 - 2\mu\eta)^k [J(\underline{w}^{(0)}) - J(\underline{w}^*)] + \frac{L\eta C^2}{4\mu}$

Let's report some considerations on these results. Firstly, we can see that, if the condition iv is verified, the quantity $(1 - 2\eta\mu)$ is smaller than 1. So, when its power k is large enough, the whole parenthesis goes to 0 and so the terms which are multiplied to it are going aswell to 0. Notice that this $\|\underline{w}^{(0)} - \underline{w}^*\|^2$ and $J(\underline{w}^{(0)}) - J(\underline{w}^*)$ are the error on the initial guess evaluated both on the values and the function. Then, there are two last additional terms which comprehend three terms that are not tunable: ν, η, L. **The only paramter we can modify is η, and once it is fixed the last fractional terms become a costant and can be seen as a noise.**

Let's now prove the first result (1):

$$\begin{aligned}
\|\underline{w}^{(k+1)} - \underline{w}^*\|^2 &= \|\underline{w}^{(k)} - \eta \nabla J_{i_k}(\underline{w}^{(k)}) - \underline{w}^*\|^2 \\
&= \|\underline{w}^{(k)} - \underline{w}^*\|^2 - 2\eta \nabla J_{i_k}(\underline{w}^{(k)})^\intercal (\underline{w}^{(k)} - \underline{w}^*) + \eta^2 \|\nabla J_{i_k}(\underline{w}^{(k)})\|^2 \\
&\leq \|\underline{w}^{(k)} - \underline{w}^*\|^2 - 2\eta \nabla J_{i_k}(\underline{w}^{(k)})^\intercal (\underline{w}^{(k)} - \underline{w}^*) + \eta^2 C^2
\end{aligned}$$

In the first step we have substitued the definition of stochastic gradient descent. Now, we use the iii condition in the last term of the formula to express the following inequality:

54

Now we take the expectation from both sides and we exploit condition v (we rewrite the expectation of the gradient) to obtain the following:

$$\mathbb{E}[\|\underline{w}^{(k+1)} - \underline{w}^*\|^2] \leq \|\underline{w}^{(k)} - \underline{w}^*\|^2 - 2\eta \nabla J_{i_k}(\underline{w}^{(k)})^\intercal (\underline{w}^{(k)} - \underline{w}^*) + \eta^2 C^2$$

Now we want to bound the term next to the gradient by means of the *iii* relation of the theorem above, the second last one. The inequality of the theorem is $(\nabla f(x) - \nabla f(y))^\intercal (x - y) \geq \mu \|x - y\|^2$ and we are going to apply it as $(\nabla J(\underline{w}^{(k)}) - \nabla J(\underline{w}^*))^\intercal (\underline{w}^{(k)} - \underline{w}^*) \geq \mu \|x - y\|^2$. But, $\nabla J(\underline{w}^*)$ is 0 because $\underline{w}^*$ is the minimizer of J. So, we can notice that the formula just written is equal to $\nabla J_{i_k}(\underline{w}^{(k)})^\intercal (\underline{w}^{(k)} - \underline{w}^*)$. So, we can substitute the term in the inequality and we obtain:

$$\mathbb{E}[\|\underline{w}^{(k+1)} - \underline{w}^*\|^2] \leq \|\underline{w}^{(k)} - \underline{w}^*\|^2 - 2\eta\mu \|\underline{w}^{(k)} - \underline{w}^*\|^2 + \eta^2 C^2$$
$$= (1 - 2\eta\mu)\|\underline{w}^{(k)} - \underline{w}^*\|^2 + \eta^2 C^2$$

Now, we can see that in the left part there is the expectation on the next iteration, while in the right part there is the error on the current iteration. So, we can iterate the formula and we obtain:

$$\mathbb{E}[\|\underline{w}^{(k)} - \underline{w}^*\|^2] \leq (1 - 2\eta\mu)^k \|\underline{w}^{(0)} - \underline{w}^*\|^2 + \sum_{i=0}^{\overset{+\infty}{\cancel{k}}} (1 - 2\eta\mu)^i \eta^2 C^2$$

Where k as upper bound of the sum is substitued with $+\infty$ because all terms in the sum are positive. Now, we can notice that the sum is a geometric series and we can compute it as:

$$\sum_{i=1}^{\infty} r^i = \frac{1}{1-r} \qquad |r| < 1$$

So if we substitute $r = 1 - 2\eta\mu$ we obtain:

$$\sum_{i=0}^{\infty} (1 - 2\eta\mu)^i \eta^2 C^2 = \frac{\eta^2 C^2}{1 - (1 - 2\eta\mu)} = \frac{\eta^2 C^2}{2\eta\mu} = \frac{\eta C^2}{2\mu}$$

