Lecture 1 Recap

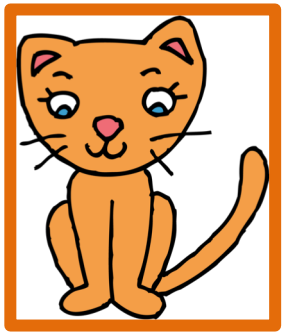
Machine learning

Unsupervised learning

Supervised learning

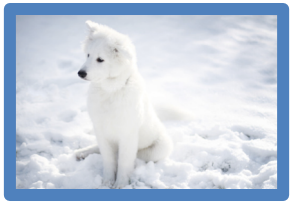
Reinforcement learning

Nearest Neighbor







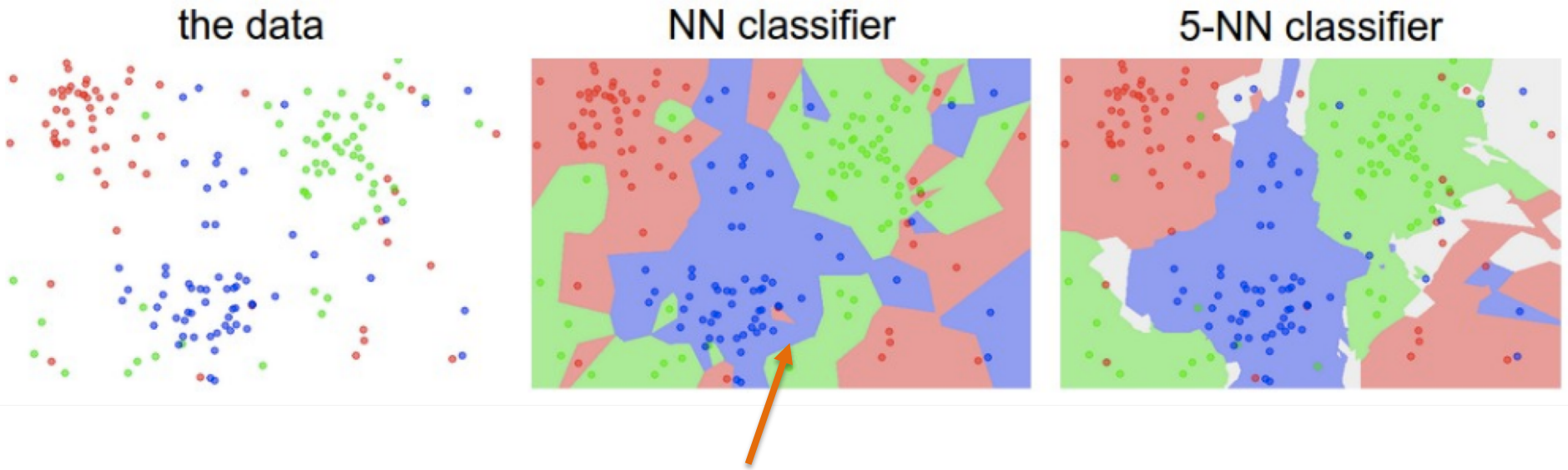




Nearest Neighbor

NN classifier = dog

distance

Nearest Neighbor



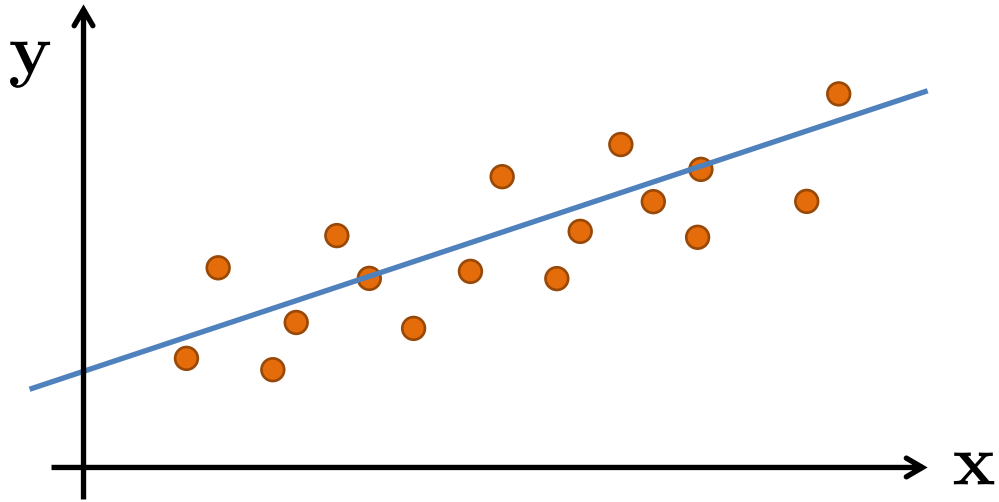
Decision boundaries

Courtesy of Stanford course cs231n

Linear Regression

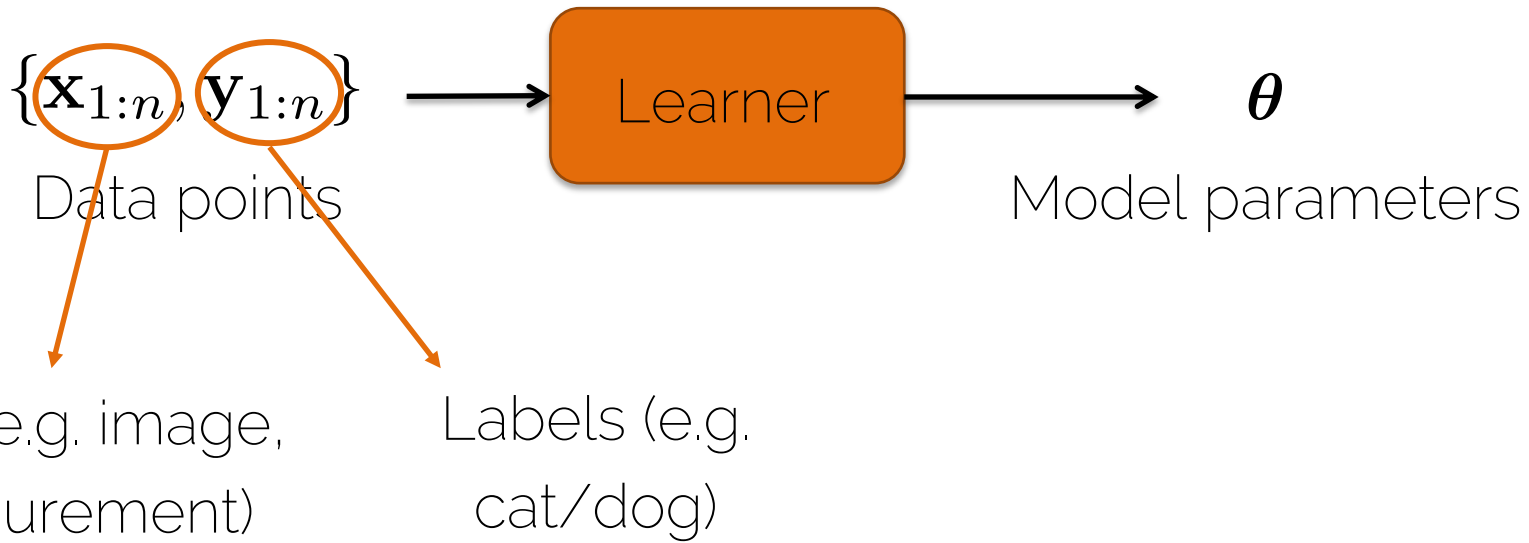
Linear regression

- Supervised learning
- Find a linear model that explains a target $\mathbf{y}$ given the inputs $\mathbf{X}$



