

Deep Learning & Applied AI


Overfitting and going nonlinear

Emanuele Rodolà
rodola@di.uniroma1.it



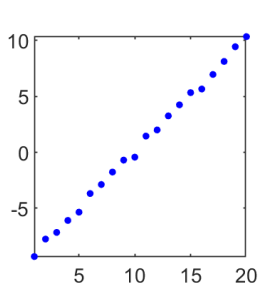
A glimpse into neural networks

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

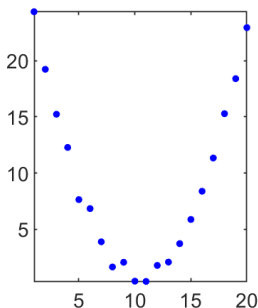

$$f_{\Theta}(\mathbf{x}) = \mathbf{y}$$

Parametrized models

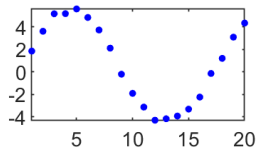
The parameters describe the behavior of the network, and must be **solved for**.



$$y = ax + b$$



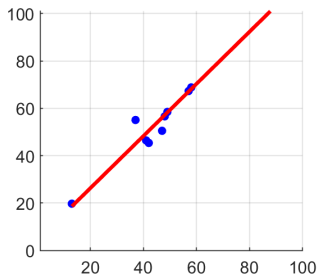
$$y = ax^2 + bx + c$$



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

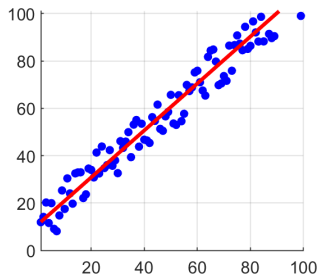
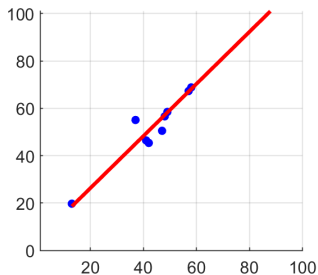
From a technical standpoint, our task is to determine the parameters Θ .

Data distribution



Assumption: **linear** model

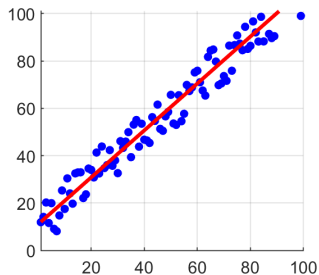
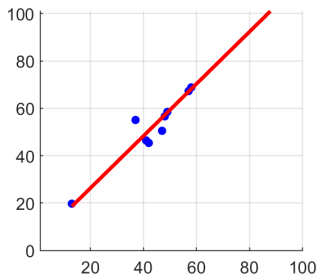
Data distribution



Assumption: **linear** model

More data allows us to improve our prediction

Data distribution

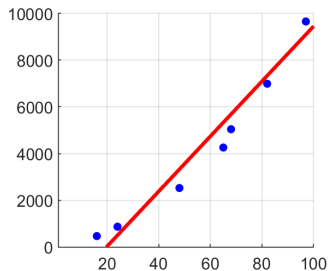


Assumption: **linear** model

More data allows us to improve our prediction

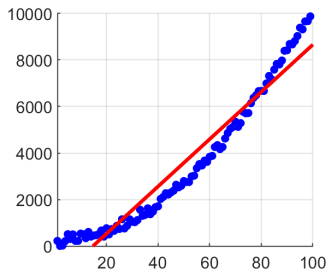
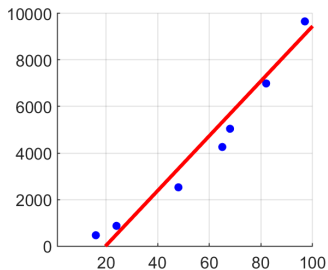
What if the assumption (i.e. linear prior here) is **wrong**?