And so the first result of the theorem:

$$\mathbb{E}[\|\underline{w}^{(k)} - \underline{w}^*\|^2] \leq (1 - 2\eta\mu)^k \|\underline{w}^{(0)} - \underline{w}^*\|^2 + \frac{\eta C^2}{2\mu}$$

Let's now prove the second result (2): We start from:

$$J(\underline{w}^{(k+1)}) \leq J(\underline{w}^{(k)}) + \nabla J(\underline{w}^{(k)})^\intercal (\underline{w}^{(k+1)} - \underline{w}^{(k)}) + \frac{L}{2} \|\underline{w}^{(k+1)} - \underline{w}^{(k)}\|^2$$

Now, we can substitute the definition of stochastic gradient descent and we obtain:

$$J(\underline{w}^{(k+1)}) \leq J(\underline{w}^{(k)}) + \nabla J(\underline{w}^{(k)})^\intercal (\underline{w}^{(k)} - \eta \nabla J_{i_k}(\underline{w}^{(k)}) - \underline{w}^{(k)}) + \frac{L}{2} \|\underline{w}^{(k)} - \eta \nabla J_{i_k}(\underline{w}^{(k)}) - \underline{w}^{(k)}\|^2$$
$$\leq J(\underline{w}^{(k)}) - \eta \nabla J(\underline{w}^{(k)})^\intercal \nabla J_{i_k}(\underline{w}^{(k)}) + \frac{L\eta^2 C^2}{2}$$

We subtract $J(\underline{w}^*)$ from both sides and we and take the expectation from both sides. We can notice that the expectation of the gradient is equal to the gradient of the function, i.e., thanks to property v of the last theorem $\mathbb{E}[\nabla J_{i_k}(\underline{w}^{(k)})] = \nabla J(\underline{w}^{(k)})$. So there is the product of the gradient times itself which results in its norm. Eventually, we obtain:

$$\mathbb{E}[J(\underline{w}^{(k+1)}) - J(\underline{w}^*)] \leq J(\underline{w}^{(k)}) - J(\underline{w}^*) - \eta \|\nabla J(\underline{w}^{(k)})\| + \frac{L\eta^2 C^2}{2} \quad (\dagger)$$

Now, take into account the i condition of the theorem. We want to find the terms which minimize both terms. The inequality to consider is the following:

$$J(\underline{y}) \geq J(\underline{x}) + \nabla J(\underline{x})^\intercal (\underline{y} - \underline{x}) + \frac{\mu}{2} \|\underline{y} - \underline{x}\|^2$$

We consider side separately:

- LHS: $\underline{y} = \underline{x}^*$

55

- RHS: $\underline{y} = \underline{x} - \dfrac{1}{\mu}\nabla J(\underline{x})$

To obtain the second result (RHS) you should compute the derivative wrt y and set it to 0. If you substitute the term of the RHS in the inequality you obtain:

$$J(\underline{w}^*) > J(\underline{w}) - \frac{1}{2\mu}\|\nabla J(\underline{w})\|^2$$

Going back to inequality (†) we have:

$$\mathbb{E}[J(\underline{w}^{(k+1)}) - J(\underline{w}^*)] \leq J(\underline{w}^{(k)}) - J(\underline{w}^*) - 2\eta\mu(J(\underline{w}^{(k)}) - J(\underline{w}^*)) + \frac{L\eta^2C^2}{2}$$

$$\leq (1 - 2\eta\mu)[J(\underline{w}^{(k)}) - J(\underline{w}^*)] + \frac{L\eta^2C^2}{2}$$

Now as in the previous proof, we can iterate the inequality:

$$\mathbb{E}[J(\underline{w}^{(k)}) - J(\underline{w}^*)] \leq (1 - 2\eta\mu)^k[J(\underline{w}^{(0)}) - J(\underline{w}^*)] + \sum_{i=0}^{+\infty \!\!\!\!\! /}(1 - 2\eta\mu)^i\frac{L\eta^2C^2}{2}$$

$$\leq (1 - 2\eta\mu)^k[J(\underline{w}^{(0)}) - J(\underline{w}^*)] + \frac{L\eta^2C^2}{4\mu}$$

And this concludes the proof of convergence of the stochastic gradient descent method for functions with good properties.