Linear regression

Training



Linear regression

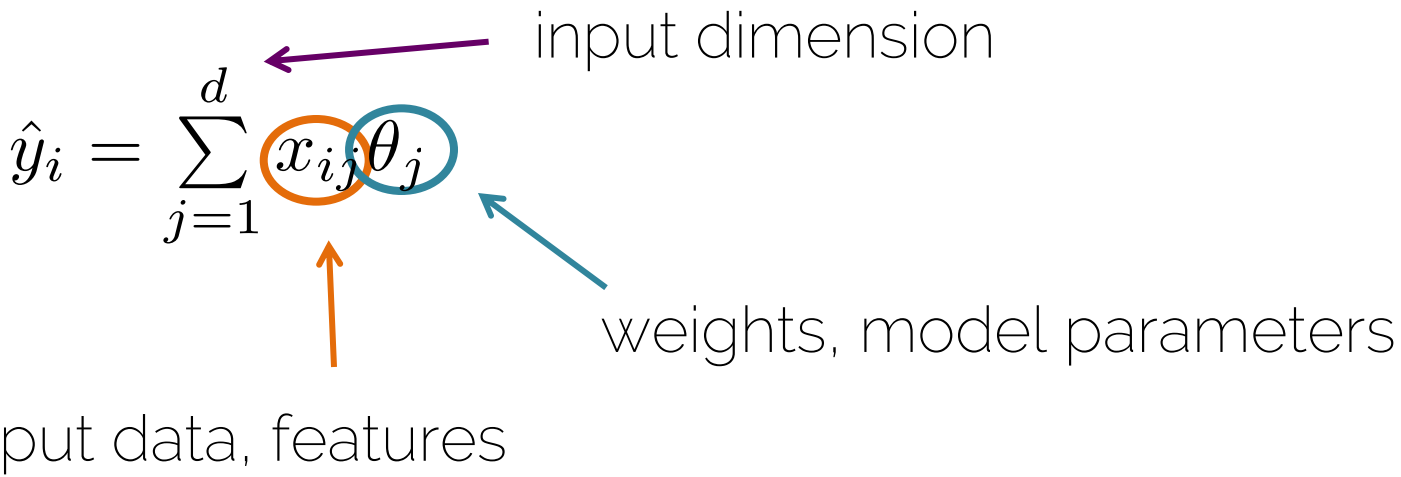
can be a Neural Network

Training

Testing

Linear prediction

- A linear model is expressed in the form

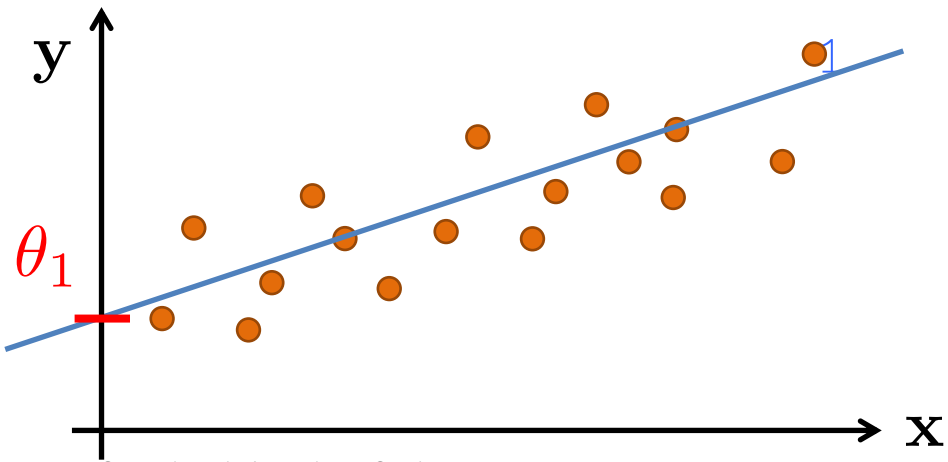


Linear prediction

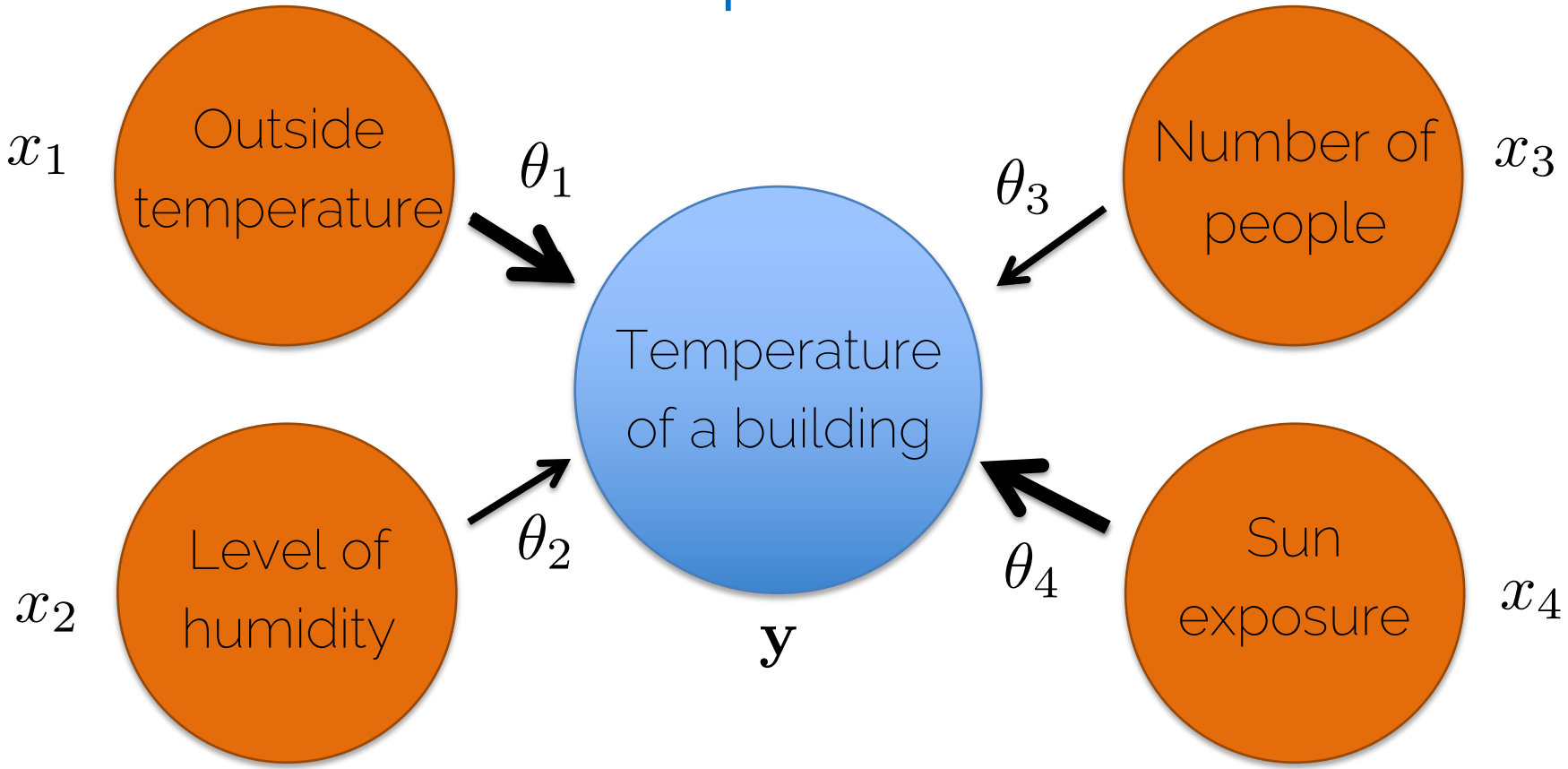
- A linear model is expressed in the form

$$\hat{y}_i = \sum_{j=1}^{d} x_{ij}\theta_j = x_{i1}\theta_1 + x_{i2}\theta_2 + \cdots + x_{id}\theta_d$$

