Machine Learning

Regression problems

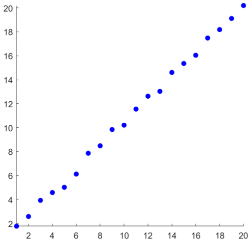
Emanuele Rodolà
rodola@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

2nd semester a.y. 2023/2024 · March 11, 2024

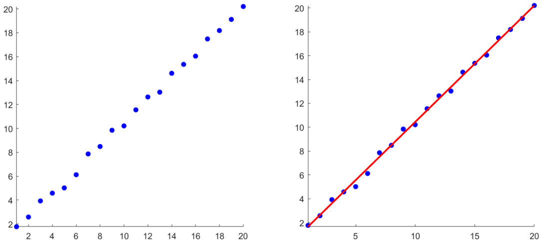
Linear regression

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



Linear regression

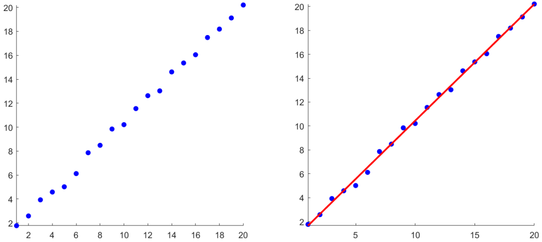
We start from the simplest non-trivial case for a learning model:



$$y_i = ax_i + b$$

Linear regression

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



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

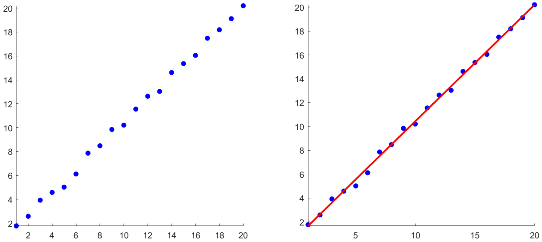
Model: linear + bias

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

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 that minimize 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 that minimize 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 $i = 1, \ldots, n$.

Problem: Choose a and b that minimize 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 that minimize 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 or energy 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 that minimize the mean squared error (MSE) between input and predicted output:

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

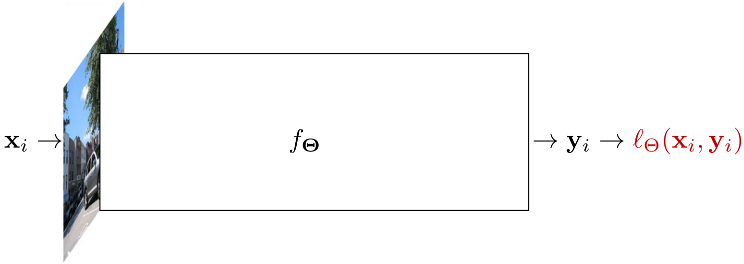
The error criterion w.r.t. the parameters is also called a loss or energy function, usually denoted by ℓ:

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

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

Linear regression

We are considering the following case:



where $f_{\boldsymbol{\Theta}}$ 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.

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.

Let's see what optimization problems we can solve easily!

Convexity and gradients

Convex functions

Jensen's inequality:

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

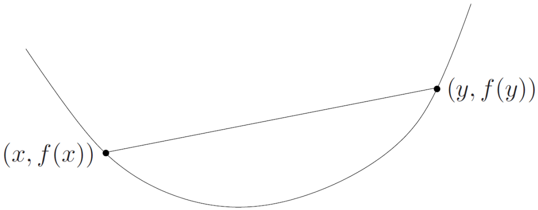
for all x, y and $\alpha \in (0, 1)$

Convex functions

Jensen's inequality:

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

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

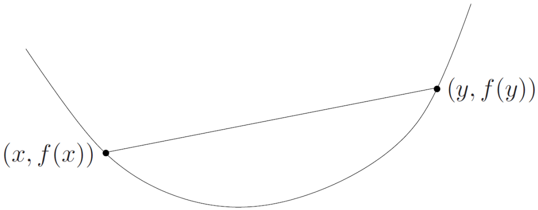


Convex functions

Jensen's inequality:

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

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



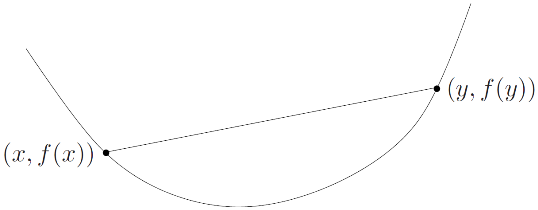
Let us further assume that f is a differentiable function, so that we can compute its derivative $\frac{df}{dx}$ at all points x.

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.

Theorem: the global minimizer x is where $\frac{df(x)}{dx} = 0$.

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

In ML we typically deal with energies over $n \gg 1$ parameters:

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

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

In ML we typically deal with energies 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 ML we typically deal with energies 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 ML we typically deal with energies over $n \gg 1$ parameters:

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

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:

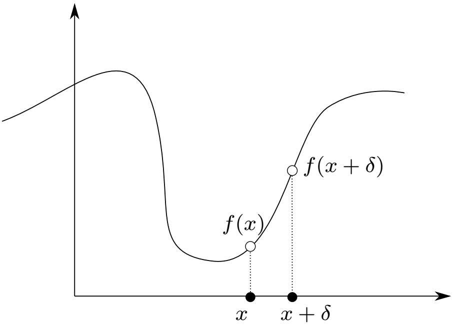
$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \mathbf{0} \quad \Longrightarrow \quad f(\mathbf{x}) \leq f(\mathbf{y}) \text{ for all } \mathbf{y} \in \mathbb{R}^n$$