Until now we have considered η as fixed value. When this is true, its value value is very important because if too small the computational cost will be huge, while if too large the algorithm will not converge. Moreover, the learning rate can be adjusted not only iteration by iteration but also depending on the different parameters we have to update.

17.4.3 Line Search Procedure

Example of picking learning rate varying on the iteration (η_k):

- initial guess: $\underline{w}^{(0)}$

- pick a direction $\underline{P}_k$ such that $\underline{P}_k^\intercal \nabla J(\underline{w}^{(k)}) < 0$ (i.e direction towards the minimum)

- typically $\underline{P}_k = \dfrac{-\nabla J(\underline{w}^{(k)})}{\|\nabla J(\underline{w}^{(k)})\|}$

- $\underline{w}^{(k+1)} = \underline{w}^{(k)} + \eta_k\underline{P}_k$

- η_k is chosen to minimize ($\arg\min_\eta$) $J(\underline{w}^{(k)} + \eta_k\underline{P}_k)$

Obviously the choice of η_k is a minimization problem, that can be solved by means of analytical methods or by iterative procedures depending on the case.

Example

Consider the function:

$$J(\underline{x}) = x_1 - x_2 + 2x_1x_2 + 2x_1^2 + x_2^2$$

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \nabla J(\underline{x}) = \begin{bmatrix} 1 + 2x_2 + 4x_1 \\ -1 + 2x_1 + 2x_2 \end{bmatrix}$$

Let's define the initial guess and so the initial gradient:

$$\underline{x}^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \nabla J(\underline{x}^{(0)}) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Now, we have to solve:

$$J(\underline{x}^{(0)} - \eta\nabla J(\underline{x}^{(0)})) = J\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} - \eta\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) = J\left(\begin{bmatrix} -\eta \\ \eta \end{bmatrix}\right) = \eta^2 - 2\eta$$

Now we have to minimize this function so we compute the derivative and set it to 0:

$$\frac{d}{d\eta}J\left(\begin{bmatrix} -\eta \\ \eta \end{bmatrix}\right) = 2\eta - 2 = 0 \implies \eta = 1$$

This means that, for the first iteration, the value of the learning rate which minimizes the function is $\eta = 1$. This can be done for each iteration. This procedure is like cutting the function in one direction and then choose the learning rate which minimizes that 1D function.

17.4.4 Adagrad

Adagrad is an algorithm which adapts the learning rate to the parameters, performing smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features, and larger updates (i.e. high learning rates) for parameters associated with infrequent features. For this reason, it is well-suited for dealing with sparse data.

$$\eta = \begin{cases} \text{for rare features: } \eta \text{ large} \\ \text{for frequent features: } \eta \text{ small} \end{cases}$$

$$\eta_{k,d} \propto \frac{1}{\sum_{i=1}^{k} \nabla J_{d_i}}$$

Where k is the number of iteration, d is the feature and d_i is the feature at the i-th iteration. Drawback: the denominator is a sum of all the gradients, so the learning rate decreases over time. This means that the learning rate will be very small after a lot of iterations and it happens that the algorithm gets stuck in a certain point (i.e. we are not learning anymore).

To address this problem, **Adadelta** and **RSMProp** take some sort of average gradients.

18 Appendix

18.1 Functional Analysis

A real vector space V is a set with operations $+ : V \times V \to V$ and $\cdot : \mathbb{R} \times V \to V$ such that:

1. $u + v = v + u$

2. $(u + v) + w = u + (v + w)$

3. $\lambda \cdot (u + v) = \lambda \cdot u + \lambda \cdot v$

4. $(\lambda + \mu) \cdot u = \lambda \cdot u + \mu \cdot u$

Where $u, v, w \in V$ and $\lambda, \mu \in \mathbb{R}$. Moreover there is an element 0 such that $0 + v = v$ and an element $-v$ such that $v + (-v) = 0$.

Examples of real vector spaces are: $\mathbb{R}^n$, $P_K(I)$: set of polynomials of order $\leq K$ on the interval I, $C^k(I)$: set of functions with k continuous derivatives on the interval I.

A basis for the vector space V is the minimal set of vectors (basis function) that span V. The basis functions must be **linearly independent** and the number of basis functions is the **dimension** of the vector space. Any vector $v \in V$ can be written as a linear combination of the basis functions:

$$v = c_1 \phi_1 + c_2 \phi_2 + \cdots + c_n \phi_n$$

Example: the dimension of $P_K)I=$ is $K + 1$ and the basis functions are $\{1, x, x^2, \ldots, x^K\}$.