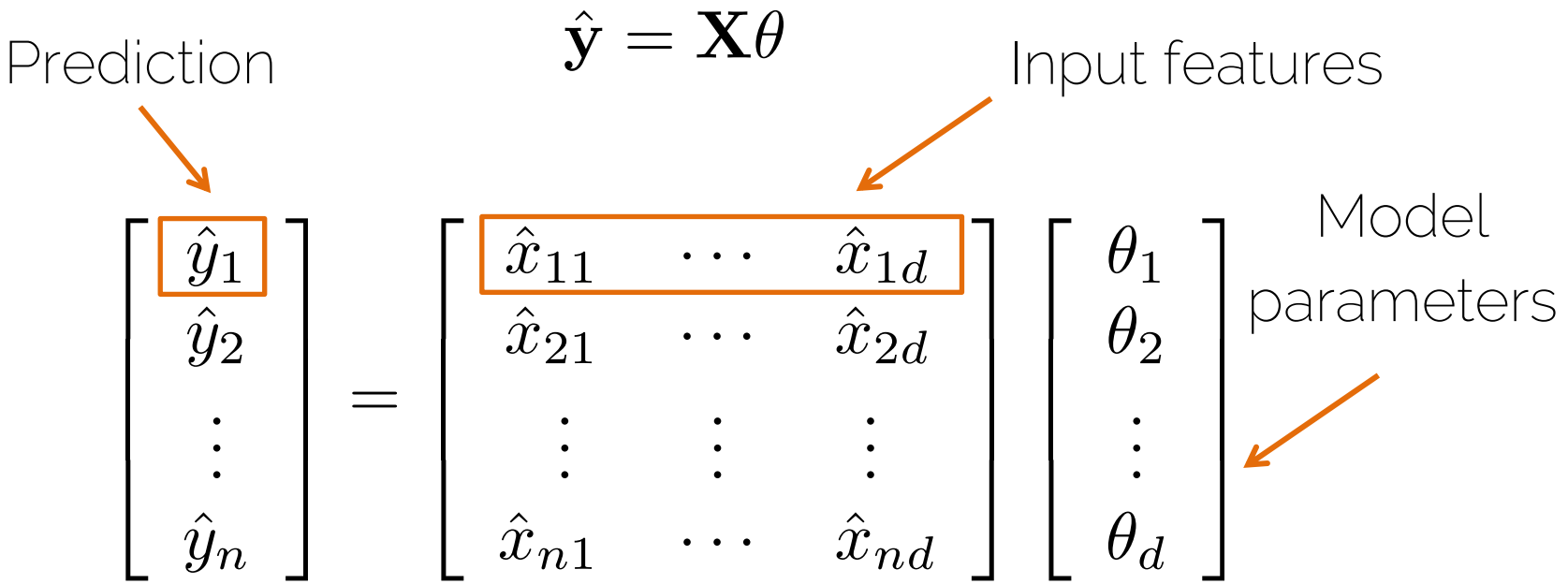
bias



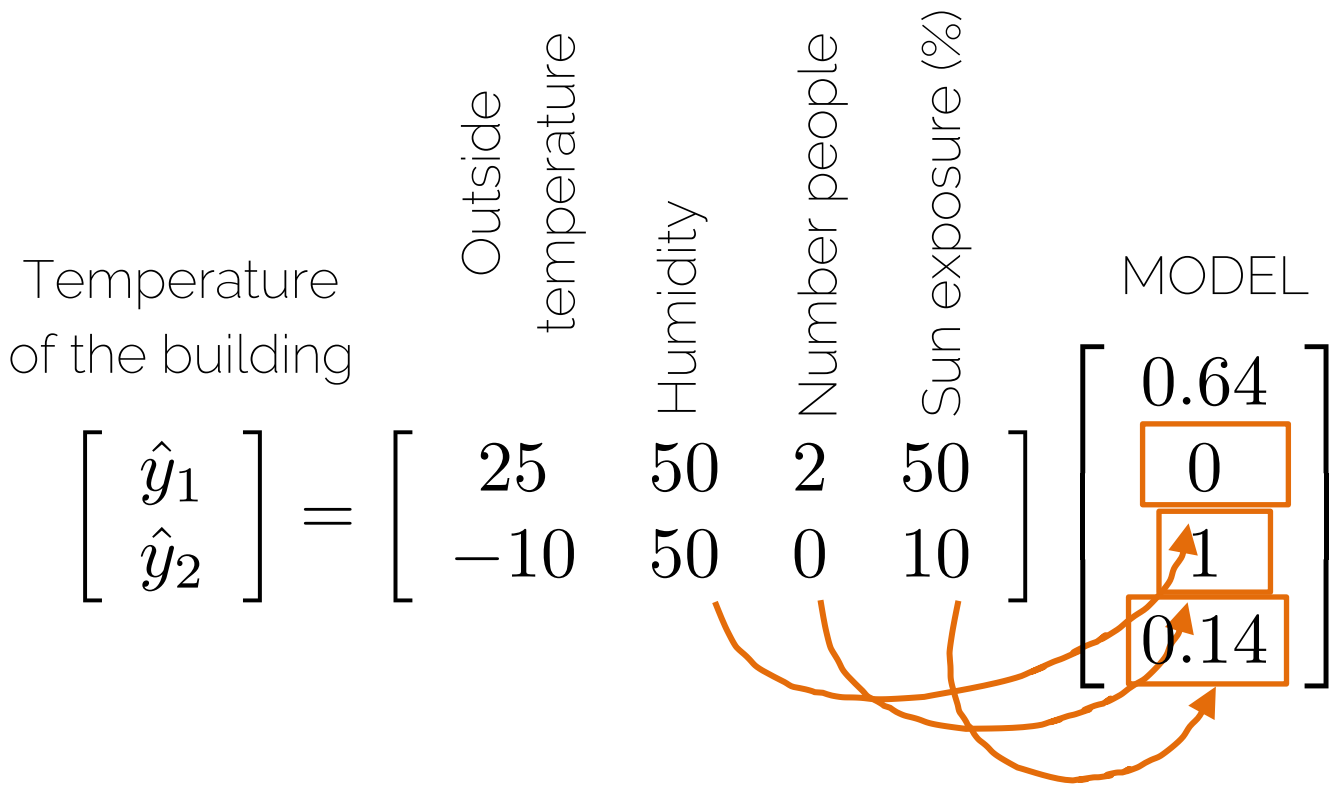
Linear prediction



Linear prediction



Linear prediction



Linear prediction

Temperature of the building

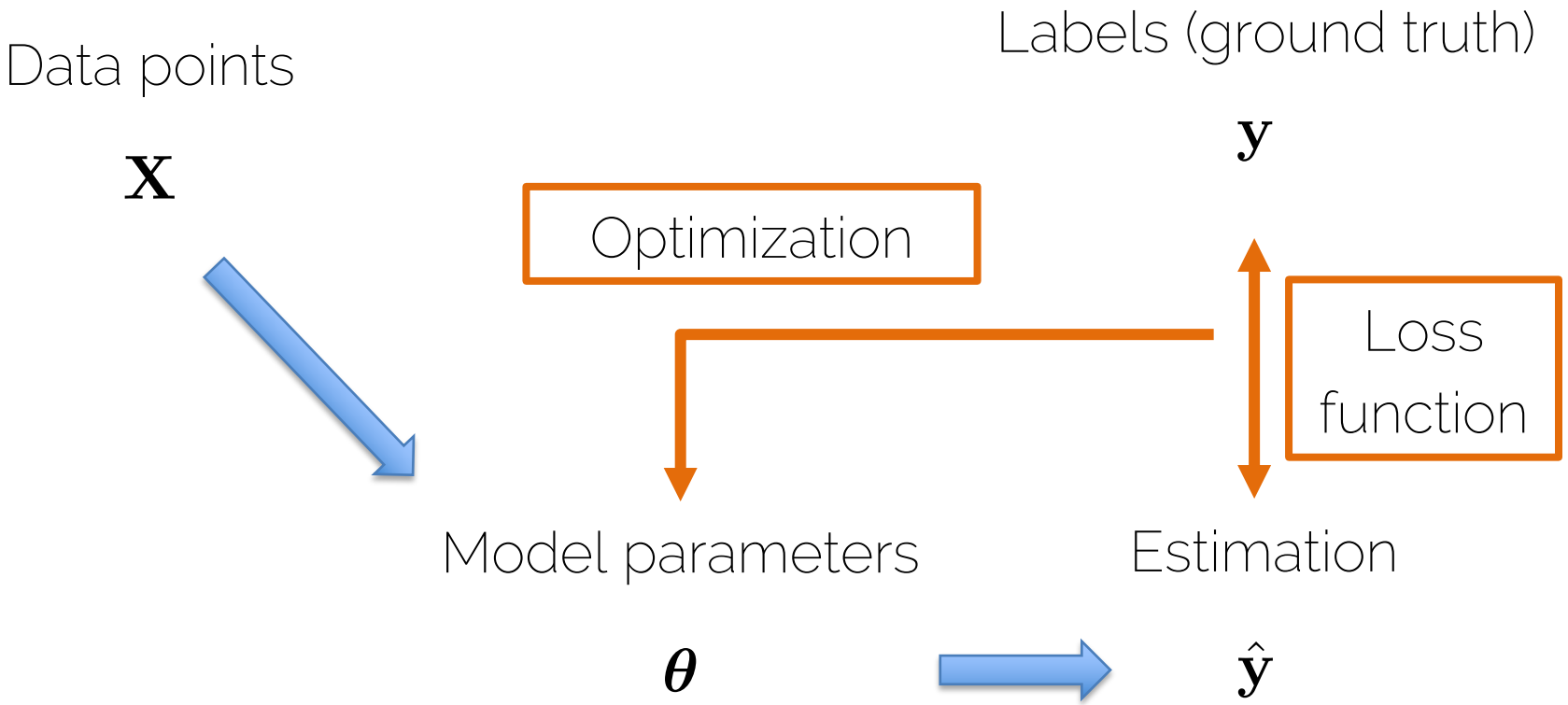
$$\begin{bmatrix} 25 \\ -5 \end{bmatrix} = \begin{bmatrix} 25 & 50 & 2 & 50 \\ -10 & 50 & 0 & 10 \end{bmatrix} \begin{bmatrix} 0.64 \\ 0 \\ 1 \\ 0.14 \end{bmatrix}$$

Outside temperature | Humidity | Number people | Sun exposure (%)

MODEL

How do we obtain the model?

How to obtain the model?

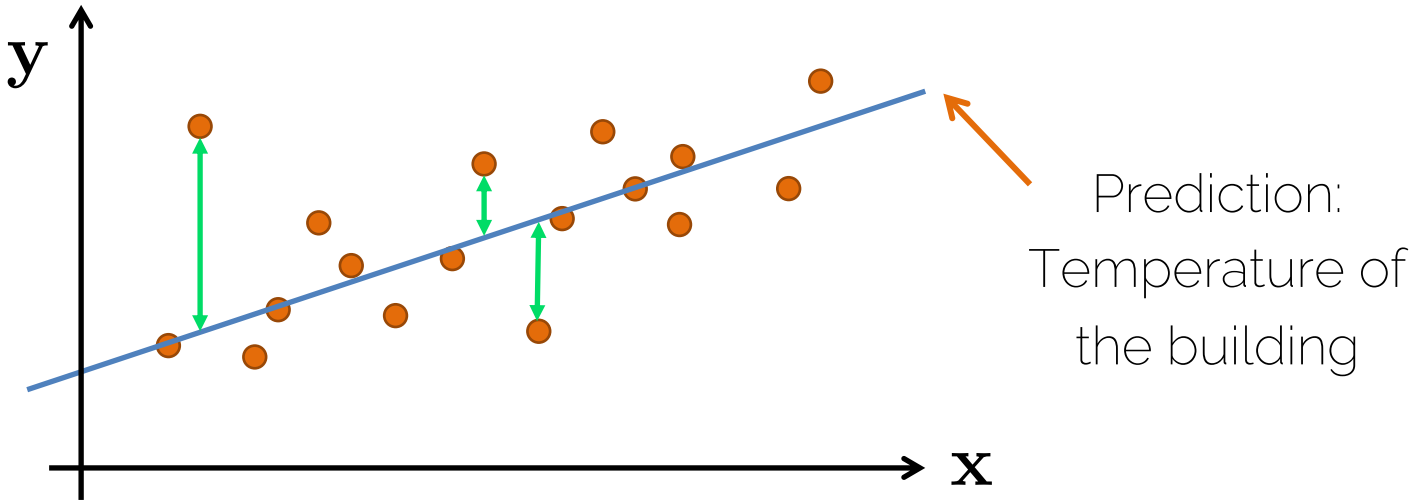


How to obtain the model?

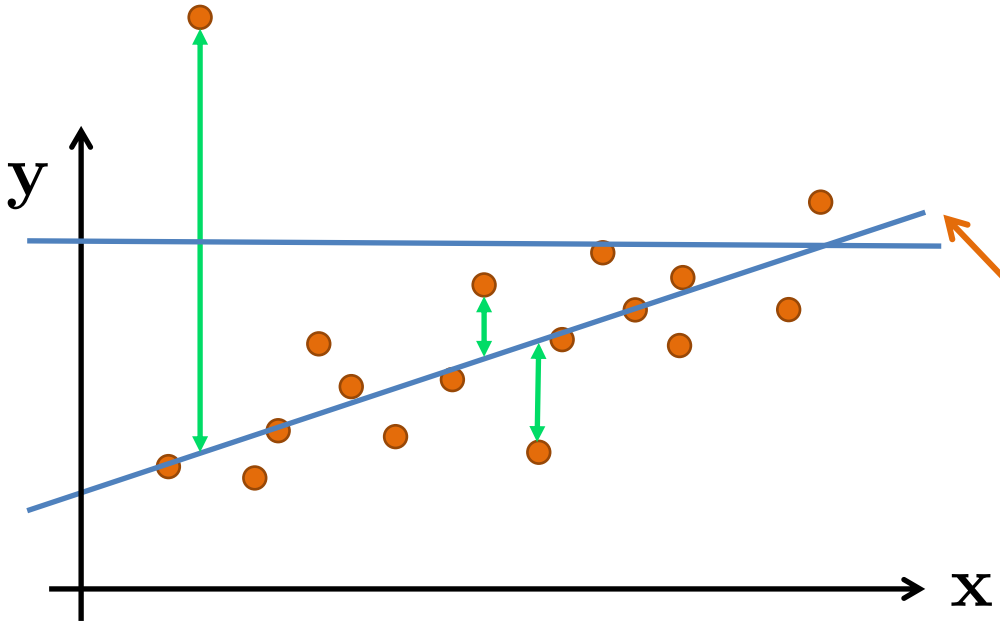
- Loss function: measures how good my estimation is (how good my model is) and tells the optimization method how to make it better.