The gradient

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

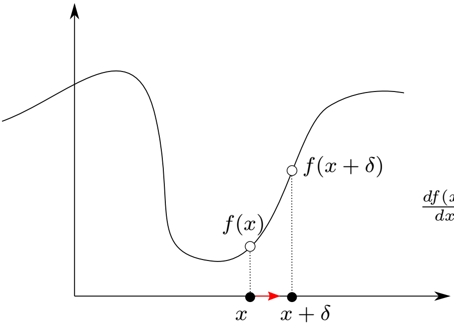
The gradient

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

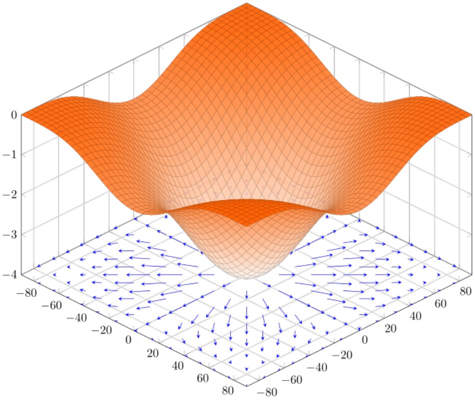
The gradient $\nabla_{\mathbf{x}} f(\mathbf{x})$ encodes the direction of steepest ascent of f at point $\mathbf{x}$. In the simple 1D case:



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

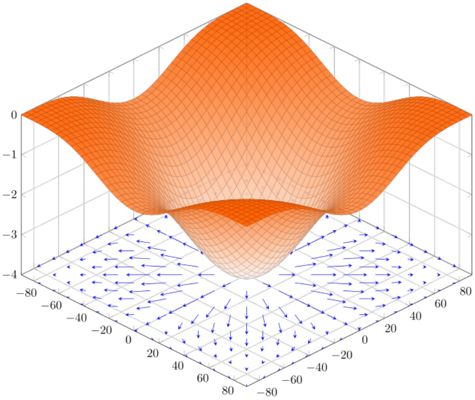
The gradient

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 steepness.

Convex functions: Global minima

To summarize:

If $f(x)$ is convex, then a global minimizer is found by setting $\frac{df(x)}{dx} = 0$ and solving for x.

Convex functions: Global minima

To summarize:

If $f(x)$ is convex, then a global minimizer is found by setting $\frac{df(x)}{dx} = 0$ and solving for x.

The above only makes sense if $f : \mathbb{R} \to \mathbb{R}$.

Convex functions: Global minima

To summarize:

If $f(x)$ is convex, then a global minimizer is found by setting $\frac{df(x)}{dx} = 0$ and solving for x.

The above only makes sense if $f : \mathbb{R} \to \mathbb{R}$.

If $f(\mathbf{x})$ is multi-dimensional, i.e. $f : \mathbb{R}^n \to \mathbb{R}$, do we have a notion of convexity and derivative?

Convex functions: Global minima

To summarize:

If $f(x)$ is convex, then a global minimizer is found by setting $\frac{df(x)}{dx} = 0$ and solving for x.

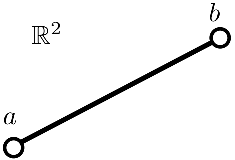
The above only makes sense if $f : \mathbb{R} \to \mathbb{R}$.

If $f(\mathbf{x})$ is multi-dimensional, i.e. $f : \mathbb{R}^n \to \mathbb{R}$, do we have a notion of convexity and derivative?

If yes, can we find global minimizers as easily as in the former case?

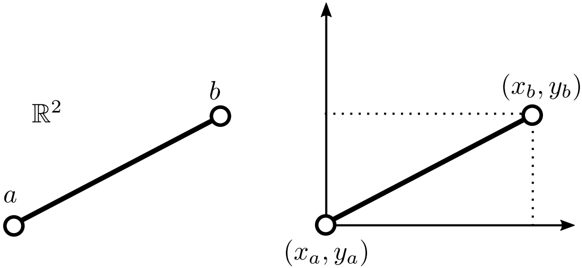
Vector lengths

How to measure the length of the gradient? Let's first start from the definition of Euclidean distance, which measures the length of any straight line connecting two points:



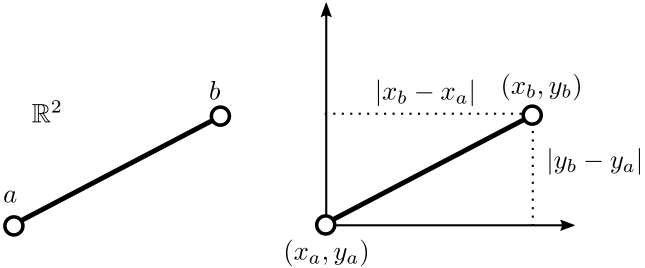
Vector lengths

How to measure the length of the gradient? Let's first start from the definition of Euclidean distance, which measures the length of any straight line connecting two points:



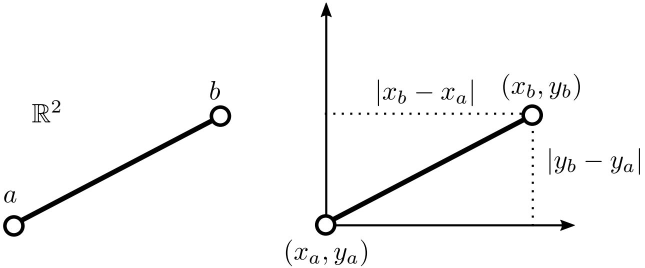
Vector lengths

How to measure the length of the gradient? Let's first start from the definition of Euclidean distance, which measures the length of any straight line connecting two points:



Vector lengths

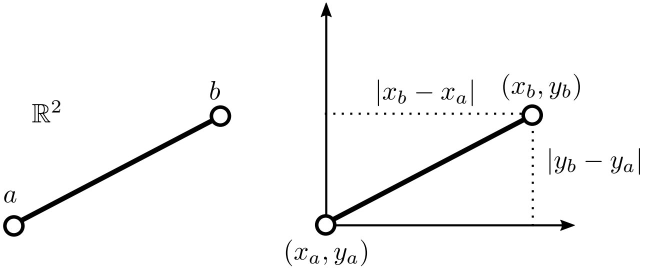
How to measure the length of the gradient? Let's first start from the definition of Euclidean distance, which measures the length of any straight line connecting two points:



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

Vector lengths

How to measure the length of the gradient? Let's first start from the definition of Euclidean distance, which measures the length of any 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 $p \geq 1$:

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

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 $\mathbb{R}^k$:

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

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 $\mathbb{R}^k$:

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

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:

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

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 $\mathbb{R}^k$:

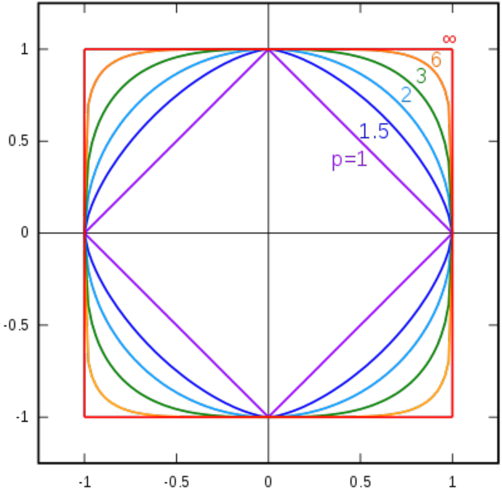
$$\|\mathbf{x} - \mathbf{y}\|_p = (\textstyle\sum_{i=1}^k |x_i - y_i|^p)^{\frac{1}{p}}$$

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:

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

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



Unit balls on manifolds

The notion of unit ball makes sense in any metric space, as it only depends on the presence of a distance function.

anisotropic

geodesic

L_7 in $\mathbb{R}^3$

Each isoline identifies points at the same distance from the source

Normal equation

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:

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

Linear regression: Finding a solution

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

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 $\nabla_{\boldsymbol{\Theta}} \ell(\boldsymbol{\Theta}) = \mathbf{0}$:

$$\nabla_{\boldsymbol{\Theta}} \sum_{i=1}^{n} (y_i - ax_i - b)^2 = \sum_{i=1}^{n} \nabla_{\boldsymbol{\Theta}} (y_i - ax_i - b)^2$$

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:

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

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 $\nabla_{\mathbf{\Theta}} \ell(\mathbf{\Theta}) = \mathbf{0}$:

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

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:

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

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

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

Linear regression: Finding a solution

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

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 $\nabla_{\boldsymbol{\Theta}} \ell(\boldsymbol{\Theta}) = \mathbf{0}$:

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

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, in matrix notation the equations $y_i = ax_i + b$ read:

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

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, in matrix notation the equations $y_i = ax_i + b$ read:

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

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 a ML system, we manipulate matrices, vectors, and tensors.

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 $\nabla_{\boldsymbol{\theta}} \ell = \mathbf{0}$ we get:

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

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 $\nabla_{\boldsymbol{\theta}} \ell = \mathbf{0}$ we get:

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

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 $\nabla_{\boldsymbol{\theta}} \ell = \mathbf{0}$ we get:

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

We get a closed form solution to our problem (aka normal equation).

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

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

what we did is **exactly equivalent** to the element-by-element computation of slide #15/20, but we did it directly in matrix form.

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

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

what we did is **exactly equivalent** to the element-by-element computation of slide #15/20, 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:

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

what we did is **exactly equivalent** to the element-by-element computation of slide #15/20, but we did it directly in matrix form.

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

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

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

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

what we did is **exactly equivalent** to the element-by-element computation of slide #15/20, 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:

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

what we did is **exactly equivalent** to the element-by-element computation of slide #15/20, but we did it directly in matrix form.

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

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

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

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

what we did is **exactly equivalent** to the element-by-element computation of slide #15/20, 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:

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

what we did is **exactly equivalent** to the element-by-element computation of slide #15/20, but we did it directly in matrix form.

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

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

A note on the gradient in matrix form

In the previous slide, for the differentiation step:

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

what we did is **exactly equivalent** to the element-by-element computation of slide #15/20, 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

Until now we have seen the case where:

$$y_i = ax_i + b \quad \text{for } i = 1, \ldots, n$$

that is, each data point is one-dimensional (just one number).

Linear regression: Higher dimensions

Until now we have seen the case where:

$$y_i = ax_i + b \quad \text{for } i = 1, \ldots, n$$

that is, each data point is one-dimensional (just one number).

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

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

Linear regression: Higher dimensions

Until now we have seen the case where:

$$y_i = ax_i + b \quad \text{for } i = 1, \ldots, n$$

that is, each data point is one-dimensional (just one number).

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

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

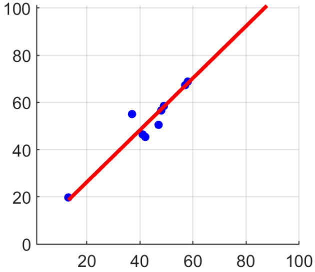
Defining the matrices $\mathbf{X} = \begin{pmatrix} | & | & \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots \\ | & | & \\ 1 & 1 & \end{pmatrix}, \mathbf{Y} = \begin{pmatrix} | & | & \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots \\ | & | & \end{pmatrix}, \mathbf{\Theta} = \begin{pmatrix} \mathbf{A} \\ \mathbf{b}^\top \end{pmatrix}$,

we get a closed-form solution to $\nabla_{\mathbf{\Theta}} \ell(\mathbf{\Theta}) = \mathbf{0}$:

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

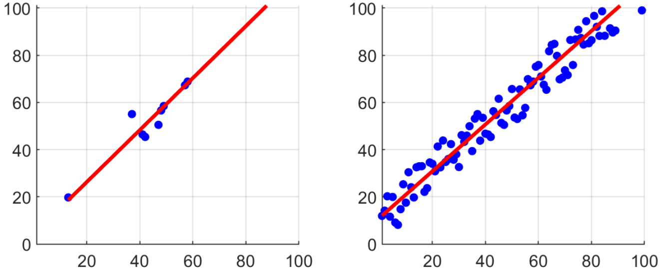
Polynomial regression

Data distribution



Assumption: linear model

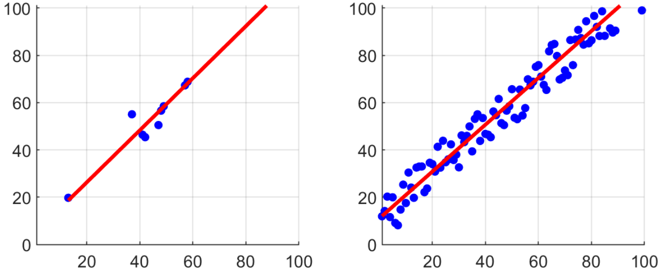
Data distribution



Assumption: linear model

More data allows us to improve our prediction

Data distribution

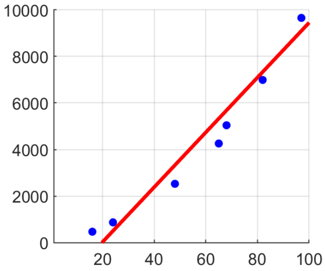


Assumption: linear model

More data allows us to improve our prediction

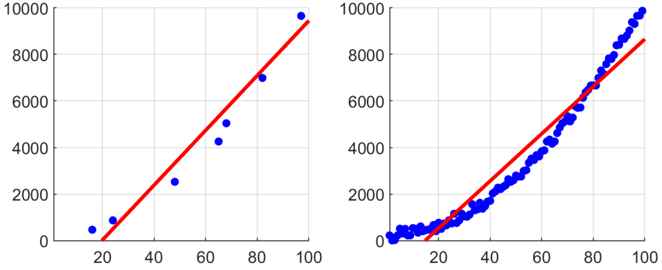
What if the assumption (i.e. linear prior here) is **wrong**?