Let V be a vector space over $\mathbb{R}$. An inner product $(,)$ is a function $V \times V \to \mathbb{R}$ with the following properties:

1. $\forall u \in V, (u, u) \geq 0$ and $(u, u) = 0 \iff u = 0$

2. $\forall u, v \in V, (u, v) = (v, u)$

3. $\forall u, v, w \in V$ and $\forall \alpha, \beta \in \mathbb{R}, (\alpha u + \beta v, w) = \alpha(u, w) + \beta(v, w)$

V together with $(,)$ is called an **inner product space**.

Example: Given two vectors in $\mathbb{R}^2$: $v = v_1 e_1 + v_2 e_2$ and $w = w_1 e_1 + w_2 e_2$, the Euclidean inner product in $\mathbb{R}^2$ is:

$$(v, w) = v_1 w_1 + v_2 w_2$$

The extension to $\mathbb{R}^n$ is obvious.

Example: An inner product in the vector space of continuous functions in [0,1] $(C([0, 1]))$ is:

$$(f, g) = \int_0^1 f(x) g(x) dx$$

where $f, g \in C([0, 1])$.

Example: An inner product in the vector space of functions with one continuous derivative in [0,1] $(C^1([0, 1]))$ is:

$$(f, g) = \int_0^1 f(x) g(x) + f'(x) g'(x) dx$$

where $f, g \in C^1([0, 1])$.

Remark: an inner product induces a norm $\|f\| = \sqrt{(f, f)}$.

A norm on a vector space V is a mapping $\| \cdot \| : V \to \mathbb{R}$ that satisfies:

1. $\|u\| \geq 0$ and $\|u\| = 0 \iff u = 0$

2. $\|\alpha u\| = |\alpha| \|u\|$

3. $\|u + v\| \leq \|u\| + \|v\|$

This is valid $\forall u, v \in V$ and $\forall \alpha \in \mathbb{R}$.
A normed vector space is a vector space equipped with a norm.

- Norms in $\mathbb{R}^n$:

$$\|v\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p} \qquad \|v\|_1 = \sum_{i=1}^{n} |v_i| \qquad \|v\|_\infty = \max_{i=1,\dots,n} |v_i|$$

- Norms in $C^0(I)$ and $P_K(I)$:

$$\|v\|_{L^p(I)} = \left(\int_I |v|^p dx \right)^{1/p} \qquad \|v\|_{L^\infty(I)} = \sup_{x \in I} |v(x)|$$

A **bilinear form** on a vector space V is mapping a $a(\cdot, \cdot) = V \times V \to \mathbb{R}$ such that:

- $a(u + v, w) = a(u, w) + a(v, w)$

- $a(u, v + w) = a(u, v) + a(u, w)$

- $a(\lambda u, v) = \lambda a(u, v)$

- $a(u, \lambda v) = \lambda a(u, v)$

This is valid $\forall u, v, w \in V$ and $\forall \lambda \in \mathbb{R}$. The bilinear form is symmetric if $a(u, v) = a(v, u) \ \forall u, v \in V$ and continuous, or bounded, if there is a constant C such that $|a(u, v)| \leq C\|u\|\|v\| \ \forall u, v \in V$.

A symmetric bilinear form $a(\cdot, \cdot)$ is called an inner product if $a(u, u) \geq 0$ with equality if and only if $u = 0 \ \forall u \in V$. Inner products are often also denoted $(\cdot, \cdot)$. An inner product defines a so-called induced norm by:

$$\|u\|^2 = (u, u)$$

on the vector space V. In particular, $a(\cdot, \cdot)$ defines the induced energy norm $\|\|u\|\|^2 = a(u, u)$. A vector space equipped with an inner product is called an inner product space. In such spaces the Cauchy-Schwarz inequality:

$$|(u, v)| \leq \|u\|\|v\|$$

holds for all $u, v \in V$.

A **Cauchy sequence** in a normed vector space V is a sequence $\{v_i\}_{i=1}^{\infty}$ of elements $v_i \in V$ such that

$$\forall \epsilon > 0 \ \exists N \in \mathbb{N} \ \text{such that} \ \|v_i - v_j\| < \epsilon \ \forall i, j \geq N$$

A normed vector space is called complete if every Cauchy sequence in V converges to an element in V.