- Optimization: changes the model in order to improve the loss function (i.e. to improve my estimation).

Linear regression: loss function

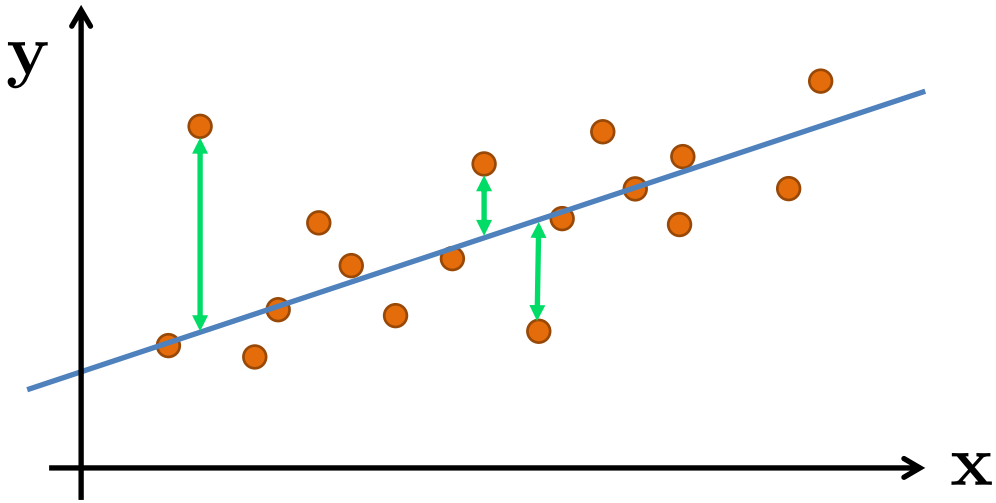


Linear regression: loss function



Prediction:
Temperature of
the building

Linear regression: loss function



Minimizing $J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$

Objective function
Energy
Cost function

Optimization: linear least squares

- Linear least squares: an approach to fit a mathematical model to the data

$$\min_{\boldsymbol{\theta}} \; J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

- Convex problem, there exists a closed-form solution that is unique.

Optimization: linear least squares

$$\min_{\boldsymbol{\theta}} \; J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i \boldsymbol{\theta} - y_i)^2$$

n training samples

The estimation comes from the linear model

Optimization: linear least squares

$$\min_{\boldsymbol{\theta}} \; J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i\boldsymbol{\theta} - y_i)^2$$

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Matrix notation

n training samples, each input vector has size d

n labels

Optimization: linear least squares

$$\min_{\boldsymbol{\theta}} \; J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i\boldsymbol{\theta} - y_i)^2$$

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Matrix notation

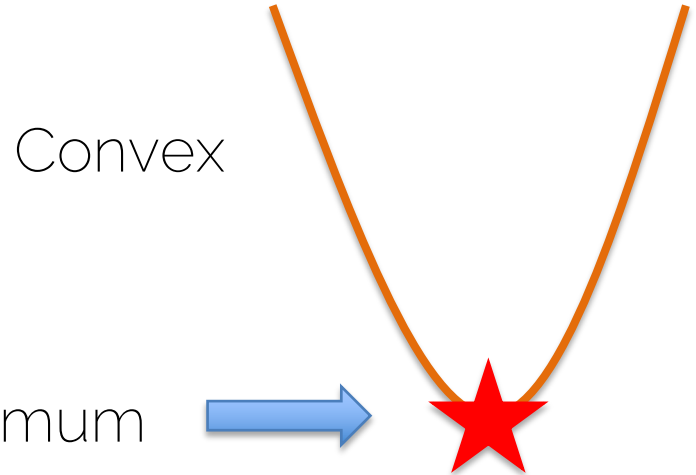
More on matrix notation in the next exercise session

Optimization: linear least squares

$$\min_{\boldsymbol{\theta}} \; J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i\boldsymbol{\theta} - y_i)^2$$

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 0$$



Optimization

$$J(\boldsymbol{\theta}) = (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - 2\boldsymbol{\theta}^T\mathbf{X}^T\mathbf{y} + \mathbf{y}^T\mathbf{y})$$

No theta there!

$$\frac{\partial \boldsymbol{\theta}^T\mathbf{A}\boldsymbol{\theta}}{\partial \boldsymbol{\theta}} = 2\mathbf{A}^T\boldsymbol{\theta}$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - 2\mathbf{X}^T\mathbf{y} = 0$$

Details in the exercise session!

Optimization

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - 2\mathbf{X}^T\mathbf{y} = 0$$

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

We have found an analytical solution to a convex problem

Inputs: Outside temperature, number of people…

Output: Temperature of the building

Is this the best estimate?

- Least squares estimate

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Maximum Likelihood

Maximum Likelihood Estimate

$p_{data}(\mathbf{y}|\mathbf{X})$ True underlying distribution

$p_{model}(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ Parametric family of distributions

Controlled by a parameter

Maximum Likelihood Estimate

- A method of estimating the parameters of a statistical model given observations,

$$p_{model}(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

Observations from $p_{data}(\mathbf{y}|\mathbf{X})$

Maximum Likelihood Estimate

- A method of estimating the parameters of a statistical model given observations, by finding the parameter values that **maximize the likelihood** of making the observations given the parameters.

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} p_{model}(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

Training samples

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} \prod_{i=1}^{n} p_{model}(y_i|\mathbf{x}_i, \boldsymbol{\theta})$$

Maximum Likelihood Estimate

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} \prod_{i=1}^{n} p_{model}(y_i|\mathbf{x}_i, \boldsymbol{\theta})$$

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} \sum_{i=1}^{n} \log p_{model}(y_i|\mathbf{x}_i, \boldsymbol{\theta})$$

Logarithmic property $log_c(ab) = log_c(a) + log_c(b)$

Back to linear regression

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} \sum_{i=1}^{n} \log p_{model}(y_i|\mathbf{x}_i, \boldsymbol{\theta})$$

Back to linear regression

$$p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$$

Gaussian or Normal distribution

Assuming $y_i = \mathcal{N}(\mathbf{x}_i\boldsymbol{\theta}, \sigma^2) = \mathbf{x}_i\boldsymbol{\theta} + \mathcal{N}(0, \sigma^2)$

mean

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \qquad x \sim \mathcal{N}(\mu, \sigma^2)$$

Back to linear regression

$$p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$$

Assuming $y_i = \mathcal{N}(\mathbf{x}_i\boldsymbol{\theta}, \sigma^2) = \mathbf{x}_i\boldsymbol{\theta} + \mathcal{N}(0, \sigma^2)$

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \qquad x \sim \mathcal{N}(\mu, \sigma^2)$$

Back to linear regression

$$p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) = (2\pi\sigma^2)^{-1/2} e^{-\frac{1}{2\sigma^2}(y_i - \mathbf{x}_i\boldsymbol{\theta})^2}$$

Assuming $y_i = \mathcal{N}(\mathbf{x}_i\boldsymbol{\theta}, \sigma^2) = \mathbf{x}_i\boldsymbol{\theta} + \mathcal{N}(0, \sigma^2)$

mean

$$p(y_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y_i - \mu)^2}$$

$$y_i \sim \mathcal{N}(\mu, \sigma^2)$$

Back to linear regression

$$p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) = (2\pi\sigma^2)^{-1/2} e^{-\frac{1}{2\sigma^2}(y_i - \mathbf{x}_i\boldsymbol{\theta})^2}$$

Original optimization problem

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} \sum_{i=1}^{n} \log p_{model}(y_i|\mathbf{x}_i, \boldsymbol{\theta})$$

Back to linear regression

$$\log((2\pi\sigma^2)^{-1/2} e^{-\frac{1}{2\sigma^2}(y_i - \mathbf{x}_i\boldsymbol{\theta})^2})$$

Matrix notation

$$\log((2\pi\sigma^2)^{-1/2} e^{-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})})$$

Canceling log and e

$$-\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})$$

Back to linear regression

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} \sum_{i=1}^{n} \log p_{model}(y_i|\mathbf{x}_i, \boldsymbol{\theta})$$

$$-\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})$$

$$\downarrow \frac{\partial}{\partial \boldsymbol{\theta}}$$

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

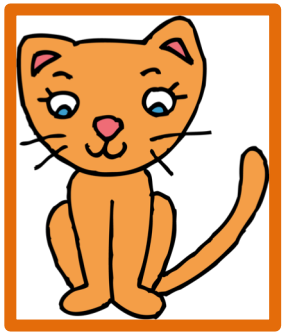
How can we find the estimate of theta?

