Deep Learning & Applied AI

Linear regression, convexity, and gradients

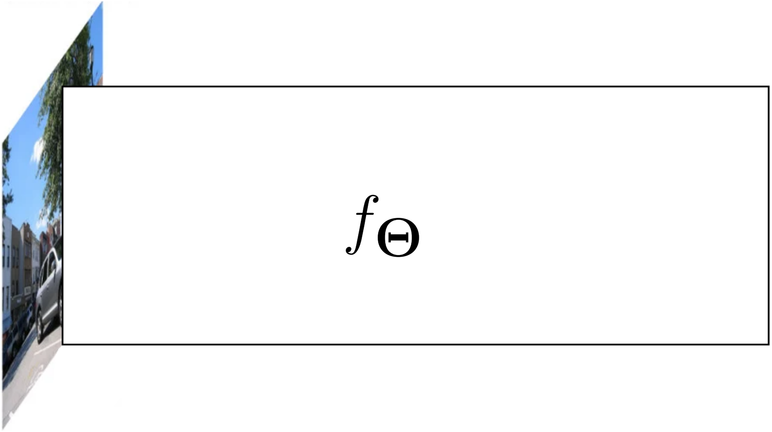
Emanuele Rodolà
rodola@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

2nd semester a.y. 2019/2020 · March 09, 2020

A glimpse into neural networks

In deep learning, we deal with highly parametrized models called deep neural networks:



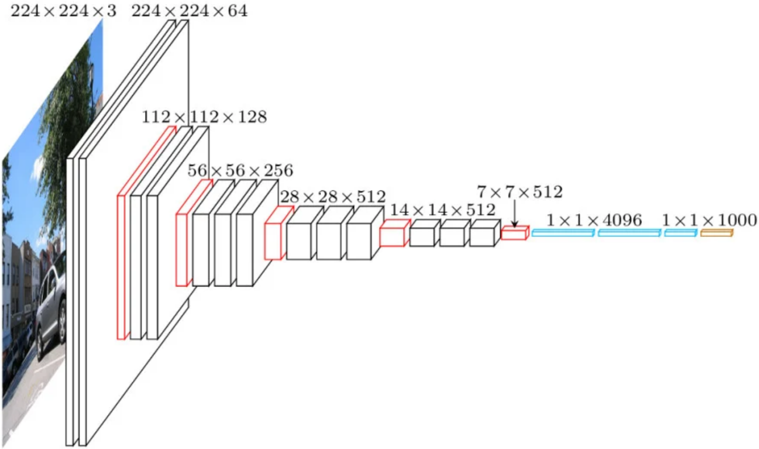
A glimpse into neural networks

In deep learning, we deal with highly parametrized models called deep neural networks:

$$f_{\boldsymbol{\Theta}}(\mathbf{x}) = \mathbf{y}$$

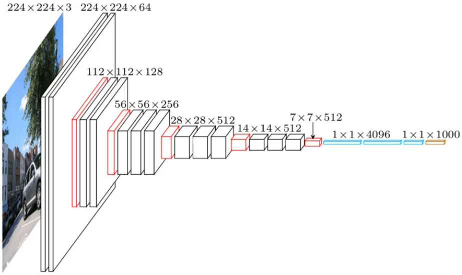
A glimpse into neural networks

In deep learning, we deal with highly parametrized models called deep neural networks:



A glimpse into neural networks

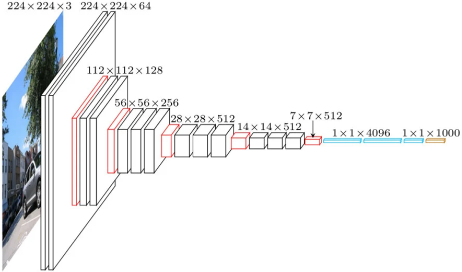
In deep learning, we deal with highly parametrized models called deep neural networks:



- Each block has a predefined structure (e.g., a linear map)

A glimpse into neural networks

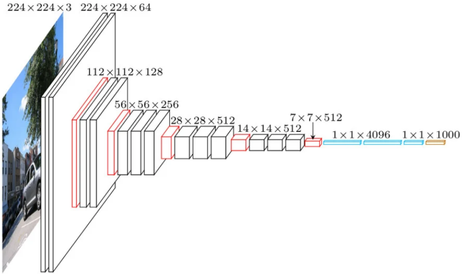
In deep learning, we deal with highly parametrized models called deep neural networks:



- Each block has a predefined structure (e.g., a linear map)
- Each block is defined in terms of unknown parameters θ

A glimpse into neural networks

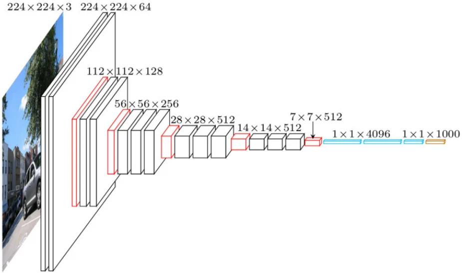
In deep learning, we deal with highly parametrized models called deep neural networks:



- Each block has a predefined structure (e.g., a linear map)
- Each block is defined in terms of unknown parameters θ
- Finding the parameter values is called training...

A glimpse into neural networks

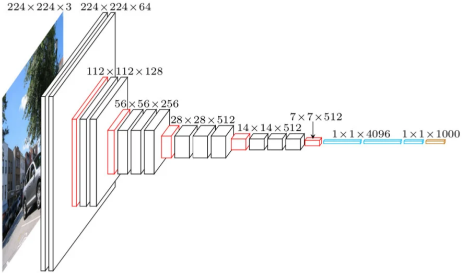
In deep learning, we deal with highly parametrized models called deep neural networks:



- Each block has a predefined structure (e.g., a linear map)
- Each block is defined in terms of unknown parameters θ
- Finding the parameter values is called training...
- ...which is done by minimizing a function called loss

A glimpse into neural networks

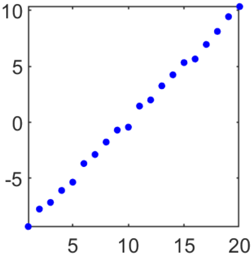
In deep learning, we deal with highly parametrized models called deep neural networks:



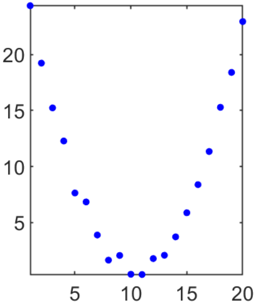
- Each block has a predefined structure (e.g., a linear map)
- Each block is defined in terms of unknown parameters θ
- Finding the parameter values is called training...
- ...which is done by minimizing a function called loss
- Minimization requires computing gradients, called backpropagation

Parametrized models

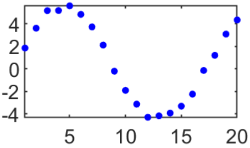
The parameters describe the behavior of the network, and must be solved for.



$$y = ax + b$$



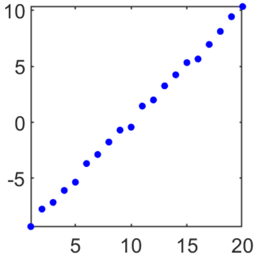
$$y = ax^2 + bx + c$$

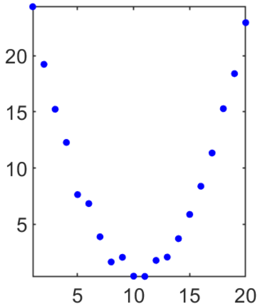


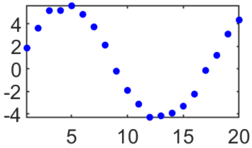
$$y = a\sin(x) + bx + c$$

Parametrized models

The parameters describe the behavior of the network, and must be solved for.