Remark: a convergent sequence is a Cauchy sequence. The converse is not true. Indeed, there are Cauchy sequences that do not converge in their space. For example, the sequence of rational numbers $(u_n)_{n \in \mathbb{N}} \in \mathbb{Q}$ given by:
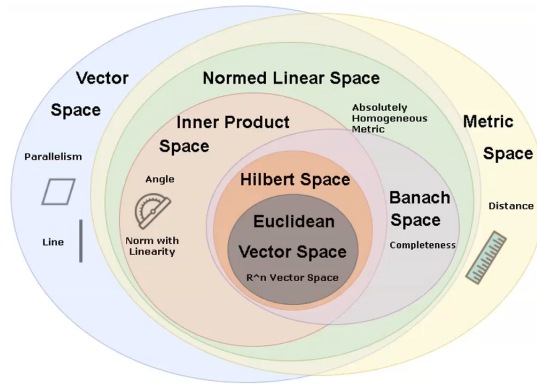
$$u_n = \frac{1}{0!} + \frac{1}{1!} + \cdots + \frac{1}{n!}$$

is Cauchy in $\mathbb{Q}$ but converges to the constant e, which belongs to $\mathbb{R}$ but not to $\mathbb{Q}$.

Definition: a vector space is said to be complete if every Cauchy sequence is also convergent.

Definition: a Banach space is a complete normed vector space.

Definition: a Hilbert space is a complete inner product space.

18.2 Riemann integral

For a function to be Riemann integrable, the infimum of the upper Riemann sum and the supremum of the lower Riemann sum should be the same.

$$m_k = \inf\{f(t) : x_{k-1} \le t \le x_k\} \qquad M_k = \sup\{f(t) : x_{k-1} \le t \le x_k\}$$

$$L(f, P) = \sum_{k=1}^{N} m_k(x_k - x_{k-1}) \qquad U(f, P) = \sum_{k=1}^{N} M_k(x_k - x_{k-1})$$

Is Riemann integral ok? Consider:

$$\mathbb{1}_{\mathbb{Q}} = \begin{cases} 1 & x \in \mathbb{Q} \\ 0 & x \notin \mathbb{Q} \end{cases}$$

In this case: $L(f, P) = 0 \ne 1 = U(f, P)$. So the function is not Riemann integrable.

18.3 Lebesque integral

Is defined as follows:

$$\int_{\mathbb{R}} \phi \, dm := \sum_{i=1}^{n} a_i m(S_i)$$

Here, instead of having a partition of the interval, we have a partition of the set. The function ϕ is the characteristic function of the set S_i and a_i is the value of the function in the set S_i. **We are still computing a "rectangular" area but this time the base is not fixed to a certain "dx" (or Δx) but it is the measure of the set S_i which can be large. The total area is not computed by stacking side to side small and tall rectangulars but by stacking on over the other larger and shorter (of height a)rectangulars.**
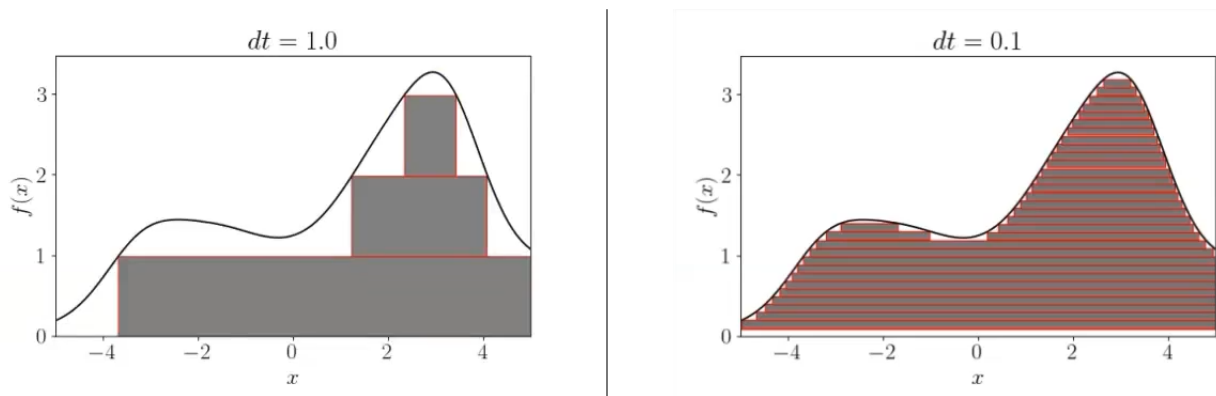
Of course, as it was done in Riemann integral, you will reach a better approximation of the area by considering thinner and thinner layers of rectangulars. **We need to define the concept of "measure"!**

Theorem: every countable set has measure 0.