Linear regression

- Maximum Likelihood Estimate (MLE) corresponds to the Least Squares Estimate (given the assumptions)

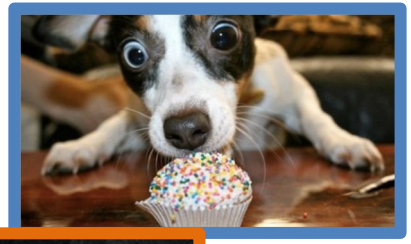
- Introduced the concepts of loss function and optimization to obtain the best model for regression

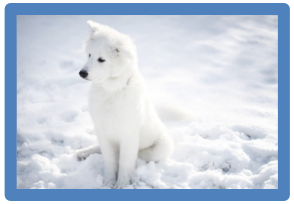
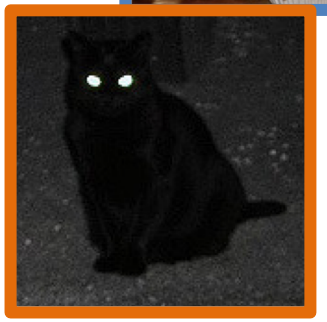
Image classification









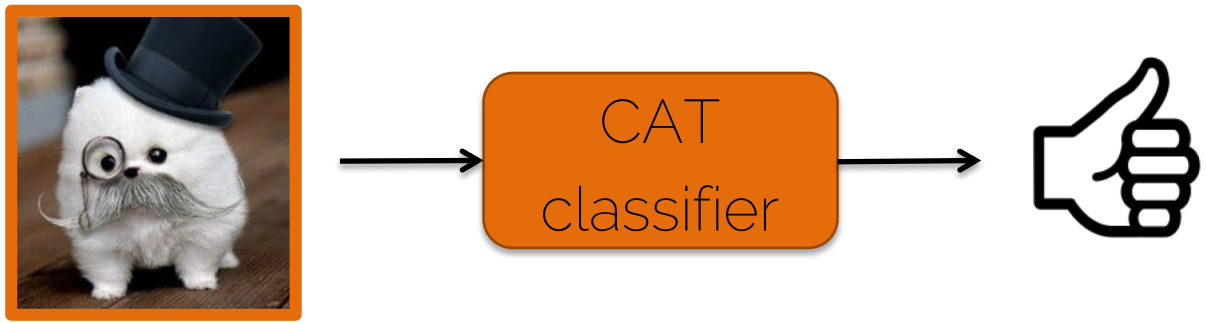



Regression vs. Classification

- Regression: predict a continuous output value (e.g. temperature of a room)

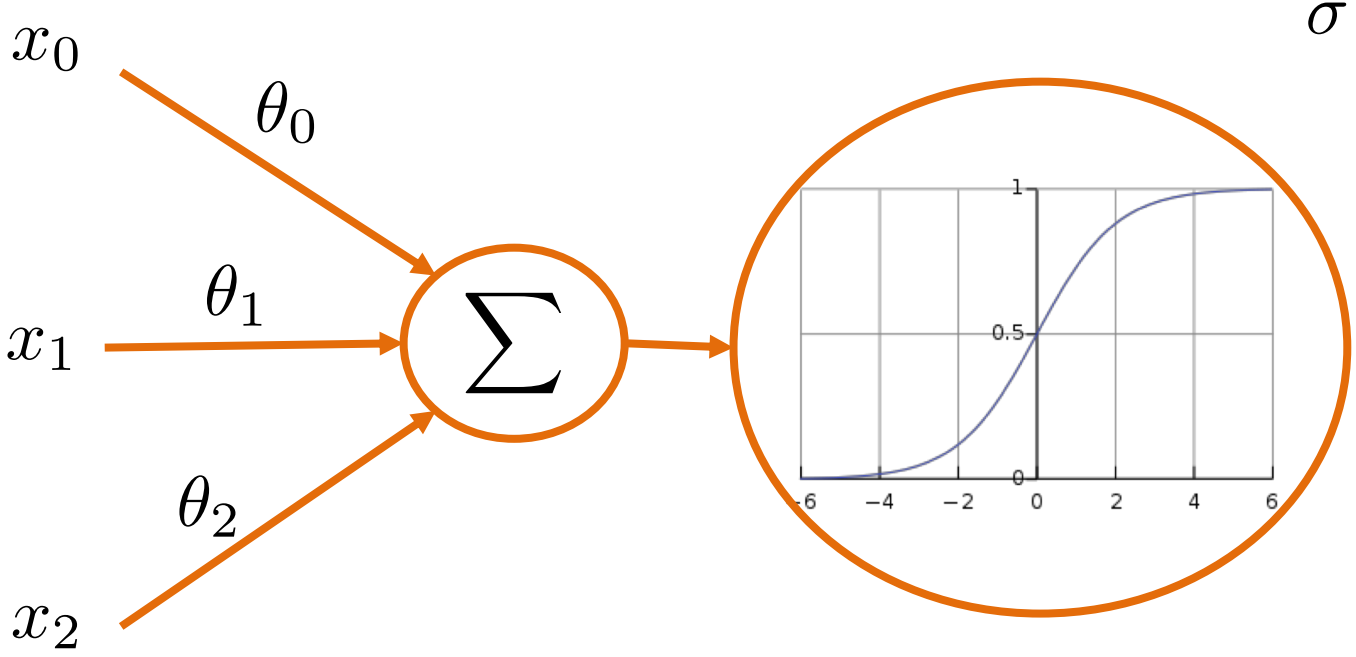
- Classification: predicts a discrete value
  - Binary classification: output is either 0 or 1 
  - Multi-class classification: set of N classes

Logistic regression



Sigmoid for binary predictions

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



1

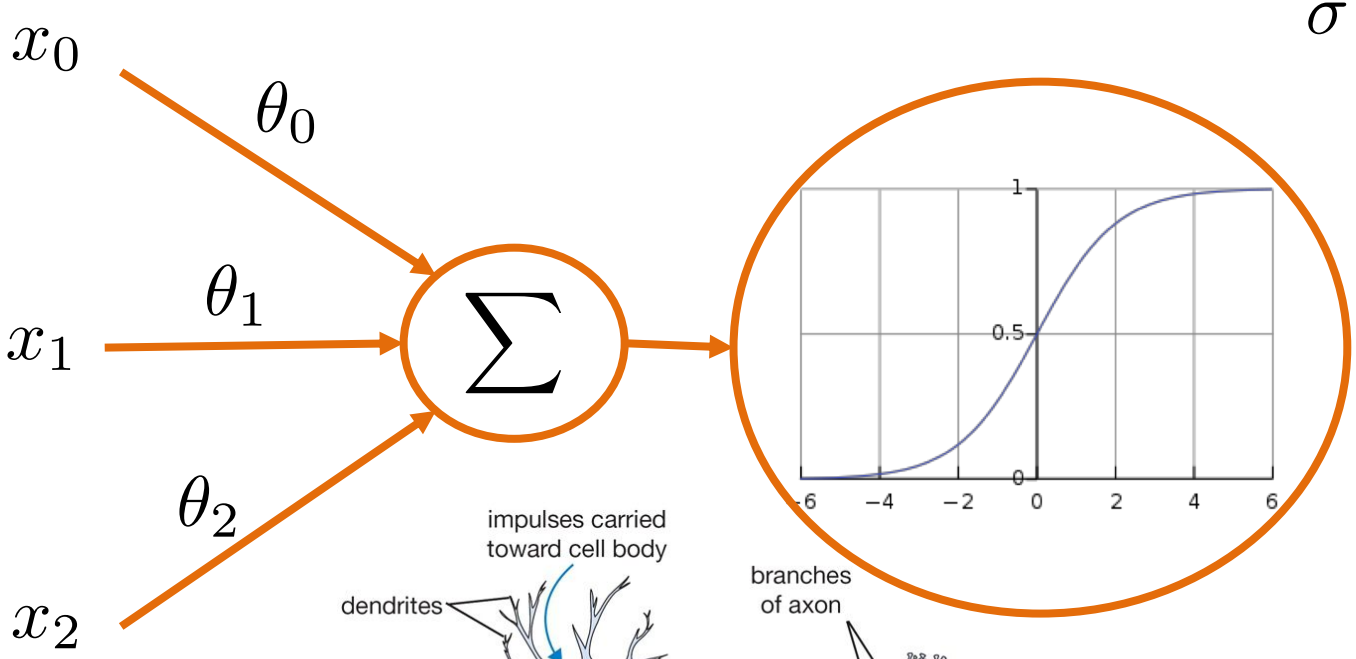
Can be interpreted as a probability

0

$$p(y_i = 1 | \mathbf{x}_i, \boldsymbol{\theta})$$

Spoiler alert: 1 layer Neural Network

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



1

Can be interpreted as a probability

0

$$p(y_i = 1 | \mathbf{x}_i, \boldsymbol{\theta})$$

Logistic regression

- Probability of a binary output

$$\hat{\mathbf{y}} = p(\mathbf{y} = 1 | \mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i = 1 | \mathbf{x}_i, \boldsymbol{\theta})$$

$$\hat{y}_i = \sigma(\mathbf{x}_i \boldsymbol{\theta})$$

Logistic regression

- Probability of a binary output

$$\hat{\mathbf{y}} = p(\mathbf{y} = 1 | \mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i = 1 | \mathbf{x}_i, \boldsymbol{\theta})$$

Bernoulli trial

Model for coins

$$p(x|\phi) = \phi^x (1-\phi)^{1-x} = \begin{cases} \phi & \text{if} \quad x = 1 \\ 1-\phi & \text{if} \quad x = 0 \end{cases}$$

Logistic regression

- Probability of a binary output

$$\hat{\mathbf{y}} = p(\mathbf{y} = 1|\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i = 1|\mathbf{x}_i, \boldsymbol{\theta})$$

$$\hat{\mathbf{y}} = \prod_{i=1}^{n} \hat{y}_i^{y_i} (1 - \hat{y}_i)^{(1-y_i)}$$

Model for coins

Logistic regression: loss function

- Probability of a binary output

$$p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = \quad \hat{\mathbf{y}} = \prod_{i=1}^{n} \hat{y}_i^{y_i} (1-\hat{y}_i)^{(1-y_i)}$$

- Maximum Likelihood Estimate

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} \log p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta})$$

Logistic regression: loss function

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \hat{\mathbf{y}} = \prod_{i=1}^{n} \hat{y}_i^{y_i} (1 - \hat{y}_i)^{(1-y_i)}$$

$$\sum_{i=1}^{n} \log \left(\hat{y}_i^{y_i} (1 - \hat{y}_i)^{(1-y_i)} \right)$$

$$\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

Logistic regression: loss function

$$\mathcal{L}(\hat{y}_i, y_i) = y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

$$y_i = 1 \longrightarrow \mathcal{L}(\hat{y}_i, 1) = \log \hat{y}_i$$

Maximize! $$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

Logistic regression: loss function

$$\mathcal{L}(\hat{y}_i, y_i) = y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

$$y_i = 1 \longrightarrow \mathcal{L}(\hat{y}_i, 1) = \log \hat{y}_i$$

We want. $\log \hat{y}_i$ large, since logarithm is a monotonically increasing function, we also want $\hat{y}_i$ large (1 is the largest value my estimate can take!)