Data distribution



Assumption: linear model

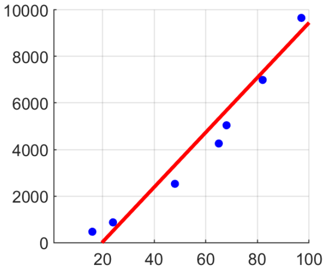
Data distribution

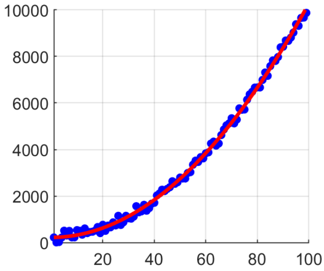


Assumption: linear model

More data confutes our assumptions

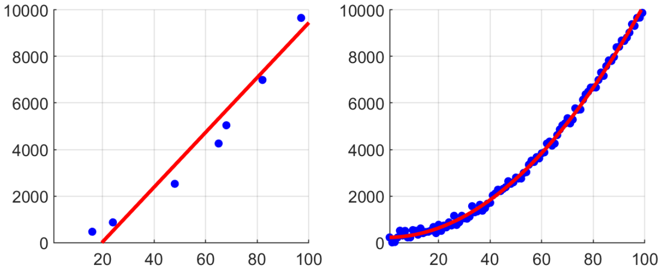
Data distribution





Assumption: quadratic model

Data distribution

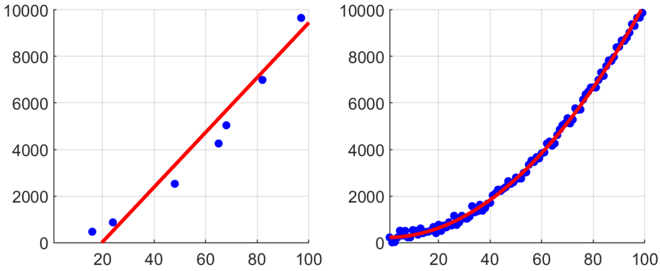


Assumption: quadratic model

Key questions:

- How to select the correct distribution?

Data distribution

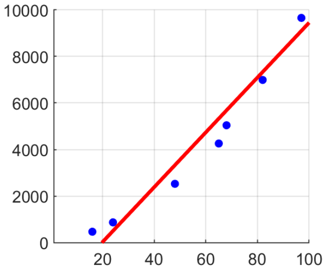


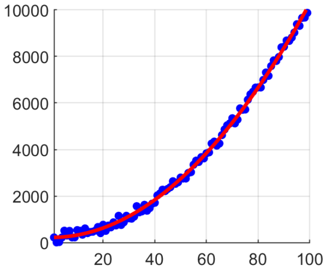
Assumption: quadratic model

Key questions:

- How to select the correct distribution?
- How much data do we need?

Data distribution





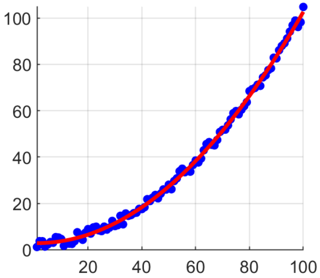
Assumption: quadratic model

Key questions:

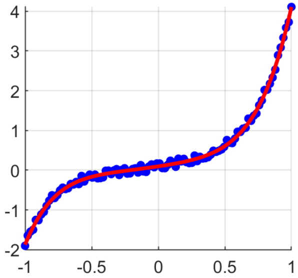
- How to select the correct distribution?
- How much data do we need?
- What if the correct distribution does not admit a simple expression?

Polynomial regression

After the linear model, the simplest thing is a polynomial model.



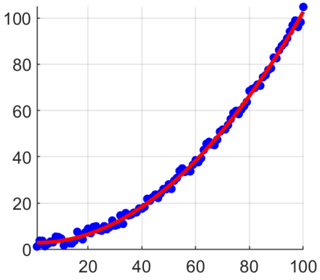
2nd-order



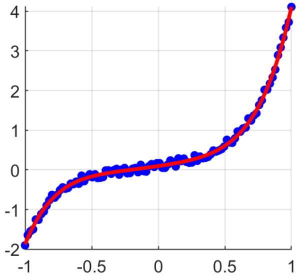
5th-order

Polynomial regression

After the linear model, the simplest thing is a polynomial model.



2nd-order

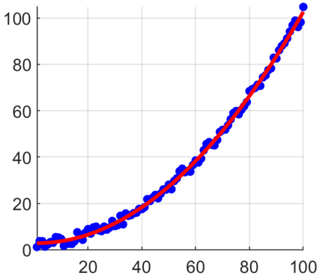


5th-order

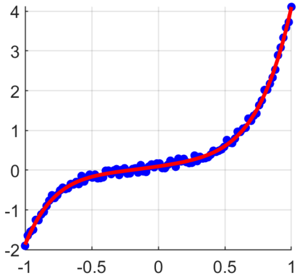
The number of parameters grows with the order.

Polynomial regression

After the linear model, the simplest thing is a polynomial model.



2nd-order



5th-order

The number of parameters grows with the order.

More data are needed to make an informed decision on the order.