The integral with Lebesque is easy: $1 \times S_1 + 0 \times S_2$ where S_1 is the set of all irrational numbers in [0,1] and S_2 is the set of rational numbers. S_2 is countable so $\mu(S_2) = 0$ moreover since S_1 and S_2 are disjoint we have $\mu([0,1]) = 1 \implies \mu(S_1) = \mu([0,1] - \mu(S_2)) = 1 - 0 = 1$. So the integral is 1!

Remark: if a function is Riemann integrable then it is also Lebesque integrable and the two integrals coincide. The converse is not true.

With the Lebesque integral we can define the L^p spaces:

$$L^p(|\Omega) = \{v : \Omega \to \mathbb{R} : \|v\|_{L^p(\Omega)} < \infty\}$$

So the space L^p is the set of functions which have norm which is bounded. The norm is defined as:

$$\|v\|_{L^p(\Omega)} = \left(\int_\Omega |v|^p dm \right)^{1/p}$$

With $1 \le p \le \infty$. In particular, if $p = \infty$ we have:

$$\|v\|_{L^\infty(\Omega)} = \sup_{x \in \Omega} |v(x)|$$

Remark: the $L^p(\Omega)$ spaces are Banach spaces for $1 \le p \le \infty$.

Remark: for $p = 2$ we have $\|v\|_{L^2(\Omega)} = (v,v)_{L^2(\Omega)}$ where $(v,w)_{L^2(\Omega)} = \int_\Omega uv dx$ is the inner product and hence $L^2(\Omega)$ is a Hilbert space. For $p \neq 2$ the norm $\|v\|_{L^p(\Omega)}$ is not given by any inner product and hence $L^p(\Omega)$ is only a Banach space.
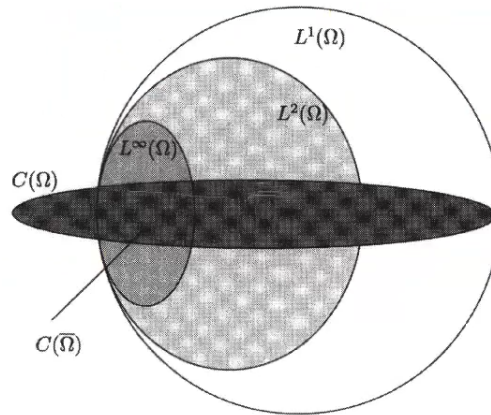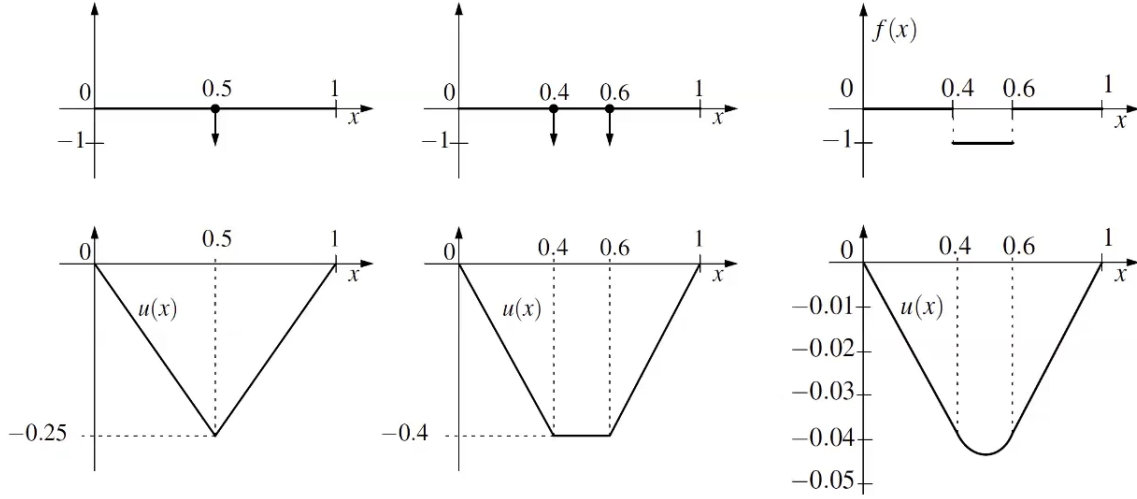

In the picture there is also $C(\Omega)$ which corresponds to the set of continuous functions in Ω. $\bar{\Omega}$ is the closure of Ω.