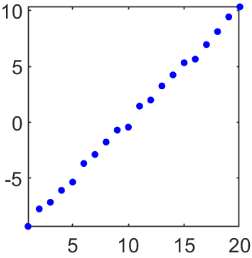


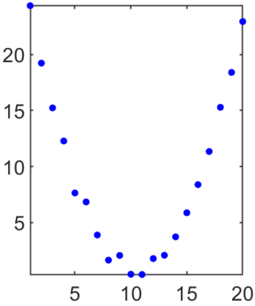


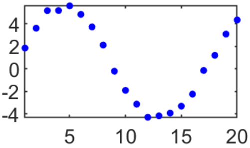
$$f_{a,b}(x) = ax + b$$

Parametrized models

The parameters describe the behavior of the network, and must be solved for.







$f_{\Theta_f}(x)$
$\Theta_f \equiv \{a, b\}$

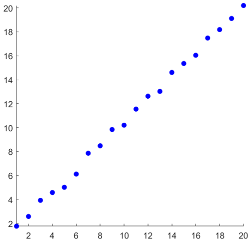
$g_{\Theta_g}(x)$

$h_{\Theta_h}(x)$

From a technical standpoint, our task is to determine the parameters Θ.

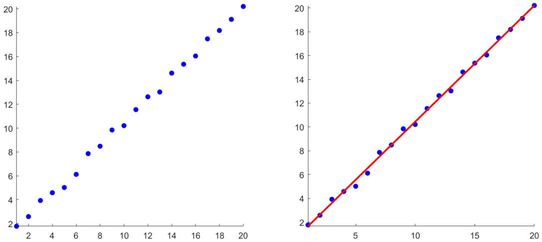
Linear regression

We start from the simplest non-trivial case for a learning model:



Linear regression

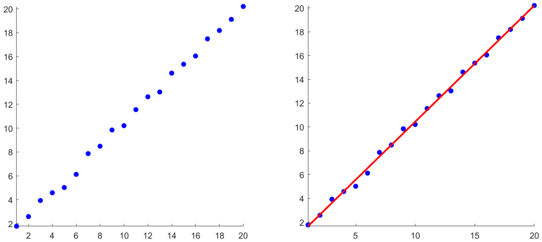
We start from the simplest non-trivial case for a learning model:



$$y_i = ax_i + b$$

Linear regression

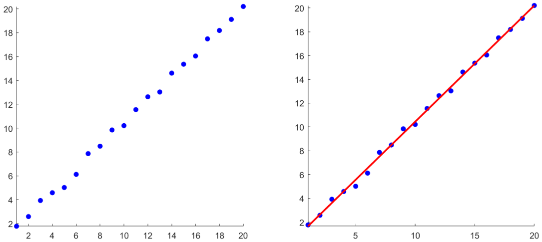
We start from the simplest non-trivial case for a learning model:



$$y_i = ax_i + b + \text{noise}$$

Linear regression

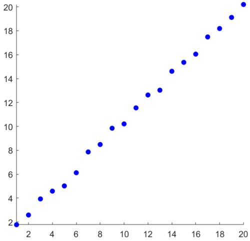
We start from the simplest non-trivial case for a learning model:

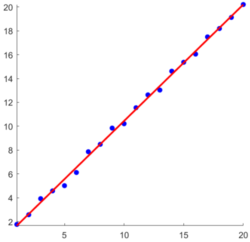


$$y_i = ax_i + b + \cancel{\text{noise}}$$

Linear regression

We start from the simplest non-trivial case for a learning model:





$$f_\Theta(x_i) = y_i$$

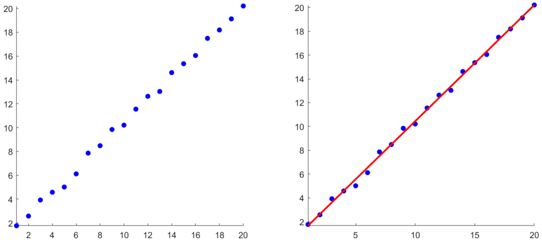
Model: linear + bias (we ignore the noise)

Parameters: $\Theta = \{a, b\}$

Data: n pairs (x_i, y_i); the x_i are called the regressors

Linear regression

We start from the simplest non-trivial case for a learning model:



$$f_\Theta(x_i) = y_i$$

Model: linear + bias (we ignore the noise)

Parameters: $\Theta = \{a, b\}$

Data: n pairs (x_i, y_i); the x_i are called the regressors

Given a and b, we have a mapping that gives new output from new input.

Linear regression

The equations:

$$f_\Theta(x_i) = y_i$$

must approximately hold for all $i = 1, \ldots, n$.

Linear regression

The equations:

$$f_\Theta(x_i) = y_i$$

must approximately hold for all $i = 1, \ldots, n$.

Problem: Choose a and b minimizing the mean squared error (MSE) between input and predicted output:

$$\epsilon = \min_{a,b \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f_\Theta(x_i))^2$$

where $\Theta = \{a, b\}$.

Linear regression

The equations:

$$f_\Theta(x_i) = y_i$$

must approximately hold for all $i = 1, \ldots, n$.

Problem: Choose a and b minimizing the mean squared error (MSE) between input and predicted output:

$$\epsilon = \min_{a,b \in \mathbb{R}} \sum_{i=1}^{n} (y_i - f_\Theta(x_i))^2$$

where $\Theta = \{a, b\}$.

Linear regression

The equations:

$$f_\Theta(x_i) = y_i$$

must approximately hold for all $i = 1, \dots, n$.

Problem: Choose a and b minimizing the mean squared error (MSE) between input and predicted output:

$$\epsilon = \min_{a,b \in \mathbb{R}} \sum_{i=1}^{n} (y_i - f_\Theta(x_i))^2$$

where $\Theta = \{a, b\}$.

When f_Θ is linear, this is called a least-squares approximation problem.

Linear regression: Loss function

The equations:

$$f_\Theta(x_i) = y_i$$

must approximately hold for all $i = 1, \ldots, n$.

Problem: Choose a and b minimizing the mean squared error (MSE) between input and predicted output:

$$\epsilon = \min_\Theta \ell_\Theta(\{x_i, y_i\})$$

The error criterion w.r.t. the parameters is also called a loss function, usually denoted by ℓ:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} (y_i - f_\Theta(x_i))^2$$

Linear regression: Loss function

The equations:

$$f_\Theta(x_i) = y_i$$

must approximately hold for all $i = 1, \ldots, n$.