Data distribution



Assumption: linear model

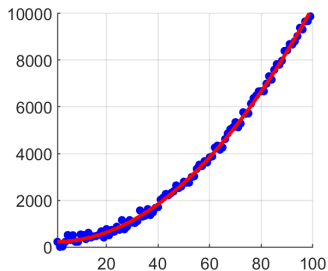
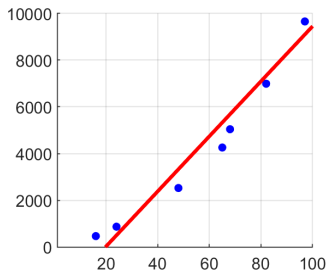
Data distribution



Assumption: **linear** model

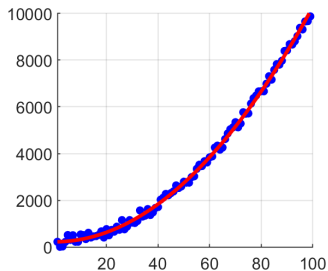
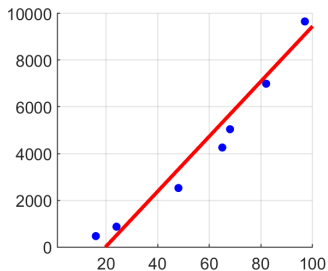
More data **confutes** our assumptions

Data distribution



Assumption: quadratic model

Data distribution

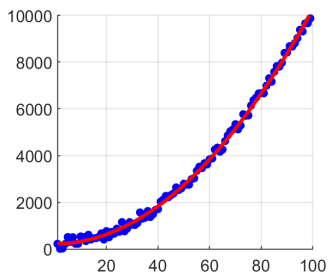
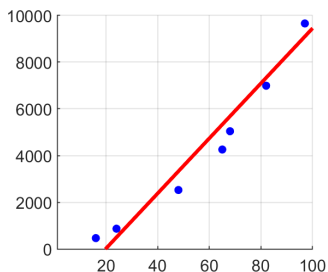


Assumption: **quadratic** model

Key questions:

- How to select the **correct distribution**?

Data distribution

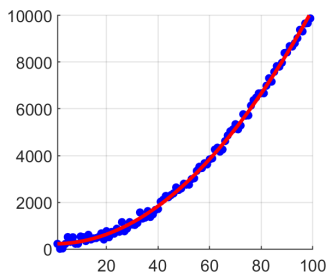
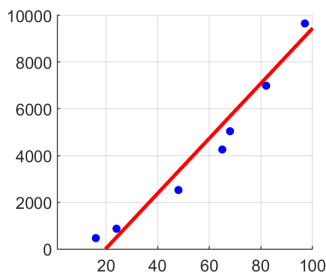


Assumption: **quadratic** model

Key questions:

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

Data distribution



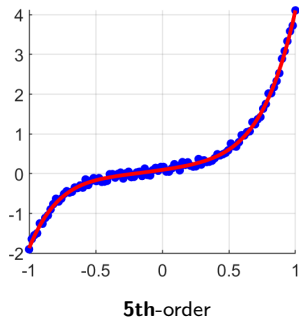
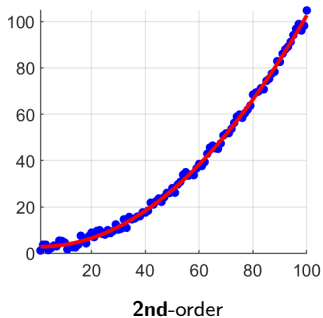
Assumption: **quadratic** model

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**?

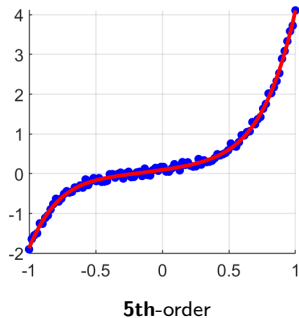
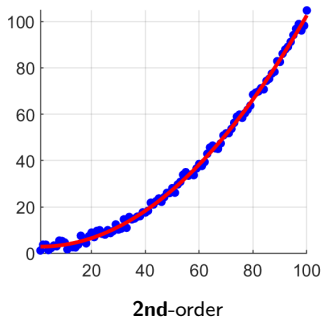
Polynomial regression

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



Polynomial regression

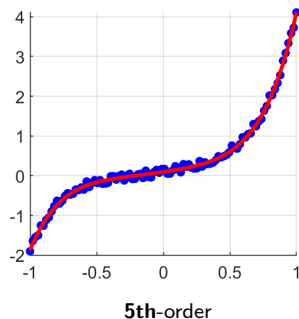
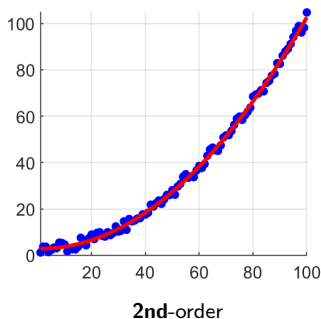
After the linear model, the simplest thing is a **polynomial model**.



The number of **parameters** grows with the order.

Polynomial regression

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



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.

$$y_i = a_3x_i^3 + a_2x_i^2 + a_1x_i + b \quad \text{for all data points } i = 1, \dots, n$$

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.