Polynomial regression

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

Linear regression with polynomial features

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

Linear regression with polynomial features

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

$$y_i = a_3 x_i^3 + a_2 x_i^2 + a_1 x_i + b \quad \text{for all data points } i = 1, \ldots, n$$

Linear regression with polynomial features

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

$$y_i = b + \sum_{j=1}^{k} a_j x_i^j \quad \text{for all data points } i = 1, \ldots, n$$

Linear regression with polynomial features

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

$$y_i = b + \sum_{j=1}^{k} a_j x_i^j \quad \text{for all data points } i = 1, \ldots, n$$

Linear regression with polynomial features

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

$$y_i = b + \sum_{j=1}^{k} a_j x_i^j \quad \text{for all data points } i = 1, \dots, n$$

In matrix notation:

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} x_1^k & x_1^{k-1} & \cdots & x_1 & 1 \\ x_2^k & x_2^{k-1} & \cdots & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^k & x_n^{k-1} & \cdots & x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a_k \\ a_{k-1} \\ \vdots \\ a_1 \\ b \end{pmatrix}}_{\boldsymbol{\theta}}$$

Linear regression with polynomial features

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

$$y_i = b + \sum_{j=1}^{k} a_j x_i^j \quad \text{for all data points } i = 1, \ldots, n$$

In matrix notation:

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} x_1^k & x_1^{k-1} & \cdots & x_1 & 1 \\ x_2^k & x_2^{k-1} & \cdots & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^k & x_n^{k-1} & \cdots & x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a_k \\ a_{k-1} \\ \vdots \\ a_1 \\ b \end{pmatrix}}_{\boldsymbol{\theta}}$$

The same exact least-squares solution as with linear regression applies, with the requirement that $k < n$.

Underfitting and overfitting

Polynomial fitting

An application of the Stone-Weierstrass theorem tells us:

If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x.

Polynomial fitting

An application of the Stone-Weierstrass theorem tells us:

If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x.

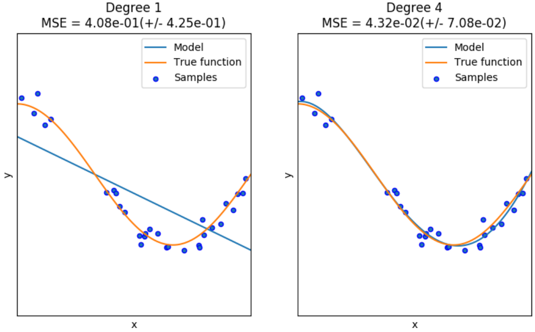
Thus, we can try to fit a polynomial in many cases.

Polynomial fitting

An application of the Stone-Weierstrass theorem tells us:

If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x.

Thus, we can try to fit a polynomial in many cases.

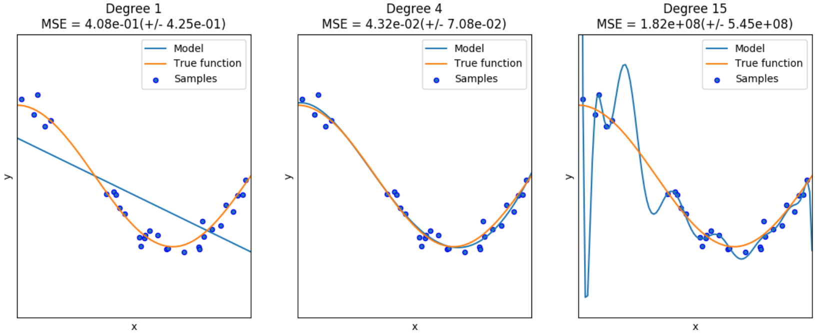


Polynomial fitting

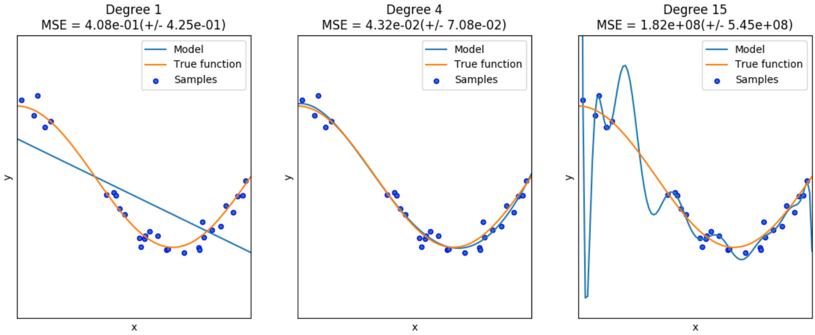
An application of the Stone-Weierstrass theorem tells us:

If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x.

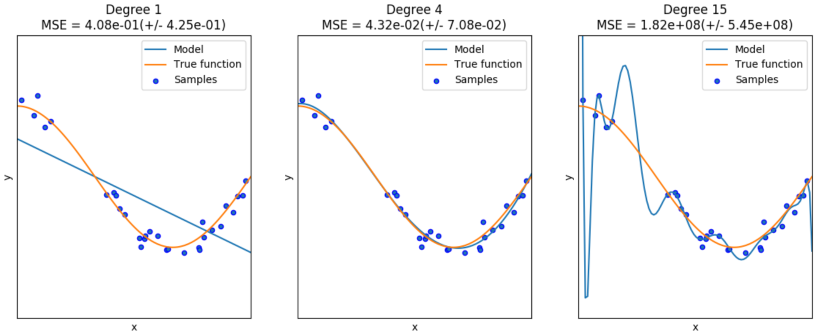
Thus, we can try to fit a polynomial in many cases.



Underfitting vs. Overfitting

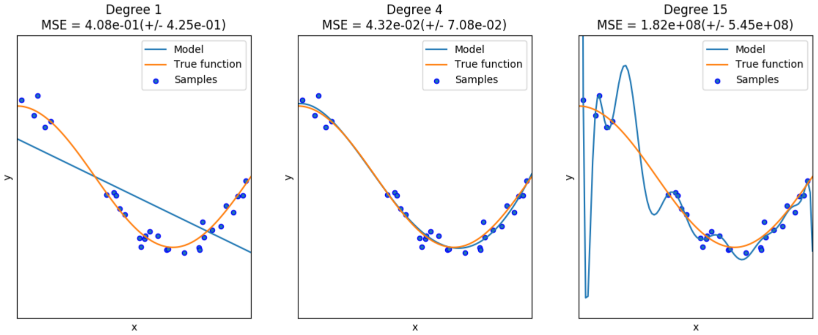


Underfitting vs. Overfitting



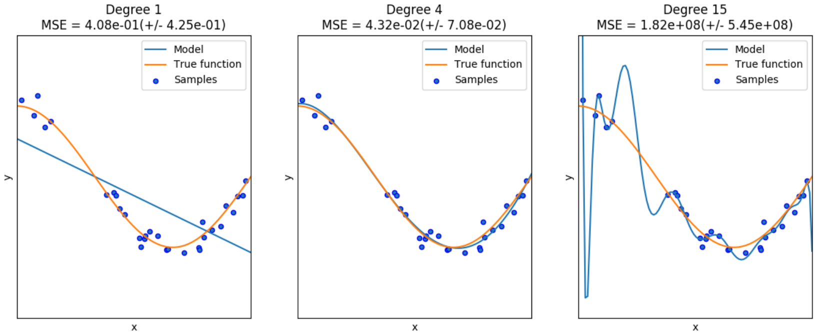
- Underfitting: Not sufficiently fitting the data (large MSE).