Problem: Choose a and b minimizing the mean squared error (MSE) between input and predicted output:

$$\epsilon = \min_\Theta \ell_\Theta(\{x_i, y_i\})$$

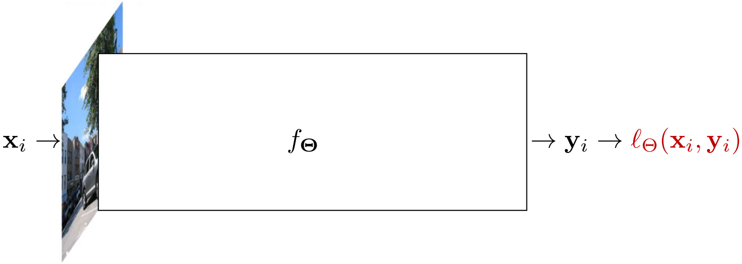
The error criterion w.r.t. the parameters is also called a loss function, usually denoted by ℓ:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} (y_i - f_\Theta(x_i))^2$$

Remark: We minimize the loss w.r.t. the parameters Θ, and **not** w.r.t. the data (x_i, y_i). Also, the loss is defined on the entire dataset, not on just one data point.

Linear regression

We are considering the following case:



where $f_{\mathbf{\Theta}}$ is linear, and ℓ_{Θ} is quadratic.

Optimization

We need to solve the general minimization problem:

$$\epsilon = \min_{\Theta} \ell(\Theta)$$

Optimization

We need to solve the general minimization problem:

$$\epsilon = \min_{\Theta} \ell(\Theta)$$

In particular, we are interested in the minimizer Θ.

Optimization

We need to solve the general minimization problem:

$$\epsilon = \min_{\Theta} \ell(\Theta)$$

In particular, we are interested in the minimizer Θ.

Finding minimizers for general ℓ is an open problem. The research area is broadly called optimization.

In general, the optimization method depends on the properties of ℓ.

Optimization

We need to solve the general minimization problem:

$$\epsilon = \min_{\Theta} \ell(\Theta)$$

In particular, we are interested in the minimizer Θ.

Finding minimizers for general ℓ is an open problem. The research area is broadly called optimization.

In general, the optimization method depends on the properties of ℓ.

We will mostly deal with unconstrained problems.

Convex functions

Jensen's inequality:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$$

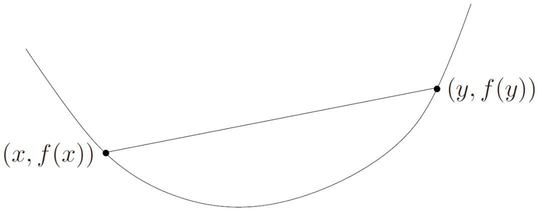
for all x, y and $\alpha \in [0, 1]$

Convex functions

Jensen's inequality:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$$

for all x, y and $\alpha \in [0,1]$

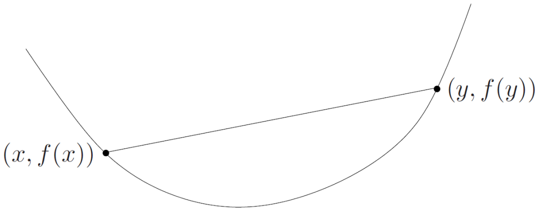


Convex functions

Jensen's inequality:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$$

for all x, y and $\alpha \in [0, 1]$



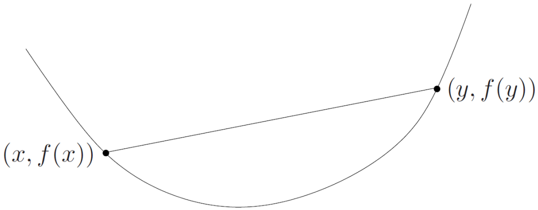
Let us further assume that f is a differentiable function, so that we can compute its derivative $\frac{df}{dx}$ at all points x.

Convex functions

Jensen's inequality:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$$

for all x, y and $\alpha \in [0,1]$



Let us further assume that f is a differentiable function, so that we can compute its derivative $\frac{df}{dx}$ at all points x.

Intuition tells us that the minimizer x is where $\frac{df(x)}{dx} = 0$.

Convex functions: Global minima

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$$

for all x, y and $\alpha \in (0,1)$

Convex functions: Global minima

$$f(x + \alpha(y - x)) \leq (1 - \alpha)f(x) + \alpha f(y)$$

for all x, y and $\alpha \in (0, 1)$

Convex functions: Global minima

$$\frac{f(x+\alpha(y-x))}{\alpha} \leq \frac{(1-\alpha)f(x)+\alpha f(y)}{\alpha}$$

for all x, y and $\alpha \in (0,1)$

Convex functions: Global minima

$$\frac{f(x+\alpha(y-x))}{\alpha} \leq \frac{f(x)}{\alpha} - f(x) + f(y)$$

for all x, y and $\alpha \in (0,1)$

Convex functions: Global minima

$$\frac{f(x + \alpha(y - x)) - f(x)}{\alpha} + f(x) \leq f(y)$$

for all x, y and $\alpha \in (0, 1)$

Convex functions: Global minima

$$\lim_{\alpha \to 0} \frac{f(x + \alpha(y - x)) - f(x)}{\alpha} + f(x) \leq f(y)$$

Convex functions: Global minima

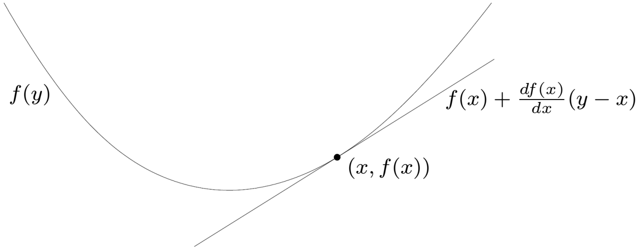
$$\lim_{\alpha \to 0} \frac{f(x + \alpha(y - x)) - f(x)}{\alpha(y - x)}(y - x) + f(x) \leq f(y)$$

Convex functions: Global minima

$$\frac{df(x)}{dx}(y-x) + f(x) \leq f(y)$$

Convex functions: Global minima

$$\underbrace{\frac{df(x)}{dx}(y-x)+f(x)}_{\text{1st}-\text{order Taylor at } f(x)} \leq f(y)$$



Thus, if $\frac{df(x)}{dx} = 0$:

$$f(x) \leq f(y)$$

and x is a global minimizer of f.

Convex functions on $\mathbb{R}^n$

In deep learning we deal with functions over $n \gg 1$ parameters:

$$f : \mathbb{R}^n \to \mathbb{R}$$

Convex functions on $\mathbb{R}^n$

In deep learning we deal with functions over $n \gg 1$ parameters:

$$f : \mathbb{R}^n \to \mathbb{R}$$

The notion of derivative is replaced by the notion of gradient:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

which is the vector of partial derivatives of f.

Convex functions on $\mathbb{R}^n$

In deep learning we deal with functions over $n \gg 1$ parameters:

$$f : \mathbb{R}^n \to \mathbb{R}$$

The notion of derivative is replaced by the notion of gradient:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

which is the vector of partial derivatives of f.