Suppose have this differential problem:

$$\begin{cases} -u''(x) = f(x) & x \in (0,1) \\ u(0) = u(1) = 0 \end{cases}$$

This, for example, is a model for an elastic string and f is some sort of external force applied to it. We have 3 different external forces as shown in the figure here:

In the first case the force is all concentrated in the point 0.5, in the second case the force is distributed in the two points 0.4 and 0.6 while in the third case the force is distributed in the whole interval. In the bottom of the picture there are the solutions of the three problems. But, as you can see, the functions $u(x)$ are not differentiable.

There is a contraddiction: the physical problem admits a solution which is not coherent with the regularity required by the differential problem. We have to rethink the definition of derivative, we have to introduce the concept of **weak derivative**.

18.4 Weak derivatives

Functions in Hilbert spaces are not regular enough for the standard definition of derivative to make sense.

- $C^k(\Omega)$: space of all $k < \infty$ times continuously differentiable functions in Ω.

- $\mathcal{D}(\Omega) = \{\varphi \in C^\infty(\Omega) \text{with compact support in } \Omega\}$

- $\alpha = (\alpha_1, \ldots, \alpha_d)$ is a multi-index, i.e. d-tuple of non negative integers

- $|\alpha| = \alpha_1 + \cdots + \alpha_d = \sum_{i=1}^d \alpha_i$: order of the multi-index

- $D^\alpha \phi = \prod_{i=1}^d \left(\dfrac{\partial}{\partial x_i}\right)^{\alpha_i} \varphi, \quad \varphi \in \mathcal{D}(\Omega)$

So, if we have to compute $\frac{\partial u}{\partial x_i}$ we can:

$$\int_\Omega \frac{\partial u}{\partial x_i}\varphi dx = -\int_\Omega u\frac{\partial \varphi}{\partial x_i}dx \qquad \forall \varphi \in \mathcal{D}(\Omega), \text{ for any } u \in C^1(\Omega)$$

We multiply by φ which is an infinitely regular function then by integration by parts and by noticing that φ has compact support so it is 0 outside a certain interval we can rewrite the integral and the derivative of u has been transfered over φ which is regular enough to take all the derivatives that you want. If you iterate this you can define:

$$\int_\Omega (D^\alpha u)\varphi dx = (-1)^{|\alpha|}\int_\Omega u(D^\alpha \varphi)dx \qquad \forall \varphi \in \mathcal{D}(\Omega), \text{ for any } u \in C^{|\alpha|}(\Omega)$$

In practice, we usually define the following space which is the set of functions v which are in $L^1(K)$ where K is a subset of Ω on which the function is different than 0 inside and equal to 0 outside:

$$L^1_{loc}(\Omega) = \{v : v \in L^1(K) \ \forall K \text{ compactly supported in } \Omega\}$$

Let $u \in L^1_{loc}(\Omega)$ (function u that belongs to that set), if there is a function $g \in L^1_{loc}(\Omega)$ such that:

$$\int_\Omega g\varphi dx = (-1)^{|\alpha|}\int_\Omega u(D^\alpha \varphi)dx \qquad \forall \varphi \in \mathcal{D}(\Omega)$$

then we say that g is the weak derivative $D^\alpha u$ of u. So, essentially, apart from the formalism, the idea is that the weak derivative amounts to take the derivative you want to compute, multiply by a function which is regular enough,

apply many times (α in general) the integration by parts and, due to the fact that the function φ is compactly supported, then you can end up with the final relation.

Remark: weak and classical derivatives share many properties, such as linearity, the chain rule and the differentiation of products, for instance.

Example: Let $\Omega = (-1, 1)$ and let $u = 3 - |x|$; the weak derivative $g = D^1 u$ is given by:

$$g = \begin{cases} 1 & -1 < x \leq 0 \\ -1 & 0 < x < 1 \end{cases}$$

Since:

$$-\int_{-1}^{1} u D^1 \varphi dx = -\int_{-1}^{0} u D^1 \varphi dx - \int_{0}^{1} u D^1 \varphi dx$$

$$= \int_{-1}^{0} D^1 u \varphi dx - \lfloor u\varphi \rfloor_{-1}^{0} + \int_{0}^{1} D^1 u \varphi dx - \lfloor u\varphi \rfloor_{0}^{1}$$

$$= \int_{-1}^{0} \varphi dx - (3\varphi(0) - 2\varphi(-1)) - \int_{0}^{1} \varphi dx - (2\varphi(1) - 3\varphi(0))$$

$$= \int_{-1}^{1} g\varphi dx$$

Where we have that $\varphi(-1) = \varphi(1) = 0$ since φ has compact support.