Logistic regression: loss function

$$\mathcal{L}(\hat{y}_i, y_i) = y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

$$y_i = 1 \longrightarrow \mathcal{L}(\hat{y}_i, 1) = \log \hat{y}_i$$

$$y_i = 0 \longrightarrow \mathcal{L}(\hat{y}_i, 0) = \log(1 - \hat{y}_i)$$

We want. $\log(1 - \hat{y}_i)$ large, so we want $\hat{y}_i$ to be small (0 is the smallest value my estimate can take!)

Logistic regression: loss function

$$\mathcal{L}(\hat{y}_i, y_i) = y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

Referred to as **cross-entropy** loss

- Related to the multi-class loss you will see in this course (also called softmax loss)

Logistic regression: optimization

- Loss function

$$\mathcal{L}(\hat{y}_i, y_i) = y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

- Cost function

$$C(\boldsymbol{\theta}) = -\frac{1}{n} \sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

Minimization

$$\hat{y}_i = \sigma(\mathbf{x}_i \boldsymbol{\theta})$$

Logistic regression: optimization

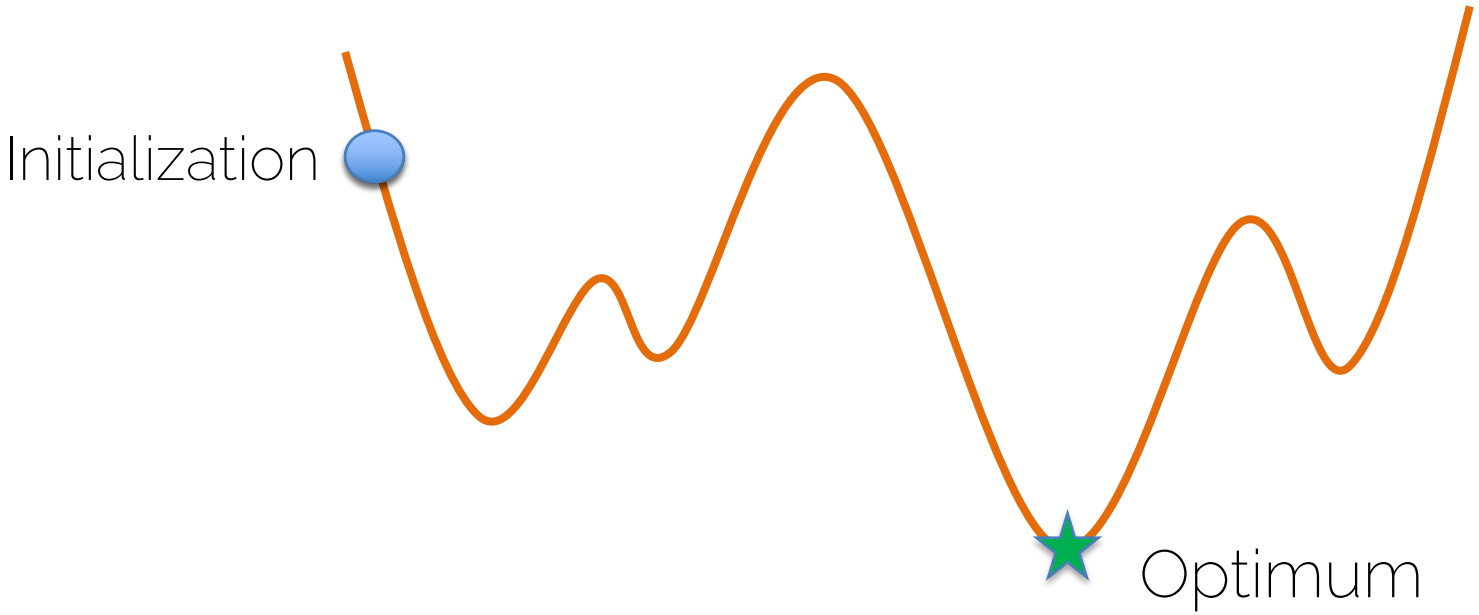
- No closed-form solution

- Make use of an iterative method → gradient descent

Gradient descent

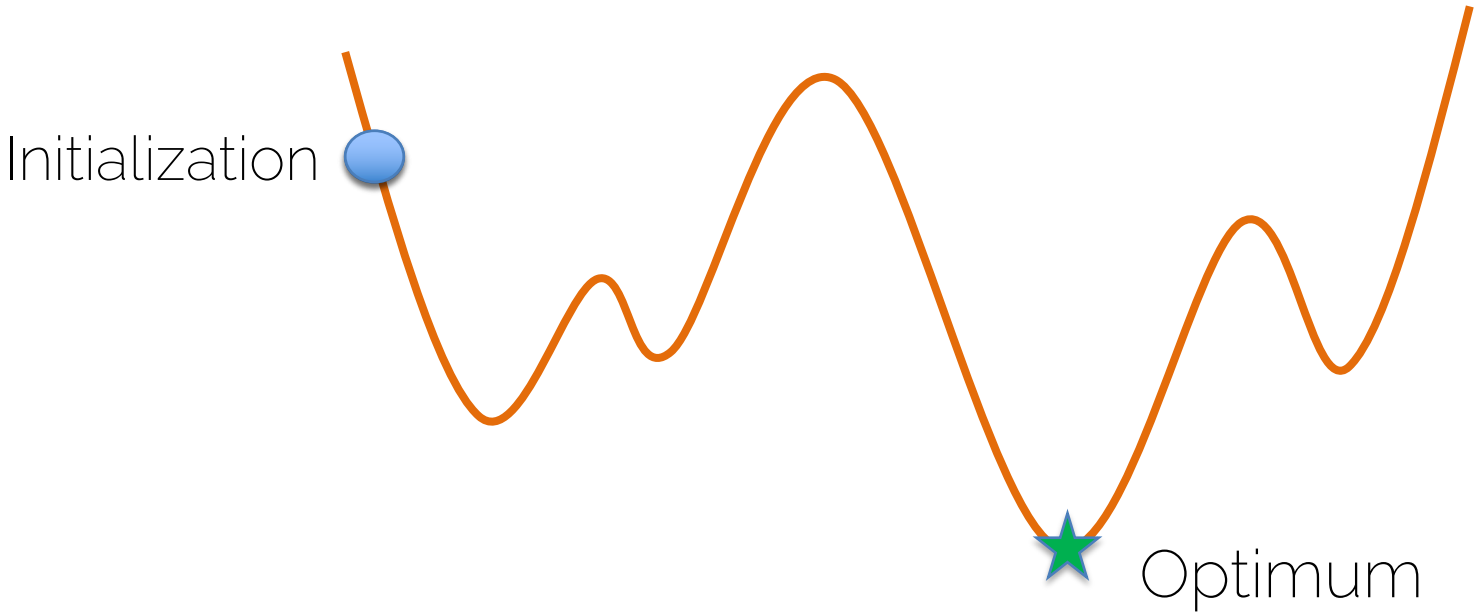
Following the slope

$$\mathbf{x}^* = \arg\min f(\mathbf{x})$$



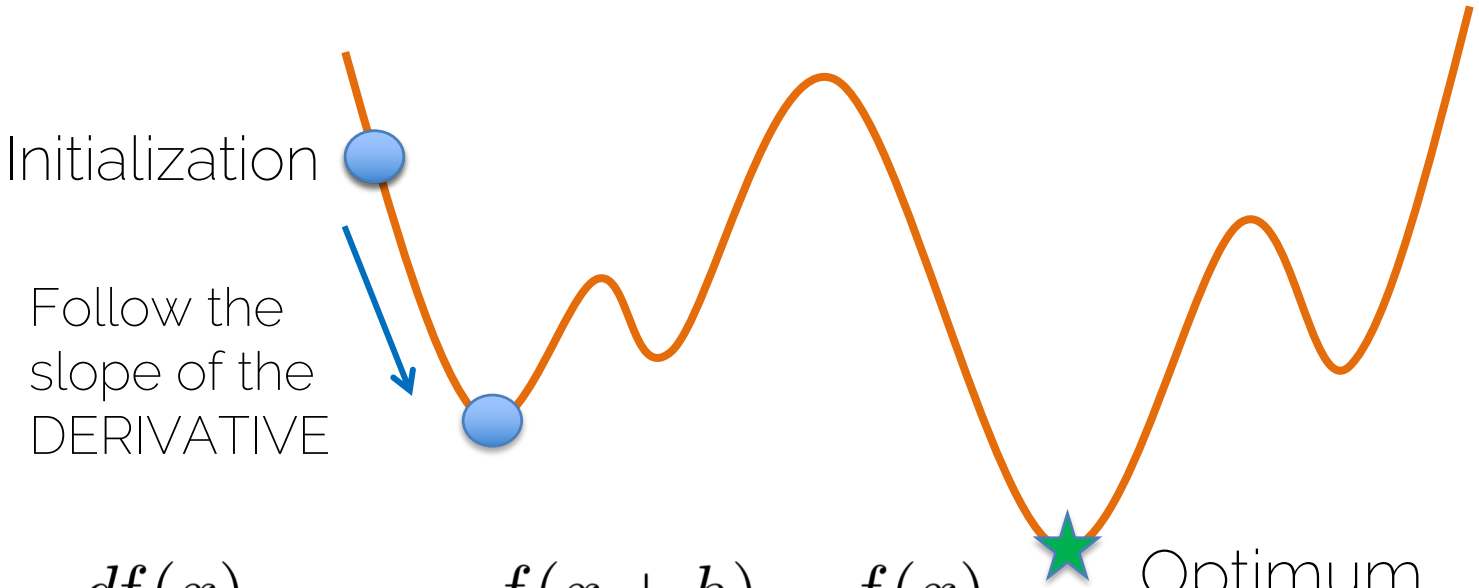
Following the slope

$$\mathbf{x}^* = \arg\min f(\mathbf{x})$$



Following the slope

$$\mathbf{x}^* = \arg\min f(\mathbf{x})$$



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

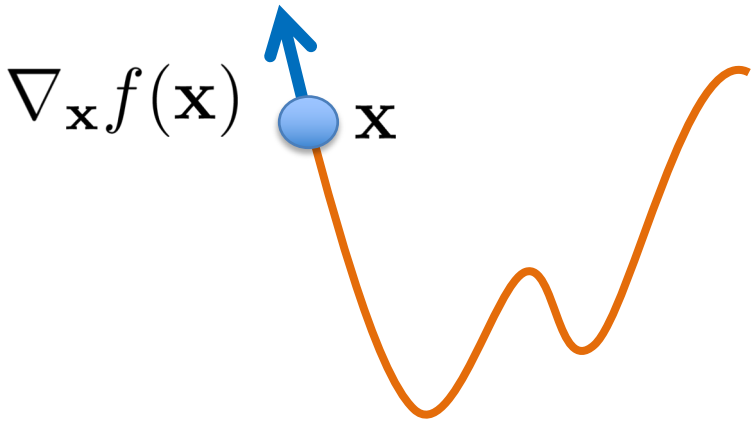
Gradient steps

- From derivative to gradient

$$\frac{df(x)}{dx} \longrightarrow \nabla_{\mathbf{x}} f(\mathbf{x})$$

Direction of greatest increase of the function

- Gradient steps in direction of negative gradient



$$\mathbf{x}' = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x})$$

Learning rate

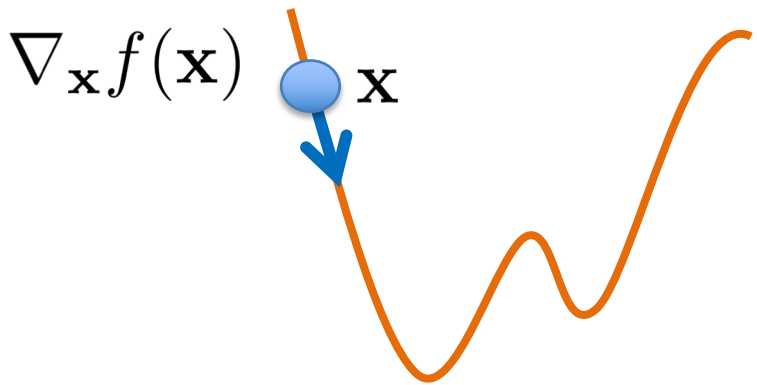
Gradient steps

- From derivative to gradient

$$\frac{df(x)}{dx} \longrightarrow \nabla_{\mathbf{x}} f(\mathbf{x})$$

Direction of greatest increase of the function

- Gradient steps in direction of negative gradient



$$\mathbf{x}' = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x})$$

SMALL Learning rate

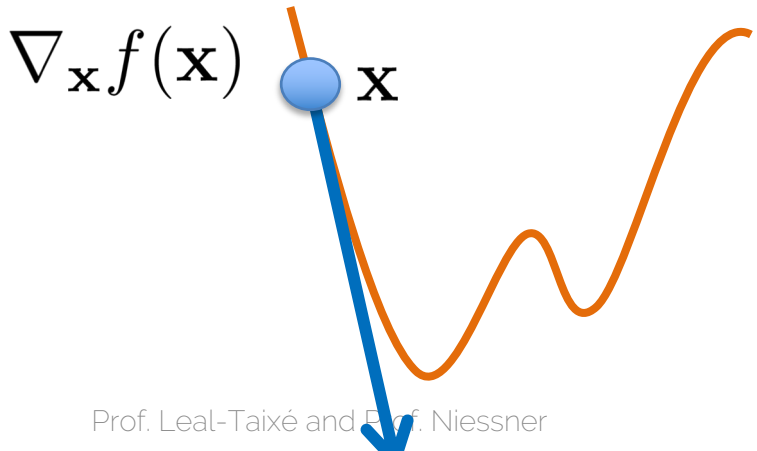
Gradient steps

- From derivative to gradient

$$\frac{df(x)}{dx} \longrightarrow \nabla_{\mathbf{x}} f(\mathbf{x})$$

Direction of greatest increase of the function

- Gradient steps in direction of negative gradient

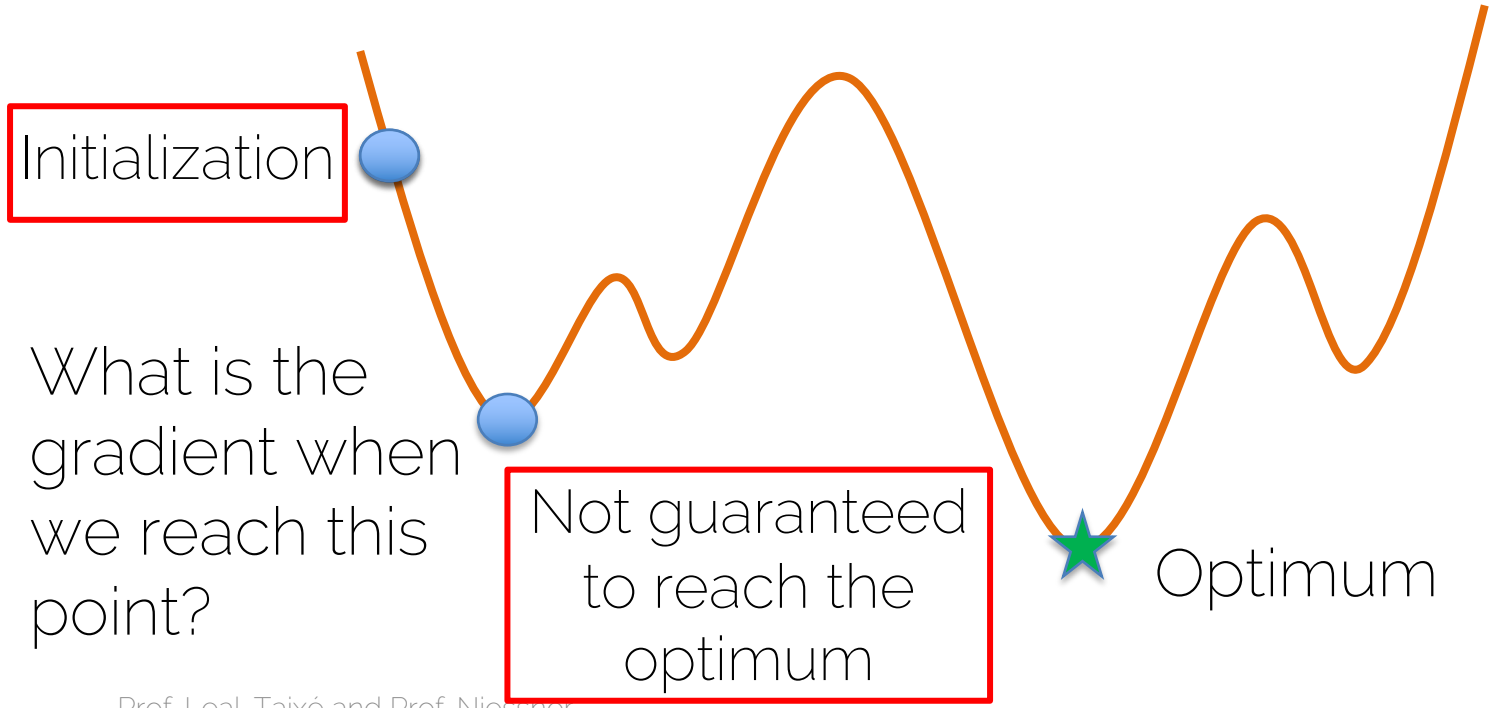


$$\mathbf{x}' = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x})$$

LARGE Learning rate

Convergence

$$\mathbf{x}^* = \arg\min f(\mathbf{x})$$



Numerical gradient

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

- Approximate
- Slow evaluation

Analytical gradient

- Exact and fast

Recall Linear Regression $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Analytical gradient

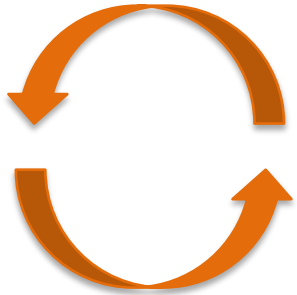
$$2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - 2\mathbf{X}^T\mathbf{y}$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 0$$

A step in that direction

Gradient descent for least squares

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \left(2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - 2\mathbf{X}^T\mathbf{y}\right)$$



Convex, always converges to the same solution

Gradient descent for logistic regression

$$C(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)$$

$$\frac{\partial C(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

Explained in the following lectures when the following topics are introduced:

- Computational graphs
- Backpropagation of the gradients

Regularization

Regularization

Loss $J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta})$

L2 regularization

L1 regularization

Max norm regularization

Dropout

Regularization: example

- Input: 3 features $\mathbf{x} = [1, 2, 1]$

- Two linear classifiers that give the same result:

$\boldsymbol{\theta}_1 = [0, 0.75, 0]$ → Ignores 2 features

$\boldsymbol{\theta}_2 = [0.25, 0.5, 0.25]$ → Takes information from all features

Regularization: example

Loss $J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta})$

L2 regularization $R(\boldsymbol{\theta}) = \sum_{i=1}^{n} \theta_i^2$

$\boldsymbol{\theta}_1 \longrightarrow 0 + 0.75 * 0.75 + 0 = 0.5625$

$\boldsymbol{\theta}_2 \longrightarrow 0.25^2 + 0.5^2 + 0.25^2 = 0.375$ Minimization

$\mathbf{x} = [1, 2, 1]$ $\boldsymbol{\theta}_1 = [0, 0.75, 0]$ $\boldsymbol{\theta}_2 = [0.25, 0.5, 0.25]$

Regularization: example

Loss $J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta})$

L1 regularization $R(\boldsymbol{\theta}) = \sum_{i=1}^{n} |\theta_i|$

$\boldsymbol{\theta}_1 \longrightarrow \quad 0 + 0.75 + 0 = 0.75$ Minimization

$\boldsymbol{\theta}_2 \longrightarrow \quad 0.25 + 0.5 + 0.25 = 1$

$\mathbf{x} = [1, 2, 1] \qquad \boldsymbol{\theta}_1 = [0, 0.75, 0] \qquad \boldsymbol{\theta}_2 = [0.25, 0.5, 0.25]$

Regularization: example

- Input: 3 features $\mathbf{x} = [1, 2, 1]$

- Two linear classifiers that give the same result:

$\boldsymbol{\theta}_1 = [0, 0.75, 0]$ ⟶ Ignores 2 features

$\boldsymbol{\theta}_2 = [0.25, 0.5, 0.25]$ ⟶ Takes information from all features

Regularization: example

- Input: 3 features $\mathbf{x} = [1, 2, 1]$

- Two linear classifiers that give the same result:

$\boldsymbol{\theta}_1 = [0, 0.75, 0]$ → L1 regularization enforces **sparsity**

$\boldsymbol{\theta}_2 = [0.25, 0.5, 0.25]$ → Takes information from all features

Regularization: example

- Input: 3 features $\mathbf{x} = [1, 2, 1]$

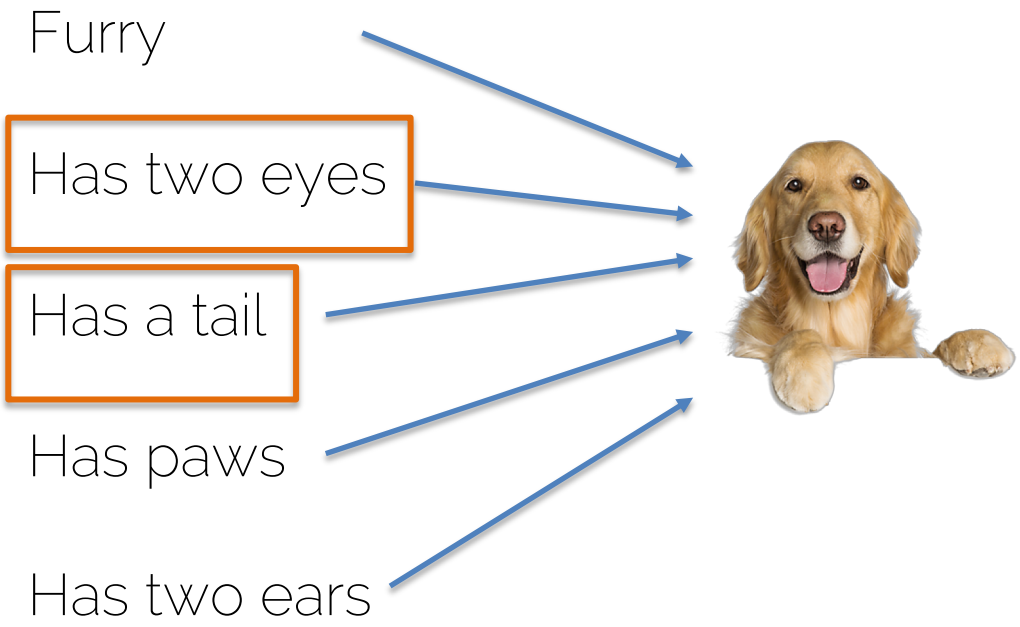
- Two linear classifiers that give the same result:

$\boldsymbol{\theta}_1 = [0, 0.75, 0]$ → L1 regularization enforces **sparsity**

$\boldsymbol{\theta}_2 = [0.25, 0.5, 0.25]$ → L2 regularization enforces that the weights have **similar values**

Regularization: effect

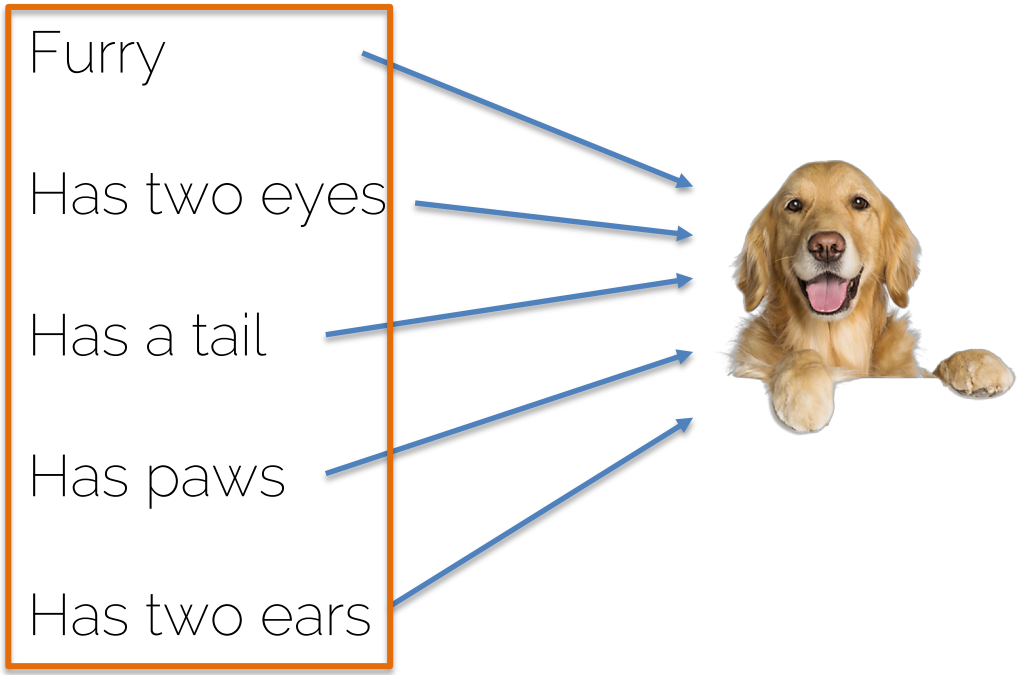
- Dog classifier takes different inputs



L1 regularization will focus all the attention to a few key features

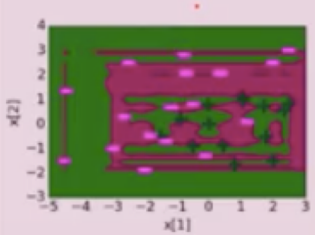
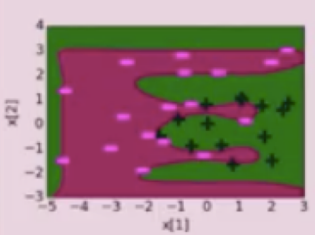
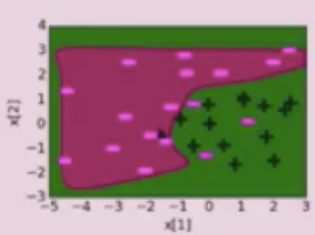
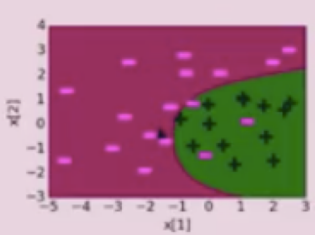
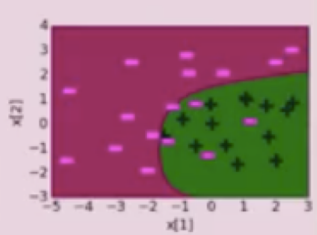
Regularization: effect

- Dog classifier takes different inputs



L2 regularization will take all information into account to make decisions

Regularization

Regularization	λ = 0	λ = 0.00001	λ = 0.001	λ = 1	λ = 10
Decision boundary					

What is the goal of regularization?

What happens to the training error?

Credits: University of Washington

Regularization

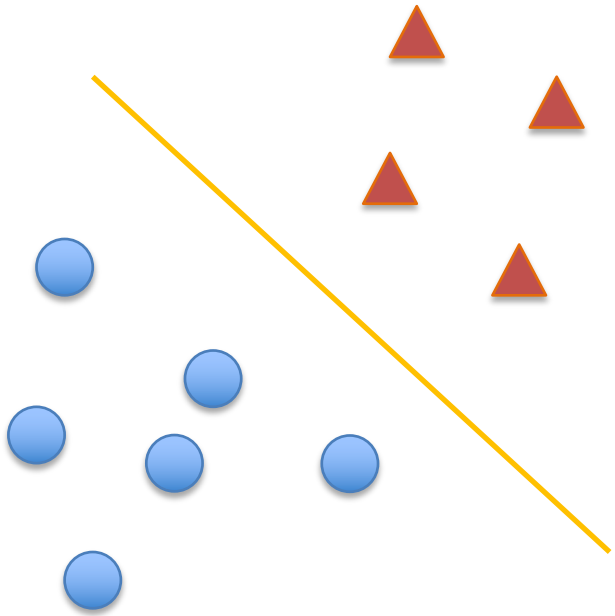
- Any strategy that aims to

Lower validation error

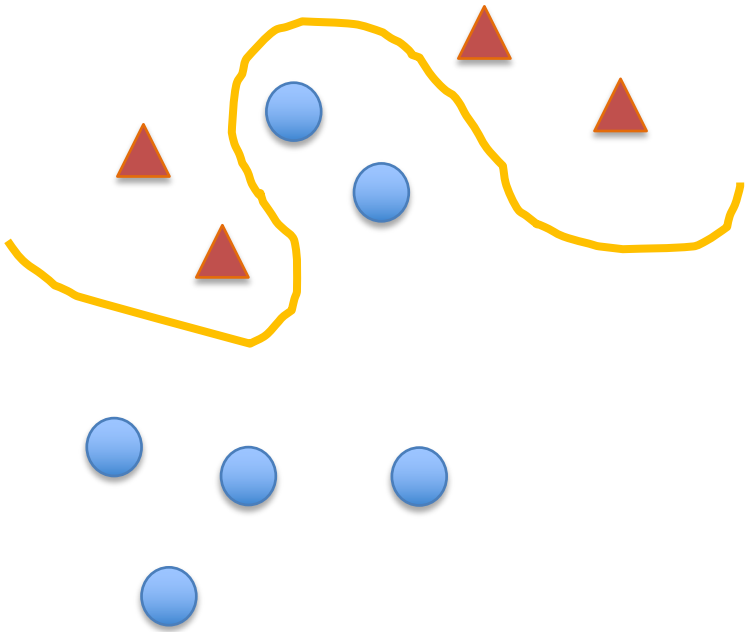
Increasing training error

Beyond linear

This lecture



Next lectures



Next lectures

- Next Thursday: holiday!
- Thursday November 8th: Introduction to Neural Networks
- Next Tuesday: Math refresh #2, particularly useful for the jump to NN!