Convexity is defined as before:

$$f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y})$$

Convex functions on $\mathbb{R}^n$

In deep learning we deal with functions over $n \gg 1$ parameters:

$$f : \mathbb{R}^n \to \mathbb{R}$$

The notion of derivative is replaced by the notion of gradient:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

which is the vector of partial derivatives of f.

Convexity is defined as before:

$$f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y})$$

and we also have the global optimality condition:

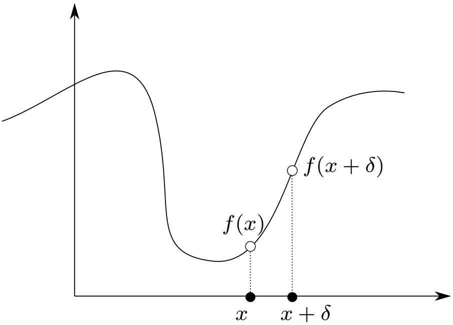
$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \mathbf{0} \quad \Longrightarrow \quad f(\mathbf{x}) \leq f(\mathbf{y}) \text{ for all } \mathbf{y} \in \mathbb{R}^n$$

The gradient

The gradient $\nabla_{\mathbf{x}} f(\mathbf{x})$ encodes the direction of steepest ascent of f at point $\mathbf{x}$.

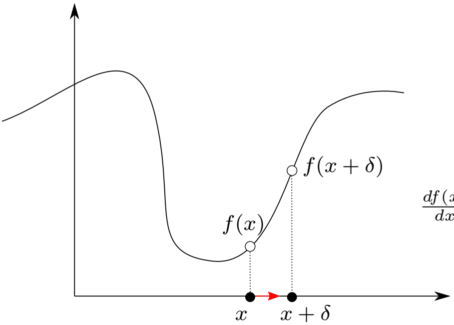
The gradient

The gradient $\nabla_{\mathbf{x}} f(\mathbf{x})$ encodes the direction of steepest ascent of f at point $\mathbf{x}$. In the simple 1D case:



The gradient

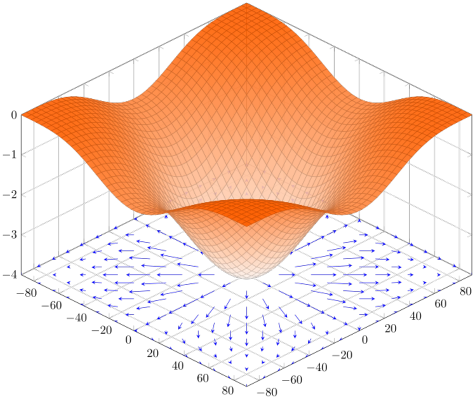
The gradient $\nabla_{\mathbf{x}} f(\mathbf{x})$ encodes the direction of steepest ascent of f at point $\mathbf{x}$. In the simple 1D case:



$$\frac{df(x)}{dx} = \lim_{\delta \to 0} \frac{f(x+\delta) - f(x)}{\delta}$$

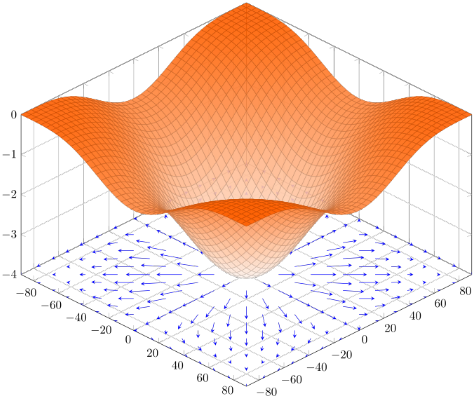
The gradient

The gradient $\nabla_{\mathbf{x}} f(\mathbf{x})$ encodes the direction of steepest ascent of f at point $\mathbf{x}$. In the more general case:



The gradient

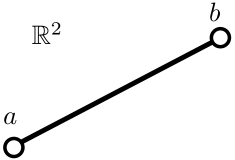
The gradient $\nabla_{\mathbf{x}} f(\mathbf{x})$ encodes the direction of steepest ascent of f at point $\mathbf{x}$. In the more general case:



The length of the gradient vector encodes its strength.

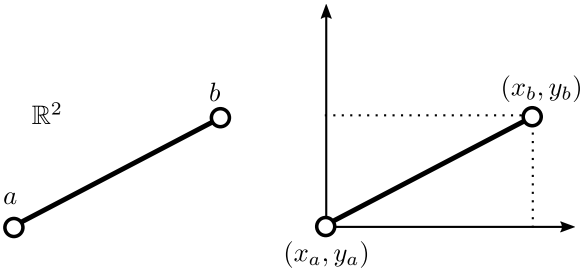
Vector lengths

The Euclidean distance measures the length of a straight line connecting two points:



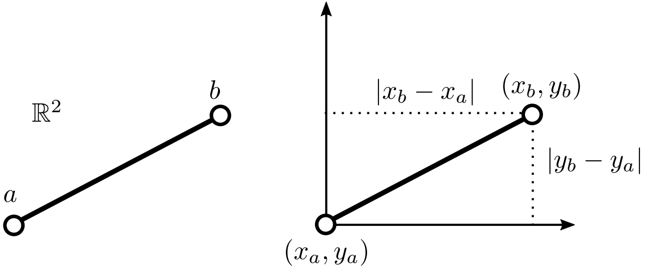
Vector lengths

The Euclidean distance measures the length of a straight line connecting two points:



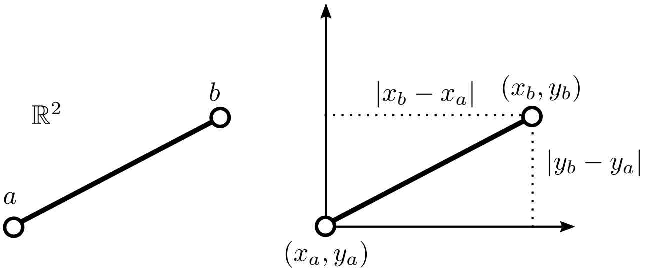
Vector lengths

The Euclidean distance measures the length of a straight line connecting two points:



Vector lengths

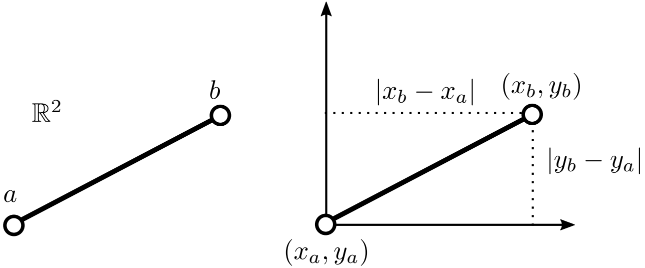
The Euclidean distance measures the length of a straight line connecting two points:



Apply Pythagoras' theorem: $d(a,b) = (|x_b - x_a|^2 + |y_b - y_a|^2)^{\frac{1}{2}}$

Vector lengths

The Euclidean distance measures the length of a straight line connecting two points:



Apply Pythagoras' theorem: $d(a, b) = (|x_b - x_a|^2 + |y_b - y_a|^2)^{\frac{1}{2}}$

In matrix notation:

$$d(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_2$$

where $\mathbf{a} = \begin{pmatrix} x_a \\ y_a \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} x_b \\ y_b \end{pmatrix}$