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

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.

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

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.

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

In matrix notation:

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

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

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 .

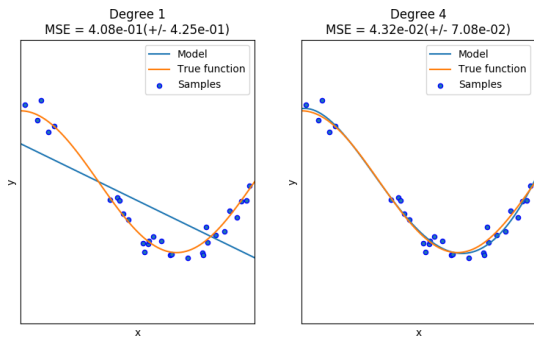
Thus, we can try to fit a polynomial in many cases.

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.

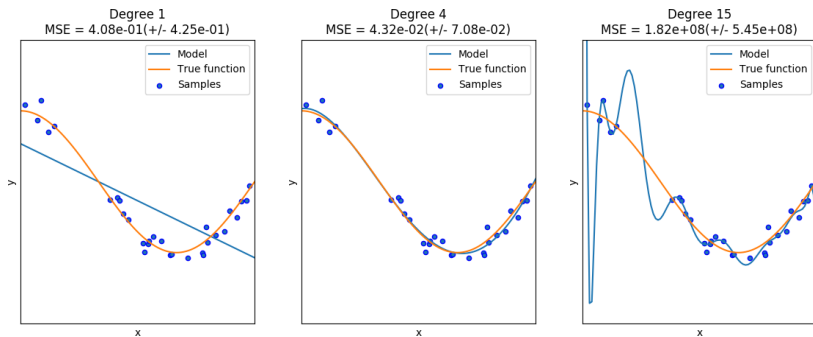


Polynomial fitting

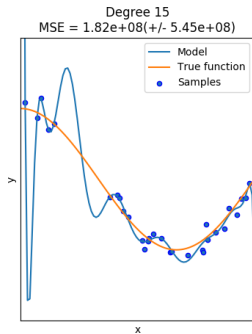
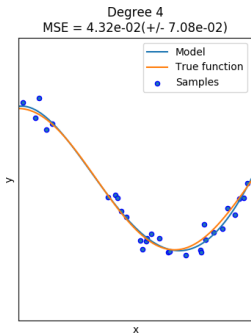
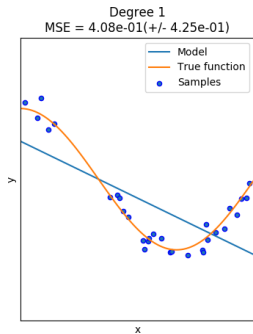
An application of the [Stone-Weierstrass theorem](#) tells us:

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 .

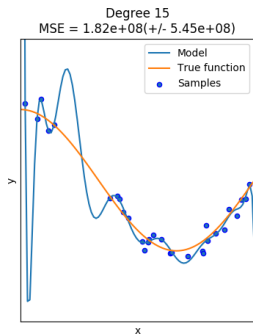
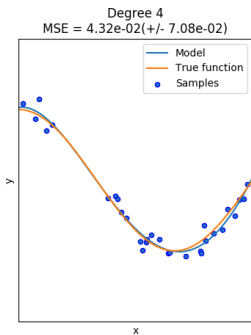
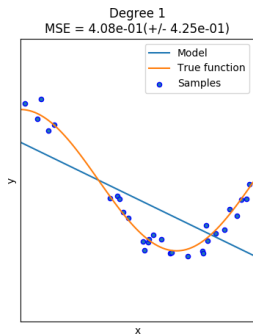
Thus, we can try to fit a polynomial in many cases.