Underfitting vs. Overfitting



- Underfitting: Not sufficiently fitting the data (large MSE).
- Overfitting: We are "learning the noise" (small MSE).

Underfitting vs. Overfitting



- Underfitting: Not sufficiently fitting the data (large MSE).
- Overfitting: We are "learning the noise" (small MSE).

Adding complexity can lead to overfitting and thus worse generalization.

Underfitting vs. Overfitting

This trade-off is always present, and still an open problem.

Different mechanisms defend us from under- and overfitting.

Underfitting vs. Overfitting

This trade-off is always present, and still an open problem.

Different mechanisms defend us from under- and overfitting.

Detection is relatively easier:

1. Estimate the model parameters on a training set.
   (the MSE is minimized on example data)

Underfitting vs. Overfitting

This trade-off is always present, and still an open problem.

Different mechanisms defend us from under- and overfitting.

Detection is relatively easier:

1. Estimate the model parameters on a training set.
   (the MSE is minimized on example data)
2. Large MSE on the training $\Rightarrow$ underfitting

Underfitting vs. Overfitting

This trade-off is always present, and still an open problem.

Different mechanisms defend us from under- and overfitting.

Detection is relatively easier:

1. Estimate the model parameters on a training set.
   (the MSE is minimized on example data)
2. Large MSE on the training $\Rightarrow$ underfitting
3. Small MSE on the training $\Rightarrow$
   Apply the model parameters to a validation set.
   (the MSE is computed on different example data)

Underfitting vs. Overfitting

This trade-off is always present, and still an open problem.

Different mechanisms defend us from under- and overfitting.

Detection is relatively easier:

1. Estimate the model parameters on a training set.
   (the MSE is minimized on example data)
2. Large MSE on the training $\Rightarrow$ underfitting
3. Small MSE on the training $\Rightarrow$
   Apply the model parameters to a validation set.
   (the MSE is computed on different example data)
4. Large MSE on the validation $\Rightarrow$ overfitting

Underfitting vs. Overfitting

This trade-off is always present, and still an open problem.

Different mechanisms defend us from under- and overfitting.

Detection is relatively easier:

1. Estimate the model parameters on a training set.
   (the MSE is minimized on example data)
2. Large MSE on the training $\Rightarrow$ underfitting
3. Small MSE on the training $\Rightarrow$
   Apply the model parameters to a validation set.
   (the MSE is computed on different example data)
4. Large MSE on the validation $\Rightarrow$ overfitting $\Rightarrow$ bad generalization

Underfitting:
large training error, large validation error

Underfitting vs. Overfitting

Underfitting:
large training error, large validation error

Overfitting:
(very) small training error, large validation error

n-fold cross-validation

We do not want to overfit on the validation set either!

n-fold cross-validation

We do not want to overfit on the validation set either!

- Split the training set into n subsets (folds)

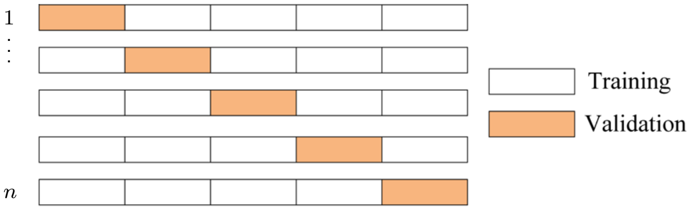
n-fold cross-validation

We do not want to overfit on the validation set either!

- Split the training set into n subsets (folds)
- Train on $n-1$ subsets and validate on the remaining 1

n-fold cross-validation

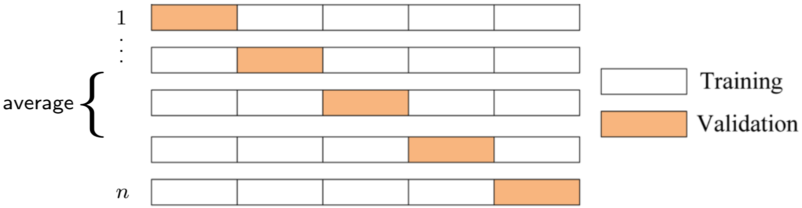
We do not want to overfit on the validation set either!



- Split the training set into n subsets (folds)
- Train on $n-1$ subsets and validate on the remaining 1
- Do this n times

n-fold cross-validation

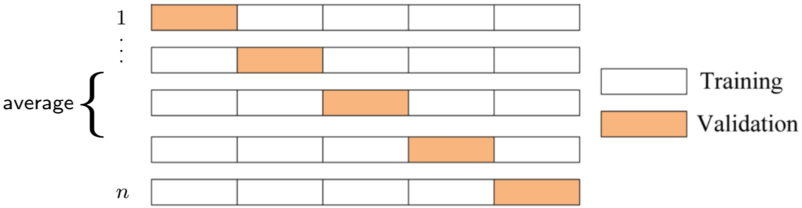
We do not want to overfit on the validation set either!



- Split the training set into n subsets (folds)
- Train on $n-1$ subsets and validate on the remaining 1
- Do this n times
- Average the score over the n folds

n-fold cross-validation

We do not want to overfit on the validation set either!



- Split the training set into n subsets (folds)
- Train on $n-1$ subsets and validate on the remaining 1
- Do this n times
- Average the score over the n folds

Example: For polynomial regression, do the above many times with different degrees, choose the run with the smallest average MSE.

Not done yet

"If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x."

So is polynomial regression all we need?

Not done yet

"If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x."

So is polynomial regression all we need?

Not really!

- Different loss than MSE
- Regularization
- Additional priors
- Intermediate features
- Flexibility
- Regression (predict a value) vs. classification (predict a category)

Regularization penalties

Sometimes our prior knowledge can be expressed in terms of an energy.

For example, avoid large parameters to counteract overfitting:

$$\min_{\mathbf{\Theta}} \underbrace{\ell_{\mathbf{\Theta}}}_{\text{data term}} + \underbrace{\lambda}_{\text{trade-off}} \cdot \underbrace{\|\Theta\|_F^2}_{\text{regularizer}}$$

Regularization penalties

Sometimes our prior knowledge can be expressed in terms of an energy.

For example, avoid large parameters to counteract overfitting:

$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}}}_{\text{data term}} + \underbrace{\lambda}_{\text{trade-off}} \cdot \underbrace{\|\Theta\|_F^2}_{\text{regularizer}}$$

Adding a quadratic penalty to the loss is also known as weight decay, ridge, or Tikhonov regularization.

Regularization penalties

Sometimes our prior knowledge can be expressed in terms of an energy.

For example, avoid large parameters to counteract overfitting:

$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}}}_{\text{data term}} + \underbrace{\lambda}_{\text{trade-off}} \cdot \underbrace{\|\Theta\|_F^2}_{\text{regularizer}}$$

Adding a quadratic penalty to the loss is also known as weight decay, ridge, or Tikhonov regularization.

More in general:

$$\min_{\boldsymbol{\Theta}} \ell_{\boldsymbol{\Theta}} + \lambda \|\Theta\|_p$$

Controlling parameter growth is generally known as shrinkage.

Regularization penalties

Sometimes our prior knowledge can be expressed in terms of an energy.

For example, avoid large parameters to counteract overfitting:

$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}}}_{\text{data term}} + \underbrace{\lambda}_{\text{trade-off}} \cdot \underbrace{\|\Theta\|_F^2}_{\text{regularizer}}$$

Adding a quadratic penalty to the loss is also known as weight decay, ridge, or Tikhonov regularization.

More in general:

$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}}}_{\text{convex}} + \lambda \underbrace{\|\Theta\|_p}_{\text{convex}}$$

Controlling parameter growth is generally known as shrinkage.

Regularization penalties

Sometimes our prior knowledge can be expressed in terms of an energy.

For example, avoid large parameters to counteract overfitting:

$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}}}_{\text{data term}} + \underbrace{\lambda}_{\text{trade-off}} \cdot \underbrace{\|\Theta\|_F^2}_{\text{regularizer}}$$

Adding a quadratic penalty to the loss is also known as weight decay, ridge, or Tikhonov regularization.

More in general:

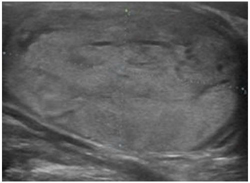
$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}} + \lambda\|\Theta\|_p}_{\text{convex}}$$

Controlling parameter growth is generally known as shrinkage.

Logistic regression

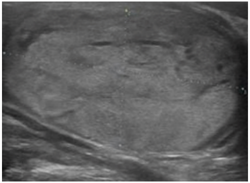
Classification

What if we want to predict a category instead of a value?

$$f_\Theta($$  $$) = \{0, 1\}$$

Classification

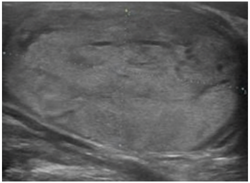
What if we want to predict a category instead of a value?

$f_\Theta($  $) = \{0, 1\}$

Possible solution: Do post-processing (e.g. thresholding) to convert linear regression to a binary output.

Classification

What if we want to predict a category instead of a value?

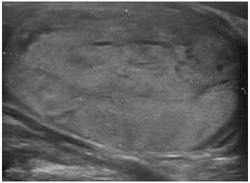
$f_\Theta($  $) = \{0, 1\}$

Possible solution: Do post-processing (e.g. thresholding) to convert linear regression to a binary output.

$\Rightarrow$ The solution is not necessarily an optimum anymore.

Classification

What if we want to predict a category instead of a value?

$f_\Theta($  $) = \{0, 1\}$

Possible solution: Do post-processing (e.g. thresholding) to convert linear regression to a binary output.

$\Rightarrow$ The solution is not necessarily an optimum anymore.

Instead: Modify the loss to minimize over categorical values directly.

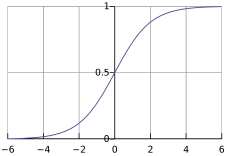
Logistic regression

New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} (y_i - \sigma(\underbrace{ax_i + b}_{\text{linear}}))^2$$

Here, σ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



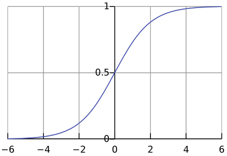
Logistic regression

New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} (y_i - \sigma(\underbrace{ax_i + b}_{\text{linear}}))^2$$

Here, σ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



σ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

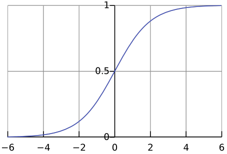
Logistic regression

New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} (y_i - \sigma(\underbrace{ax_i + b}_{\text{linear}}))^2 \qquad \text{non-convex in } a, b$$

Here, σ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



σ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

Logistic regression

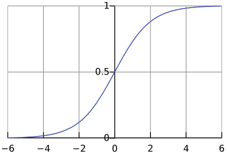
New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} c(x_i, y_i)\,, \quad \text{with}$$

$$c(x_i, y_i) = \begin{cases} -\ln(\sigma(ax_i + b)) & y_i = 1 \\ -\ln(1 - \sigma(ax_i + b)) & y_i = 0 \end{cases} \qquad \text{convex}$$

Here, σ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



σ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

Logistic regression

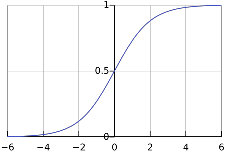
New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} c(x_i, y_i)\,, \quad \text{with}$$

$$c(x_i, y_i) = -y_i \ln(\sigma(ax_i + b)) - (1 - y_i)\ln(1 - \sigma(ax_i + b)) \quad \text{convex}$$

Here, σ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



σ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

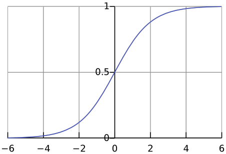
Logistic regression

New convex loss:

$$\ell_\Theta(\{x_i, y_i\}) = -\sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b))$$

Here, σ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



σ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_{\Theta} \ell_{\Theta} = 0$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$\nabla_\Theta \left(y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) \right)$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$\nabla_\Theta y_i \ln(\sigma(ax_i + b)) + \nabla_\Theta (1 - y_i) \ln(1 - \sigma(ax_i + b))$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \ln(\sigma(ax_i + b)) + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial a}$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial}{\partial a} a x_i + b$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial \sigma(ax_i + b)}{\partial(ax_i + b)} \cdot x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial}{\partial (ax_i + b)} \frac{1}{1 + e^{-(ax_i + b)}} \cdot x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{e^{-(ax_i + b)}}{(1 + e^{-(ax_i + b)})^2} \cdot x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{1}{1 + e^{-(ax_i + b)}} \frac{e^{-(ax_i + b)}}{1 + e^{-(ax_i + b)}} \cdot x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{1}{1 + e^{-(ax_i + b)}} \frac{(1 + e^{-(ax_i + b)}) - 1}{1 + e^{-(ax_i + b)}} \cdot x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{1}{1 + e^{-(ax_i + b)}} (1 - \frac{1}{1 + e^{-(ax_i + b)}}) \cdot x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \sigma(ax_i + b)(1 - \sigma(ax_i + b)) \cdot x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \sigma(ax_i + b)(1 - \sigma(ax_i + b)) \cdot x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial \ln(\sigma(ax_i + b))}{\partial \sigma(ax_i + b)} \cdot \sigma(ax_i + b)(1 - \sigma(ax_i + b)) \cdot x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{1}{\sigma(ax_i + b)} \cdot \sigma(ax_i + b)(1 - \sigma(ax_i + b)) \cdot x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{1}{\sigma(ax_i + b)} \cdot \sigma(ax_i + b)(1 - \sigma(ax_i + b)) \cdot x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(ax_i + b)) = (1 - \sigma(ax_i + b))x_i$$

Logistic regression: Finding a solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_{\Theta} \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

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

Consider the gradient of each term in the summation:

$$y_i \nabla_{\Theta} \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_{\Theta} \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(ax_i + b)) = (1 - \sigma(ax_i + b)) x_i$$

...and so on for the second term and for parameter b.

Logistic regression: Finding a solution

By looking at the partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(a x_i + b)) = (1 - \sigma(a x_i + b)) x_i$$

we see that the parameters enter the gradient in a nonlinear way.

Logistic regression: Finding a solution

By looking at the partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(a x_i + b)) = (1 - \sigma(a x_i + b)) x_i$$

we see that the parameters enter the gradient in a nonlinear way.

Thus:

- $\nabla \ell_\Theta = 0$ is not a linear system that we can solve easily.

Logistic regression: Finding a solution

By looking at the partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(ax_i + b)) = (1 - \sigma(ax_i + b))x_i$$

we see that the parameters enter the gradient in a nonlinear way.

Thus:

- $\nabla \ell_\Theta = 0$ is not a linear system that we can solve easily.
- $\nabla \ell_\Theta = 0$ is a transcendental equation $\Rightarrow$ no analytical solution.

Logistic regression: Finding a solution

By looking at the partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(ax_i + b)) = (1 - \sigma(ax_i + b))x_i$$

we see that the parameters enter the gradient in a nonlinear way.

Thus:

- $\nabla \ell_\Theta = 0$ is not a linear system that we can solve easily.
- $\nabla \ell_\Theta = 0$ is a transcendental equation $\Rightarrow$ no analytical solution.

model	**loss**	**solution**
linear regression		
linear regression + Tikhonov		
logistic regression		

Logistic regression: Finding a solution

By looking at the partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(a x_i + b)) = (1 - \sigma(a x_i + b)) x_i$$

we see that the parameters enter the gradient in a nonlinear way.

Thus:

- $\nabla \ell_\Theta = 0$ is not a linear system that we can solve easily.
- $\nabla \ell_\Theta = 0$ is a transcendental equation $\Rightarrow$ no analytical solution.

model	**loss**	**solution**
linear regression	convex	
linear regression + Tikhonov	convex	
logistic regression	convex	

Logistic regression: Finding a solution

By looking at the partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(ax_i + b)) = (1 - \sigma(ax_i + b))x_i$$

we see that the parameters enter the gradient in a nonlinear way.

Thus:

- $\nabla \ell_\Theta = 0$ is not a linear system that we can solve easily.
- $\nabla \ell_\Theta = 0$ is a transcendental equation $\Rightarrow$ no analytical solution.

model	**loss**	**solution**
linear regression	convex	least squares
linear regression + Tikhonov	convex	
logistic regression	convex	

Logistic regression: Finding a solution

By looking at the partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(a x_i + b)) = (1 - \sigma(a x_i + b)) x_i$$

we see that the parameters enter the gradient in a nonlinear way.

Thus:

- $\nabla \ell_\Theta = 0$ is not a linear system that we can solve easily.
- $\nabla \ell_\Theta = 0$ is a transcendental equation $\Rightarrow$ no analytical solution.

model	**loss**	**solution**
linear regression	convex	least squares
linear regression + Tikhonov	convex	least squares
logistic regression	convex	

Logistic regression: Finding a solution

By looking at the partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(a x_i + b)) = (1 - \sigma(a x_i + b)) x_i$$

we see that the parameters enter the gradient in a nonlinear way.

Thus:

- $\nabla \ell_\Theta = 0$ is not a linear system that we can solve easily.
- $\nabla \ell_\Theta = 0$ is a transcendental equation $\Rightarrow$ no analytical solution.

model	**loss**	**solution**
linear regression	convex	least squares
linear regression + Tikhonov	convex	least squares
logistic regression	convex	nonlinear optimization

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`

On polynomial regression vs. neural nets:
`https://ar5iv.org/abs/1806.06850`

Proof that the logistic loss is convex:
`https://math.stackexchange.com/questions/1582452/logistic-regression-prove-that-the-cost-function-is-convex`