L_p distance in $\mathbb{R}^k$

One can generalize to different power coefficients $p \geq 1$:

$$\|\mathbf{x} - \mathbf{y}\|_2 = \left(|x_1 - y_1|^2 + |x_2 - y_2|^2\right)^{\frac{1}{2}}$$

$$\Downarrow$$

$$\|\mathbf{x} - \mathbf{y}\|_p = \left(|x_1 - y_1|^p + |x_2 - y_2|^p\right)^{\frac{1}{p}}$$

L_p distance in $\mathbb{R}^k$

One can generalize to different power coefficients $p \geq 1$:

$$\|\mathbf{x} - \mathbf{y}\|_2 = (|x_1 - y_1|^2 + |x_2 - y_2|^2)^{\frac{1}{2}}$$
$$\Downarrow$$
$$\|\mathbf{x} - \mathbf{y}\|_p = (|x_1 - y_1|^p + |x_2 - y_2|^p)^{\frac{1}{p}}$$

As well as generalize from $\mathbb{R}^2$ to $\mathbb{R}^k$:

$$\|\mathbf{x} - \mathbf{y}\|_p = \left(\sum_{i=1}^{k} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

L_p distance in $\mathbb{R}^k$

One can generalize to different power coefficients $p \geq 1$:

$$\|\mathbf{x} - \mathbf{y}\|_2 = \left(|x_1 - y_1|^2 + |x_2 - y_2|^2\right)^{\frac{1}{2}}$$
$$\Downarrow$$
$$\|\mathbf{x} - \mathbf{y}\|_p = \left(|x_1 - y_1|^p + |x_2 - y_2|^p\right)^{\frac{1}{p}}$$

As well as generalize from $\mathbb{R}^2$ to $\mathbb{R}^k$:

$$\|\mathbf{x} - \mathbf{y}\|_p = \left(\sum_{i=1}^{k} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

This definition gives us the L_p distance between vectors in $\mathbb{R}^k$.

L_p distance in $\mathbb{R}^k$

One can generalize to different power coefficients $p \geq 1$:

$$\|\mathbf{x} - \mathbf{y}\|_2 = (|x_1 - y_1|^2 + |x_2 - y_2|^2)^{\frac{1}{2}}$$
$$\Downarrow$$
$$\|\mathbf{x} - \mathbf{y}\|_p = (|x_1 - y_1|^p + |x_2 - y_2|^p)^{\frac{1}{p}}$$

As well as generalize from $\mathbb{R}^2$ to $\mathbb{R}^k$:

$$\|\mathbf{x} - \mathbf{y}\|_p = \left(\sum_{i=1}^{k} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

This definition gives us the L_p distance between vectors in $\mathbb{R}^k$.

The length (or norm) of a vector is simply its distance from the origin:

$$\|\mathbf{x} - \mathbf{0}\|_2 = \|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{k} |x_i|^2}$$

L_p distance in $\mathbb{R}^k$

One can generalize to different power coefficients $p \geq 1$:

$$\|\mathbf{x} - \mathbf{y}\|_2 = (|x_1 - y_1|^2 + |x_2 - y_2|^2)^{\frac{1}{2}}$$
$$\Downarrow$$
$$\|\mathbf{x} - \mathbf{y}\|_p = (|x_1 - y_1|^p + |x_2 - y_2|^p)^{\frac{1}{p}}$$

As well as generalize from $\mathbb{R}^2$ to $\mathbb{R}^k$:

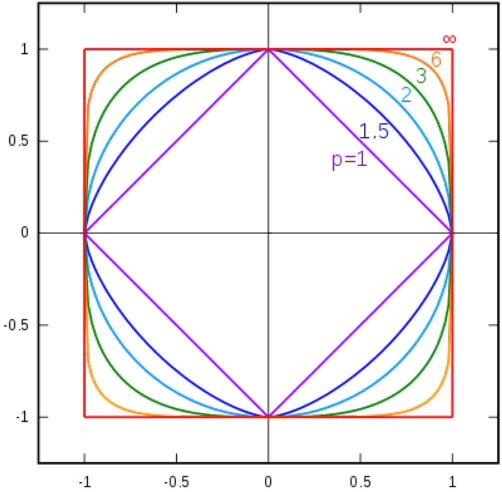
$$\|\mathbf{x} - \mathbf{y}\|_p = \left(\sum_{i=1}^k |x_i - y_i|^p\right)^{\frac{1}{p}}$$

This definition gives us the L_p distance between vectors in $\mathbb{R}^k$.

The length (or norm) of a vector is simply its distance from the origin:

$$\|\mathbf{x} - \mathbf{0}\|_2 = \|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^k |x_i|^2} = \sqrt{\mathbf{x}^\top \mathbf{x}}$$

L_p unit balls in $\mathbb{R}^2$



Convex functions: Examples

So, for convex functions $f(\mathbf{x})$, a global minimizer $\mathbf{x}$ is found by setting:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$$

and then solving for $\mathbf{x}$.

Convex functions: Examples

So, for convex functions $f(\mathbf{x})$, a global minimizer $\mathbf{x}$ is found by setting:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$$

and then solving for $\mathbf{x}$.

Some convex functions appearing in deep learning (check convexity!):

- $f(x) = |x|^p$ for $p \geq 1$
- $f(x) = x \log x$ for $x > 0$

Convex functions: Examples

So, for convex functions $f(\mathbf{x})$, a global minimizer $\mathbf{x}$ is found by setting:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$$

and then solving for $\mathbf{x}$.

Some convex functions appearing in deep learning (check convexity!):

- $f(x) = |x|^p$ for $p \geq 1$
- $f(x) = x \log x$ for $x > 0$
- every norm on $\mathbb{R}^n$, e.g., $f(\mathbf{x}) = \|\mathbf{x}\|_2$

Convex functions: Examples

So, for convex functions $f(\mathbf{x})$, a global minimizer $\mathbf{x}$ is found by setting:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$$

and then solving for $\mathbf{x}$.

Some convex functions appearing in deep learning (check convexity!):

- $f(x) = |x|^p$ for $p \geq 1$
- $f(x) = x \log x$ for $x > 0$
- every norm on $\mathbb{R}^n$, e.g., $f(\mathbf{x}) = \|\mathbf{x}\|_2$
- $f(\mathbf{x}) = \max_i \{x_i\} = \max\{x_1, \ldots, x_n\}$

Convex functions: Examples

So, for convex functions $f(\mathbf{x})$, a global minimizer $\mathbf{x}$ is found by setting:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$$

and then solving for $\mathbf{x}$.

Some convex functions appearing in deep learning (check convexity!):

- $f(x) = |x|^p$ for $p \geq 1$
- $f(x) = x \log x$ for $x > 0$
- every norm on $\mathbb{R}^n$, e.g., $f(\mathbf{x}) = \|\mathbf{x}\|_2$
- $f(\mathbf{x}) = \max_i \{x_i\} = \max\{x_1, \ldots, x_n\}$

$$f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) = \max_i \{\alpha x_i + (1-\alpha) y_i\}$$

Convex functions: Examples

So, for convex functions $f(\mathbf{x})$, a global minimizer $\mathbf{x}$ is found by setting:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$$

and then solving for $\mathbf{x}$.