18.5   Sobolev spaces

$W_k^p(\Omega)$ is the space of $L^p(\Omega)$ functions u whose weak derivatives $D^\alpha u$ with $|\alpha| \leq k$ also lies in $L^p(\Omega)$ i.e.:

$$W_k^p(\Omega) = \{u \in L_{loc}^1(\Omega) : \|u\|_{W_k^p(\Omega)} < \infty\}$$

where

$$\|u\|_{W_k^p(\Omega)} = \left(\sum_{|\alpha| \leq k} \|D^\alpha u\|_{L^p(\Omega)}^p \right)^{1/p} \qquad 1 \leq p \leq \infty$$

$$\|u\|_{W_k^p(\Omega)} = \max_{|\alpha| \leq k} \|D^\alpha u\|_{L^\infty(\Omega)} \qquad p = \infty$$

Remark: Sobolev spaces are Banach spaces for $1 \leq p \leq \infty$, for $p = 2$, $W_k^2(\Omega)$ is also a Hilbert space. The case for $p = 2$ is the most common in Finite Element analysis and we will use the notation $H^k(\Omega) = W_k^2(\Omega)$. For $k = 1$ we have:

$$H^1(\Omega) = \{v \in L^2(\Omega) : D^1 u \in L^2(\Omega)\}$$

with the following inner product and norm:

$$(u, v)_{H^1(\Omega)} = (u, v)_{L^2(\Omega)} + (\nabla u, \nabla v)_{L^2(\Omega)}$$

$$\|u\|_{H^1(\Omega)}^2 = \|u\|_{L^2(\Omega)}^2 + \|\nabla u\|_{L^2(\Omega)}^2$$

Remark: for $H^1(\Omega)$ it can be shown that its functions are continuous for $d = 1$. May lack values at certain isolated points for $d = 2$ and be discontinuous along a curve for $d = 3$.

Definition: A **linear operator** T is an operator whose domain X is a vector space and for which we have:

$$T(u + v) = T(u) + T(v), \quad \forall u, v \in X$$

$$T(\alpha u) = \alpha T(u), \quad \forall u \in X, \alpha \in \mathbb{R}$$

Let $T : X \to Y$ be a linear operator, we say that T is **bounded** if it is possible to find a number $K > 0$ such that:

$$\|Tu\| \leq K\|u\|, \quad \forall u \in X$$

The norm of a linear operator is defined as:

$$\|T\| = \sup_{u \in X, u \neq 0} \frac{\|Tu\|}{\|u\|}$$

In general it can be proven that the set $\mathcal{L}(\mathcal{X}, \mathcal{Y})$ of all bounded linear operators from a normed space X to a normed space Y is itself a normed space with the previous norm. When $Y = \mathbb{R}$, $\mathcal{L}(\mathcal{X}, \mathcal{Y})$ is the space of **bounded linear functionals** on X, and it is called the **dual space** of X (denoted with X'). Moreover

$$\|l\|_{X'} = \sup_{v \in X, v \neq 0} \frac{\|l(v)\|}{\|v\|}$$

Theorem (Riesz representation theorem): let H be a Hilbert space and l a bounded linear functional on H. Then there exists a unique element u in H such that:

$$l(v) = (v, v) \quad \forall v \in H$$

Moreover $\|l\| = \|u\|$.

Example: Let f be a linear functional on $\mathbb{R}^n$, then $f(x)$ is a real number and it is possible to find a point $y \in \mathbb{R}^n$ such that:
$$f(x) = x \cdot y$$

For example if f is defined by:
$$f(x) = x_1 + \cdots + x_n, \quad x \in \mathbb{R}^n$$

then $y = (1, \ldots, 1)$

Example: Let f be a linera functional on $L^2(0, 1)$ defined by:

$$f : L^2 \to \mathbb{R}, \quad f(v) = \int_0^{1/2} v(x) dx$$

Then there exists a unique $u \in L^2(0, 1)$ with the property that

$$f(v) = (u, v) \text{ or } \int_0^1 u(x)v(x)dx = \int_0^{1/2} v(x)dx$$

Clearly we have

$$u(x) = \begin{cases} 1, & 0 < x \leq 1/2 \\ 0, & 1/2 < x < 1 \end{cases}$$