Underfitting vs. Overfitting

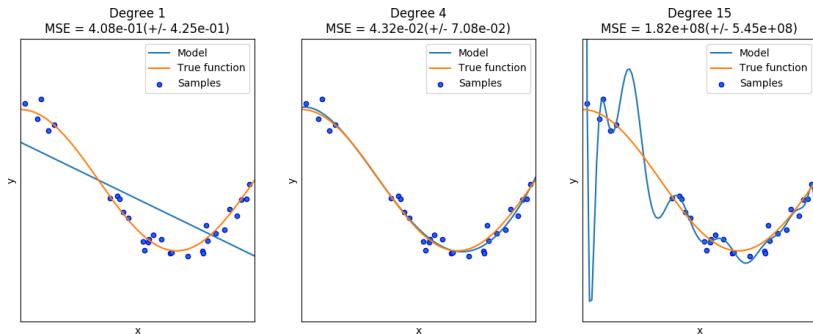


Underfitting vs. Overfitting



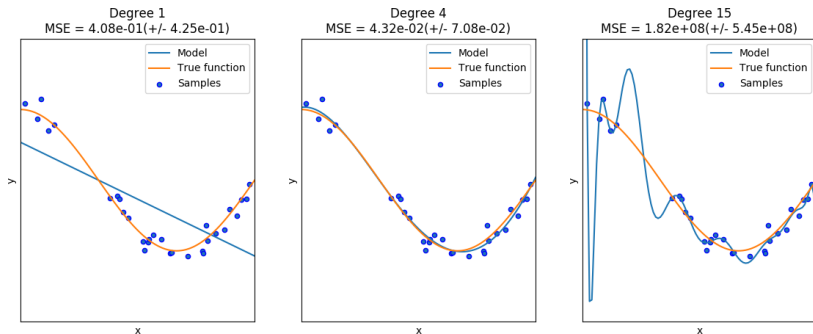
- **Underfitting:** Not sufficiently fitting the data (large MSE).

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 vs. Overfitting

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

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?

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

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?

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

From now on, we embrace the idea that many natural phenomena of interest are **nonlinear**.

Regularization penalties

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

For example, avoid **large** parameters to **counteract overfitting**:

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

Regularization penalties

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

For example, avoid **large** parameters to **counteract overfitting**:

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

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**:

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

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

More in general:

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

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**:

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

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

More in general:

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

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**:

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

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

More in general:

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

Controlling parameter growth is generally known as **shrinkage**.

Classification

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

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

Classification

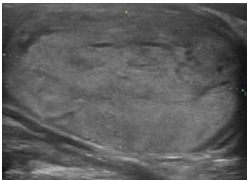
What if we want to predict a **category** instead of a value?

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

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?

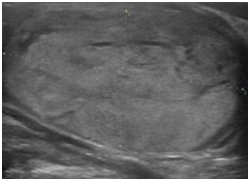
$$f_{\Theta}(\text{img}) = \{0, 1\}$$


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

⇒ The solution is not necessarily an optimum anymore.

Classification

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

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


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

⇒ The solution is not necessarily an optimum anymore.

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

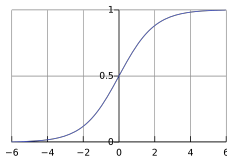
Logistic regression

New loss:

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

Here, σ is the nonlinear **logistic sigmoid**:

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



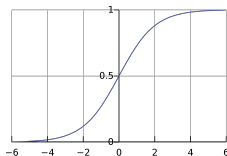
Logistic regression

New loss:

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

Here, σ is the nonlinear **logistic sigmoid**:

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



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

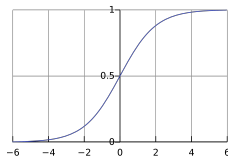
Logistic regression

New loss:

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

Here, σ is the nonlinear **logistic sigmoid**:

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



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

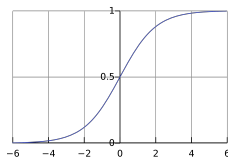
Logistic regression

New loss:

$$\ell_{\Theta}(\{x_i, y_i\}) = \sum_{i=1}^n c(x_i, y_i), \quad \text{with}$$
$$c(x_i, y_i) = \begin{cases} -\ln(\sigma(ax_i + b)) & y_i = 1 \\ -\ln(1 - \sigma(ax_i + b)) & y_i = 0 \end{cases} \quad \text{convex}$$

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

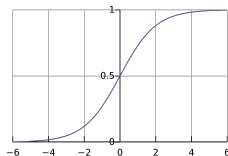
New loss:

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

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

Here, σ is the nonlinear **logistic sigmoid**:

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



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

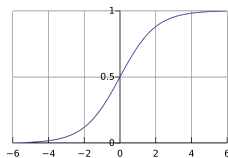
Logistic regression

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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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 similarly for the **second term** and for parameter b .

Logistic regression: Finding a solution

By looking at the partial derivative:

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

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

Logistic regression: Finding a solution

By looking at the partial derivative:

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

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.

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:

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

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	
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:

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

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	least squares
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	least squares
logistic regression	convex	nonlinear optimization

Suggested reading

On polynomial regression vs. neural nets:

<https://arxiv.org/pdf/1806.06850>

Proof that the logistic loss is convex:

<https://math.stackexchange.com/questions/1582452/>

logistic-regression-prove-that-the-cost-function-is-convex