Some convex functions appearing in deep learning (check convexity!):

- $f(x) = |x|^p$ for $p \geq 1$
- $f(x) = x \log x$ for $x > 0$
- every norm on $\mathbb{R}^n$, e.g., $f(\mathbf{x}) = \|\mathbf{x}\|_2$
- $f(\mathbf{x}) = \max_i\{x_i\} = \max\{x_1, \ldots, x_n\}$

$$\begin{aligned} f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y}) &= \max_i\{\alpha x_i + (1-\alpha) y_i\} \\ &\leq \alpha \max_i\{x_i\} + (1-\alpha) \max_i\{y_i\} \end{aligned}$$

Convex functions: Examples

So, for convex functions $f(\mathbf{x})$, a global minimizer $\mathbf{x}$ is found by setting:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$$

and then solving for $\mathbf{x}$.

Some convex functions appearing in deep learning (check convexity!):

- $f(x) = |x|^p$ for $p \geq 1$
- $f(x) = x \log x$ for $x > 0$
- every norm on $\mathbb{R}^n$, e.g., $f(\mathbf{x}) = \|\mathbf{x}\|_2$
- $f(\mathbf{x}) = \max_i\{x_i\} = \max\{x_1, \dots, x_n\}$

$$\begin{aligned} f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) &= \max_i\{\alpha x_i + (1-\alpha) y_i\} \\ &\leq \alpha \max_i\{x_i\} + (1-\alpha) \max_i\{y_i\} \\ &= \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}) \end{aligned}$$

Linear regression: Finding a solution

$$\min_{a,b\in\mathbb{R}} \sum_{i=1}^{n}(y_i - ax_i - b)^2$$

Linear regression: Finding a solution

$$\mathbf{\Theta}^* = \arg \min_{\mathbf{\Theta} \in \mathbb{R}^2} \ell(\mathbf{\Theta})$$

where $\ell : \mathbb{R}^2 \to \mathbb{R}$ is defined as:

$$\ell(a, b) = \sum_{i=1}^{n} (y_i - a x_i - b)^2$$

Linear regression: Finding a solution

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta} \in \mathbb{R}^2} \ell(\mathbf{\Theta})$$

where $\ell : \mathbb{R}^2 \to \mathbb{R}$ is defined as:

$$\ell(a, b) = \sum_{i=1}^{n} (y_i - a x_i - b)^2$$

A solution is found by setting $\nabla_{\mathbf{\Theta}} \ell(\mathbf{\Theta}) = \mathbf{0}$:

$$\nabla_{\mathbf{\Theta}} \sum_{i=1}^{n} (y_i - a x_i - b)^2 = \sum_{i=1}^{n} \nabla_{\mathbf{\Theta}} (y_i - a x_i - b)^2$$

Linear regression: Finding a solution

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}\in\mathbb{R}^2} \ell(\mathbf{\Theta})$$

where $\ell : \mathbb{R}^2 \to \mathbb{R}$ is defined as:

$$\ell(a, b) = \sum_{i=1}^{n} (y_i - ax_i - b)^2$$

A solution is found by setting $\nabla_{\mathbf{\Theta}}\ell(\mathbf{\Theta}) = \mathbf{0}$:

$$\begin{aligned}\nabla_{\mathbf{\Theta}} \sum_{i=1}^{n} (y_i - ax_i - b)^2 &= \sum_{i=1}^{n} \nabla_{\mathbf{\Theta}} (y_i - ax_i - b)^2 \\ &= \sum_{i=1}^{n} \nabla_{\mathbf{\Theta}} (\cancel{y_i^2} + a^2x_i^2 + b^2 - 2ax_iy_i - 2by_i + 2abx_i)\end{aligned}$$

Linear regression: Finding a solution

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}\in\mathbb{R}^2} \ell(\mathbf{\Theta})$$

where $\ell : \mathbb{R}^2 \to \mathbb{R}$ is defined as:

$$\ell(a,b) = \sum_{i=1}^{n} (y_i - ax_i - b)^2$$

A solution is found by setting $\nabla_{\mathbf{\Theta}}\ell(\mathbf{\Theta}) = \mathbf{0}$:

$$\begin{aligned}
\nabla_{\mathbf{\Theta}} \sum_{i=1}^{n} (y_i - ax_i - b)^2 &= \sum_{i=1}^{n} \nabla_{\mathbf{\Theta}} (y_i - ax_i - b)^2 \\
&= \sum_{i=1}^{n} \nabla_{\mathbf{\Theta}} (\cancel{y_i^2} + a^2x_i^2 + b^2 - 2ax_iy_i - 2by_i + 2abx_i) \\
&= \sum_{i=1}^{n} \begin{pmatrix} 2ax_i^2 - 2x_iy_i + 2bx_i \\ 2b - 2y_i + 2ax_i \end{pmatrix}
\end{aligned}$$

Linear regression: Finding a solution

$$\mathbf{\Theta}^* = \arg \min_{\mathbf{\Theta} \in \mathbb{R}^2} \ell(\mathbf{\Theta})$$

where $\ell : \mathbb{R}^2 \to \mathbb{R}$ is defined as:

$$\ell(a,b) = \sum_{i=1}^{n} (y_i - a x_i - b)^2$$

A solution is found by setting $\nabla_{\mathbf{\Theta}} \ell(\mathbf{\Theta}) = \mathbf{0}$:

$$\begin{aligned}
\nabla_{\mathbf{\Theta}} \sum_{i=1}^{n} (y_i - a x_i - b)^2 &= \sum_{i=1}^{n} \nabla_{\mathbf{\Theta}} (y_i - a x_i - b)^2 \\
&= \sum_{i=1}^{n} \nabla_{\mathbf{\Theta}} (\cancel{y_i^2} + a^2 x_i^2 + b^2 - 2 a x_i y_i - 2 b y_i + 2 a b x_i) \\
&= \begin{pmatrix} \sum_{i=1}^{n} 2 a x_i^2 - 2 x_i y_i + 2 b x_i \\ \sum_{i=1}^{n} 2b - 2 y_i + 2 a x_i \end{pmatrix}
\end{aligned}$$

Linear regression: Finding a solution

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta} \in \mathbb{R}^2} \ell(\mathbf{\Theta})$$

where $\ell : \mathbb{R}^2 \to \mathbb{R}$ is defined as:

$$\ell(a, b) = \sum_{i=1}^{n} (y_i - a x_i - b)^2$$

A solution is found by setting $\nabla_{\mathbf{\Theta}} \ell(\mathbf{\Theta}) = \mathbf{0}$:

$$\nabla_{\mathbf{\Theta}} \sum_{i=1}^{n} (y_i - a x_i - b)^2 = \begin{pmatrix} \sum_{i=1}^{n} 2a x_i^2 - 2 x_i y_i + 2b x_i \\ \sum_{i=1}^{n} 2b - 2y_i + 2a x_i \end{pmatrix}$$

We get 2 linear equations in the 2 unknowns a, b:

$$\begin{pmatrix} \sum_{i=1}^{n} a x_i^2 + b x_i - x_i y_i \\ \sum_{i=1}^{n} a x_i + b - y_i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Linear regression: Matrix notation

The learning model of linear regression is linear in the parameters (while it is **not** linear in x, due to the bias).

Therefore, we can use matrix notation:

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{Y}} = \underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{\boldsymbol{\theta}}$$

Linear regression: Matrix notation

The learning model of linear regression is linear in the parameters (while it is **not** linear in x, due to the bias).

Therefore, we can use matrix notation:

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{Y}} = \underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{\boldsymbol{\theta}}$$

Remark: Deep learning frameworks frequently use the alternative expression with the bias encoded separately:

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{Y}} = a \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{\mathbf{X}} + b$$

Familiarize with matrix calculus.

When implementing deep nets, we manipulate matrices, vectors, and tensors.

Familiarize with matrix calculus.

When implementing deep nets, we manipulate matrices, vectors, and tensors.

(if you need it: brief lecture on matrix manipulation tricks?)

Linear regression: Matrix notation

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{\boldsymbol{\theta}}$$

This expresses all the equations $y_i = ax_i + b$ at once and makes the linearity w.r.t. a, b evident.

The MSE is simply:

$$\ell(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2$$

Linear regression: Matrix notation

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{\boldsymbol{\theta}}$$

This expresses all the equations $y_i = ax_i + b$ at once and makes the linearity w.r.t. a, b evident.

The MSE is simply:

$$\ell(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})$$

Linear regression: Matrix notation

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{\boldsymbol{\theta}}$$

This expresses all the equations $y_i = ax_i + b$ at once and makes the linearity w.r.t. a, b evident.

The MSE is simply:

$$\ell(\boldsymbol{\theta}) = \mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\theta}$$

Linear regression: Matrix notation

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{\boldsymbol{\theta}}$$

This expresses all the equations $y_i = ax_i + b$ at once and makes the linearity w.r.t. a, b evident.

The MSE is simply:

$$\ell(\boldsymbol{\theta}) = \mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\theta}$$

Setting the gradient w.r.t. $\boldsymbol{\theta}$ to zero:

$$-2\mathbf{X}^\top \mathbf{y} + 2\mathbf{X}^\top \mathbf{X}\boldsymbol{\theta} = \mathbf{0}$$

Linear regression: Matrix notation

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{\boldsymbol{\theta}}$$

This expresses all the equations $y_i = ax_i + b$ at once and makes the linearity w.r.t. a, b evident.

The MSE is simply:

$$\ell(\boldsymbol{\theta}) = \mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\theta}$$

Setting the gradient w.r.t. $\boldsymbol{\theta}$ to zero:

$$\mathbf{X}^\top \mathbf{X}\boldsymbol{\theta} = \mathbf{X}^\top \mathbf{y}$$

Linear regression: Matrix notation

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{\boldsymbol{\theta}}$$

This expresses all the equations $y_i = ax_i + b$ at once and makes the linearity w.r.t. a, b evident.

The MSE is simply:

$$\ell(\boldsymbol{\theta}) = \mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\theta}$$

Setting the gradient w.r.t. $\boldsymbol{\theta}$ to zero:

$$\boldsymbol{\theta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

We get a closed form solution to our problem.

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

$$\mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\theta} \quad \stackrel{\nabla_{\boldsymbol{\theta}}}{\Longrightarrow} \quad -2\mathbf{X}^\top \mathbf{y} + 2\mathbf{X}^\top \mathbf{X}\boldsymbol{\theta}$$

what we did is **exactly equivalent** to the element-by-element computation of slide $\#16$, but we did it directly in matrix form.

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

$$\mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\theta} \quad \overset{\nabla_{\boldsymbol{\theta}}}{\Longrightarrow} \quad -2\mathbf{X}^\top \mathbf{y} + 2\mathbf{X}^\top \mathbf{X}\boldsymbol{\theta}$$

what we did is **exactly equivalent** to the element-by-element computation of slide #16, but we did it directly in matrix form.

Example: $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{A}\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \begin{pmatrix} \theta_1 & \cdots & \theta_n \end{pmatrix} \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} \theta_1 \\ \cdots \\ \theta_n \end{pmatrix}$$

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

$$\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta} \quad \overset{\nabla_{\boldsymbol{\theta}}}{\Longrightarrow} \quad -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta}$$

what we did is **exactly equivalent** to the element-by-element computation of slide #16, but we did it directly in matrix form.

Example: $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top\mathbf{A}\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\theta_i\theta_j$$

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

$$\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta} \quad \overset{\nabla_{\boldsymbol{\theta}}}{\Longrightarrow} \quad -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta}$$

what we did is **exactly equivalent** to the element-by-element computation of slide #16, but we did it directly in matrix form.

Example: $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top\mathbf{A}\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial\theta_1}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\theta_i\theta_j \\ \vdots \\ \frac{\partial}{\partial\theta_n}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\theta_i\theta_j \end{pmatrix}$$

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

$$\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta} \quad \overset{\nabla_{\boldsymbol{\theta}}}{\Longrightarrow} \quad -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta}$$

what we did is **exactly equivalent** to the element-by-element computation of slide #16, but we did it directly in matrix form.

<u>Example:</u> $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top\mathbf{A}\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \begin{pmatrix} \sum_j a_{1j}\theta_j + \sum_i a_{i1}\theta_i \\ \vdots \\ \sum_j a_{nj}\theta_j + \sum_i a_{in}\theta_i \end{pmatrix}$$

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

$$\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta} \quad \overset{\nabla_{\boldsymbol{\theta}}}{\Longrightarrow} \quad -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta}$$

what we did is **exactly equivalent** to the element-by-element computation of slide #16, but we did it directly in matrix form.

Example: $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top\mathbf{A}\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \begin{pmatrix} \sum_i (a_{1i} + a_{i1})\theta_i \\ \vdots \\ \sum_i (a_{ni} + a_{in})\theta_i \end{pmatrix}$$

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

$$\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta} \quad \stackrel{\nabla_{\boldsymbol{\theta}}}{\Longrightarrow} \quad -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta}$$

what we did is **exactly equivalent** to the element-by-element computation of slide #16, but we did it directly in matrix form.

<u>Example:</u> $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top\mathbf{A}\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = (\mathbf{A} + \mathbf{A}^\top)\boldsymbol{\theta}$$

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

$$\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta} \quad \overset{\nabla_{\boldsymbol{\theta}}}{\Longrightarrow} \quad -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta}$$

what we did is **exactly equivalent** to the element-by-element computation of slide #16, but we did it directly in matrix form.

Example: $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top\mathbf{A}\boldsymbol{\theta}$

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = (\mathbf{A} + \mathbf{A}^\top)\boldsymbol{\theta}$$

If $\mathbf{A}$ is symmetric (e.g., $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$), then:

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = 2\mathbf{A}\boldsymbol{\theta}$$

Linear regression: Higher dimensions

In the general case, the data points $(\mathbf{x}_i, \mathbf{y}_i)$ are vectors in $\mathbf{R}^k$:

$$\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \mathbf{b} \quad \text{for } i = 1, \ldots, n$$

Linear regression: Higher dimensions

In the general case, the data points $(\mathbf{x}_i, \mathbf{y}_i)$ are vectors in $\mathbf{R}^k$:

$$\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \mathbf{b} \quad \text{for } i = 1, \ldots, n$$

Stacking all data points into matrices $\mathbf{X} = \begin{pmatrix} | & | & \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots \\ | & | & \end{pmatrix}$ and $\mathbf{Y}$, we get:

$$\underbrace{\begin{pmatrix} y_{11} & \cdots & y_{1d} \\ y_{21} & \cdots & y_{2d} \\ \vdots & & \vdots \\ y_{n1} & \cdots & y_{nd} \end{pmatrix}}_{\mathbf{Y}^\top} = \underbrace{\begin{pmatrix} x_{11} & \cdots & x_{1d} & 1 \\ x_{21} & \cdots & x_{2d} & 1 \\ \vdots & & \vdots & \vdots \\ x_{n1} & \cdots & x_{nd} & 1 \end{pmatrix}}_{\mathbf{X}^\top} \underbrace{\begin{pmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & & \vdots \\ a_{d1} & \cdots & a_{dd} \\ b_1 & \cdots & b_d \end{pmatrix}}_{\boldsymbol{\Theta}}$$

According to which, for each output data point $\mathbf{y}_i$ we have:

$$\underbrace{\begin{pmatrix} y_{i1} \\ \vdots \\ y_{id} \end{pmatrix}}_{\mathbf{y}_i} = \begin{pmatrix} \sum_{j=1}^{d} a_{j1} x_{ij} + b_1 \\ \vdots \\ \sum_{j=1}^{d} a_{jd} x_{ij} + b_d \end{pmatrix}$$

Linear regression: Higher dimensions

In the general case, the data points $(\mathbf{x}_i, \mathbf{y}_i)$ are vectors in $\mathbf{R}^k$:

$$\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \mathbf{b} \quad \text{for } i = 1, \ldots, n$$

Stacking all data points into matrices $\mathbf{X} = \begin{pmatrix} | & | & \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots \\ | & | & \end{pmatrix}$ and $\mathbf{Y}$, we get:

$$\underbrace{\begin{pmatrix} y_{11} & \cdots & y_{1d} \\ y_{21} & \cdots & y_{2d} \\ \vdots & & \vdots \\ y_{n1} & \cdots & y_{nd} \end{pmatrix}}_{\mathbf{Y}^\top} = \underbrace{\begin{pmatrix} x_{11} & \cdots & x_{1d} & 1 \\ x_{21} & \cdots & x_{2d} & 1 \\ \vdots & & \vdots & \vdots \\ x_{n1} & \cdots & x_{nd} & 1 \end{pmatrix}}_{\mathbf{X}^\top} \underbrace{\begin{pmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & & \vdots \\ a_{d1} & \cdots & a_{dd} \\ b_1 & \cdots & b_d \end{pmatrix}}_{\boldsymbol{\Theta}}$$

The MSE reads:

$$\ell(\boldsymbol{\Theta}) = \|\mathbf{Y} - \mathbf{X}\boldsymbol{\Theta}\|_2^2 = \mathrm{tr}(\mathbf{Y}^\top\mathbf{Y}) - 2\mathrm{tr}(\mathbf{Y}^\top\mathbf{X}\boldsymbol{\Theta}) + \mathrm{tr}(\boldsymbol{\Theta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\Theta})$$

Linear regression: Higher dimensions

In the general case, the data points $(\mathbf{x}_i, \mathbf{y}_i)$ are vectors in $\mathbf{R}^k$:

$$\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \mathbf{b} \quad \text{for } i = 1, \ldots, n$$

Stacking all data points into matrices $\mathbf{X} = \begin{pmatrix} | & | & \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots \\ | & | & \end{pmatrix}$ and $\mathbf{Y}$, we get:

$$\underbrace{\begin{pmatrix} y_{11} & \cdots & y_{1d} \\ y_{21} & \cdots & y_{2d} \\ \vdots & & \vdots \\ y_{n1} & \cdots & y_{nd} \end{pmatrix}}_{\mathbf{Y}^\top} = \underbrace{\begin{pmatrix} x_{11} & \cdots & x_{1d} & 1 \\ x_{21} & \cdots & x_{2d} & 1 \\ \vdots & & \vdots & \vdots \\ x_{n1} & \cdots & x_{nd} & 1 \end{pmatrix}}_{\mathbf{X}^\top} \underbrace{\begin{pmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & & \vdots \\ a_{d1} & \cdots & a_{dd} \\ b_1 & \cdots & b_d \end{pmatrix}}_{\boldsymbol{\Theta}}$$

The MSE reads:

$$\ell(\boldsymbol{\Theta}) = \|\mathbf{Y} - \mathbf{X}\boldsymbol{\Theta}\|_2^2 = \operatorname{tr}(\mathbf{Y}^\top\mathbf{Y}) - 2\operatorname{tr}(\mathbf{Y}^\top\mathbf{X}\boldsymbol{\Theta}) + \operatorname{tr}(\boldsymbol{\Theta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\Theta})$$

The closed form solution of $\nabla_{\boldsymbol{\Theta}}\ell(\boldsymbol{\Theta}) = \mathbf{0}$ is (extra points: show me why):

$$\boldsymbol{\Theta} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$$

Wrap-up

$$\mathbf{x}_i \rightarrow f_{\mathbf{\Theta}} \rightarrow \mathbf{y}_i \rightarrow \ell_{\Theta}(\mathbf{x}_i, \mathbf{y}_i)$$

Sometimes, the learning model is linear and the loss is quadratic.

This case can be solved in closed form.

Wrap-up

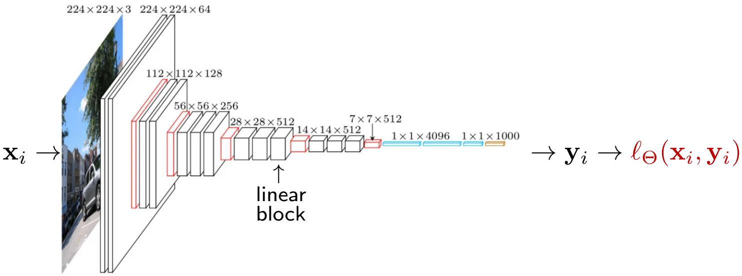
$$\mathbf{x}_i \rightarrow f_{\boldsymbol{\Theta}} \rightarrow \mathbf{y}_i \rightarrow \ell_{\Theta}(\mathbf{x}_i, \mathbf{y}_i)$$

Sometimes, the learning model is linear and the loss is quadratic.

This case can be solved in closed form.

The more data points $(\mathbf{x}_i, \mathbf{y}_i)$ we have, the better.

Wrap-up



Sometimes, the learning model is linear and the loss is quadratic.

This case can be solved in closed form.

The more data points $(\mathbf{x}_i, \mathbf{y}_i)$ we have, the better.

In deep learning, linear models usually appear as "pieces" within more complicated nonlinear models.

Suggested reading

For convexity and optimality, read Sections 3.1.1 and 3.1.3 of the book:

S. Boyd & L. Vandenberghe, "Convex optimization". Cambridge University Press, 2009

Public download link: `https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf`