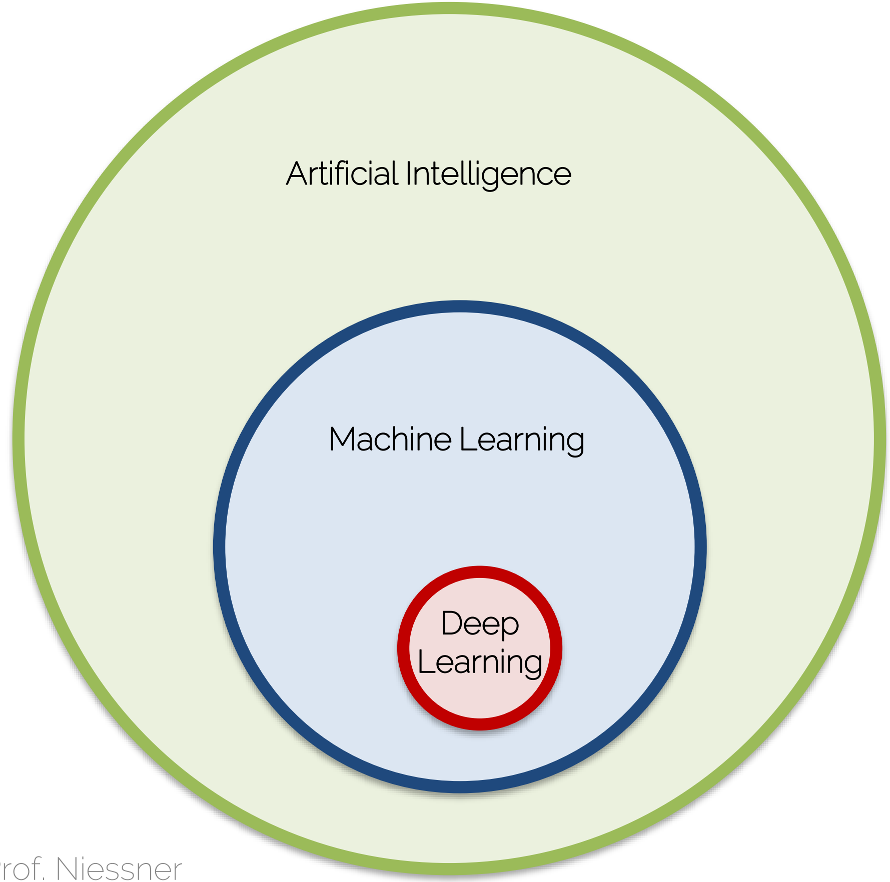
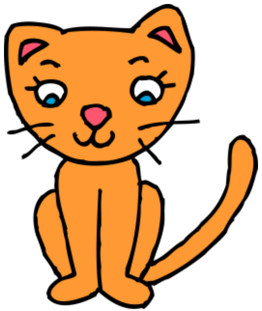
Machine Learning Basics

AI vs ML vs DL



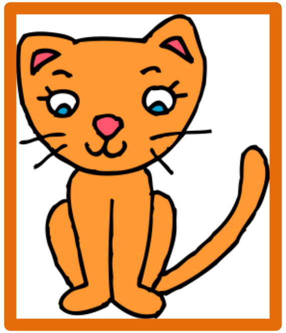
A Simple Task: Image Classification

Image Classification



Task

Image Classification







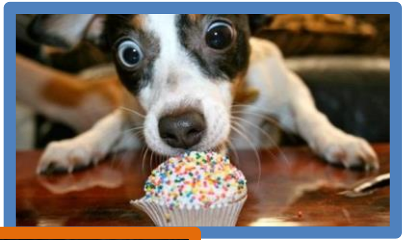
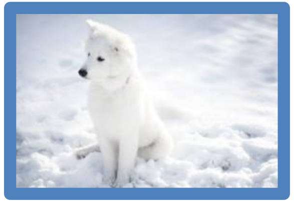
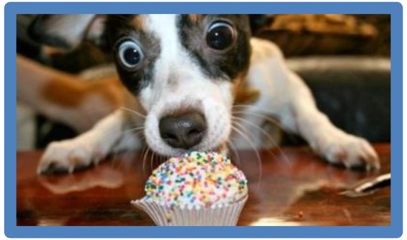



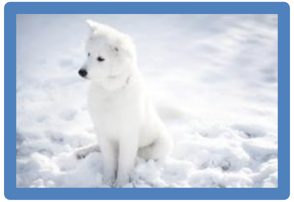
Image Classification

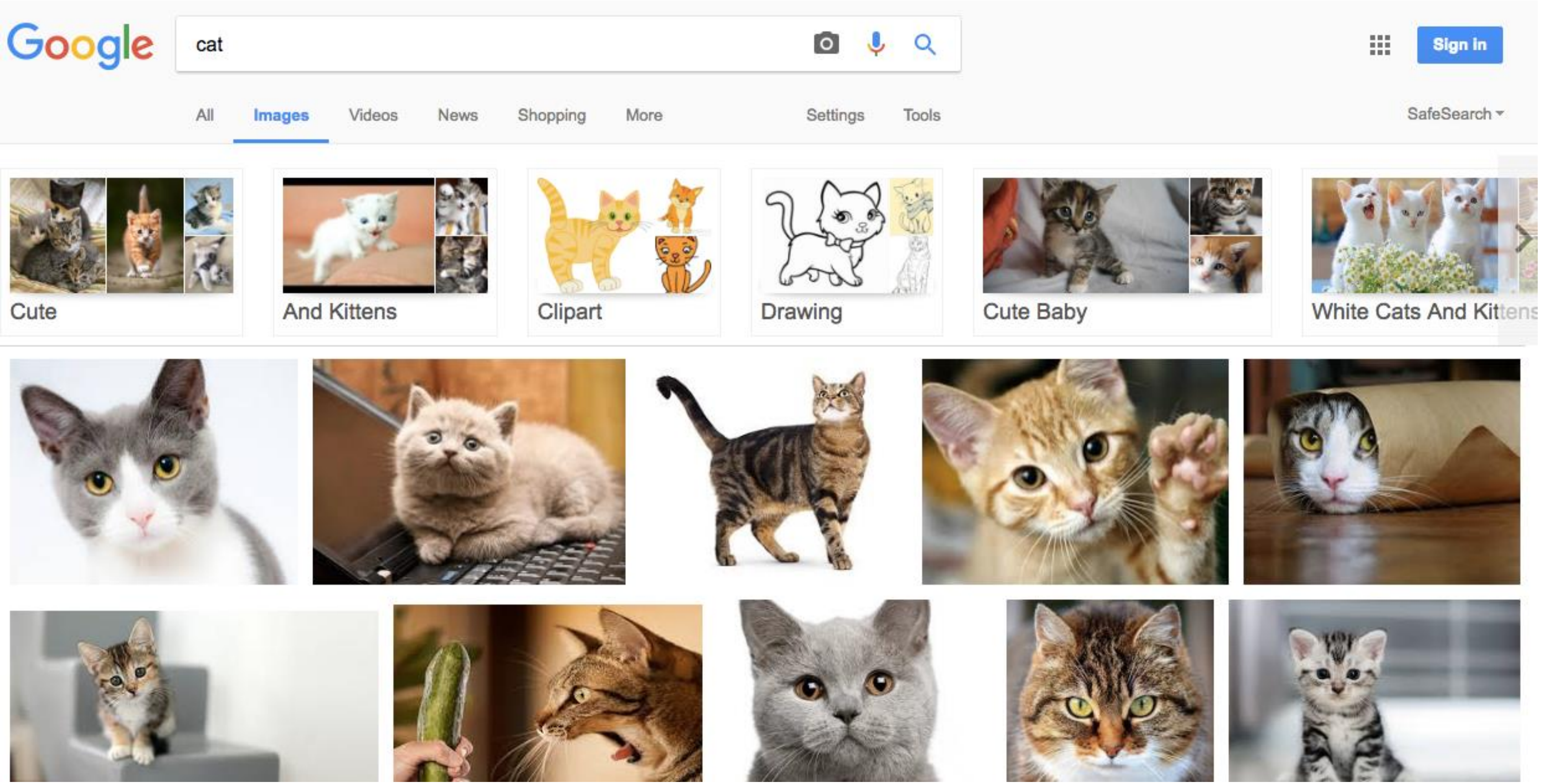


Occlusions

Image Classification

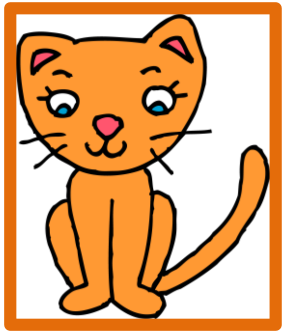
Background clutter





Pose Illumination Appearance

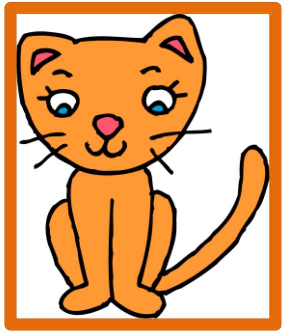
Image Classification

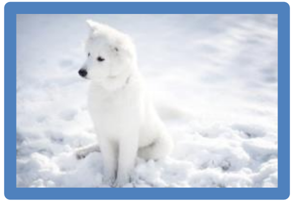


Representation

A Simple Classifier

Nearest Neighbor





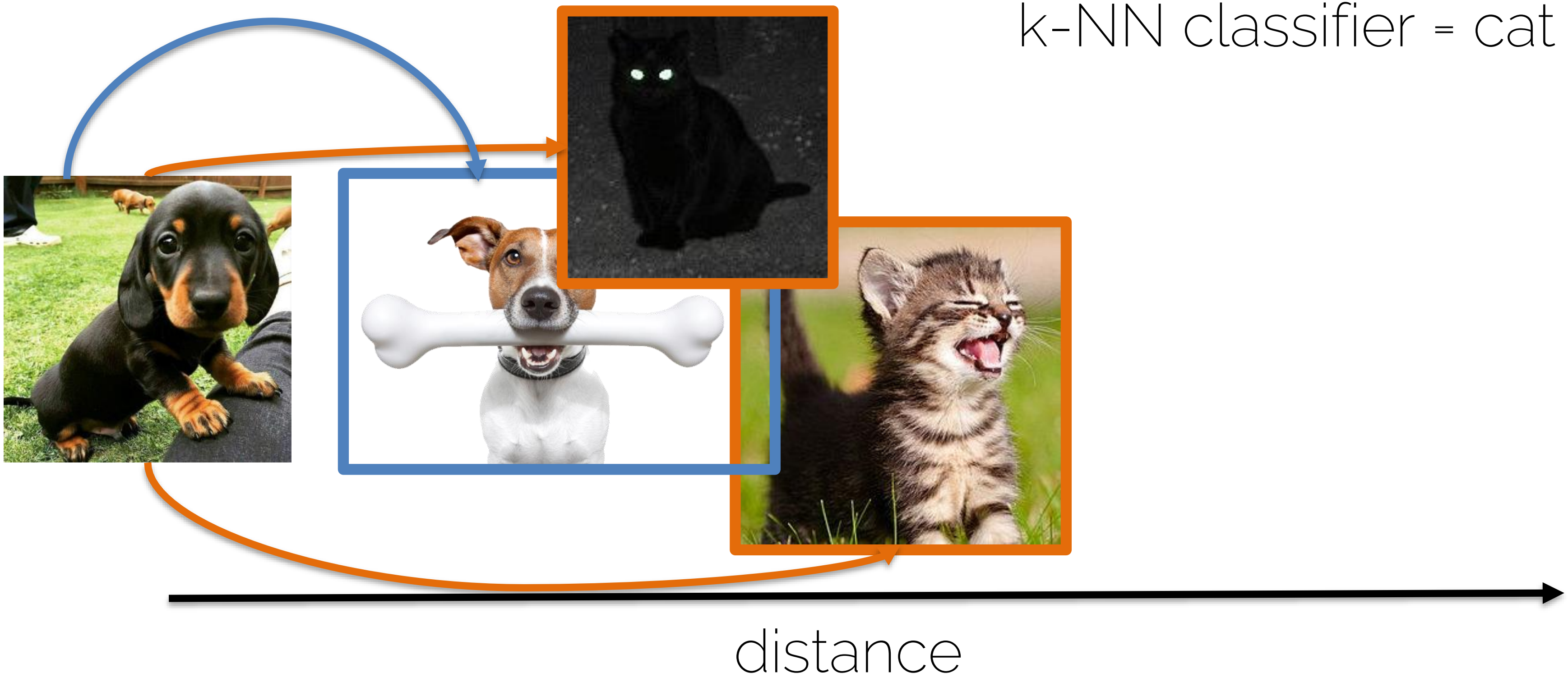
Nearest Neighbor

NN classifier = dog

distance

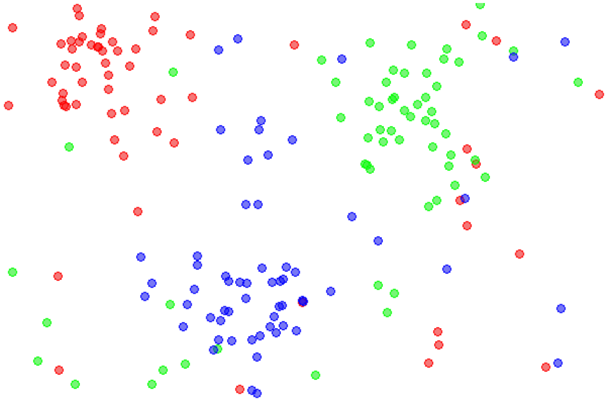
Nearest Neighbor

k-NN classifier = cat

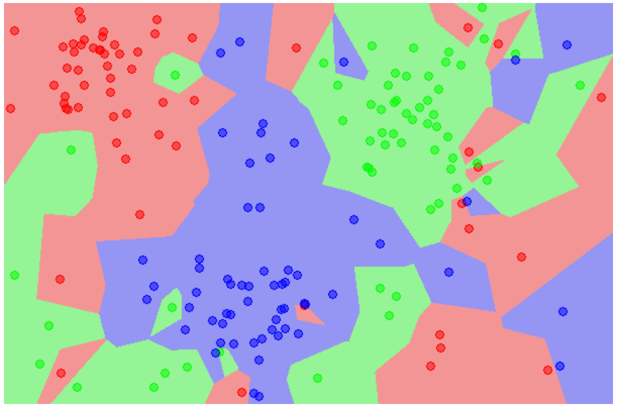


Nearest Neighbor

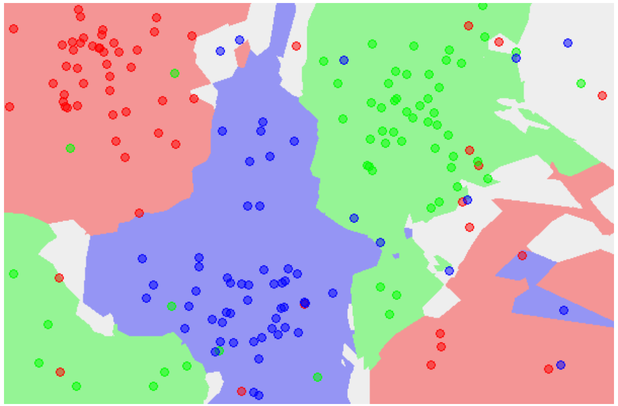
The Data



NN Classifier



5NN Classifier



How does the NN classifier perform on training data?

What classifier is more likely to perform best on test data?

What are we actually learning?

Source: https://commons.wikimedia.org/wiki/File:Data3classes.png

Nearest Neighbor

- Hyperparameters
  - L1 distance : $|x - c|$
  - L2 distance : $||x - c||_2$
  - No. of Neighbors: k

- These parameters are problem dependent.

- How do we choose these hyperparameters?

Machine Learning for Classification

Machine Learning

- How can we learn to perform image classification?

Task

Image classification

Experience

Data

Machine Learning

- $M_\theta(I) = \{\text{DOG}, \text{CAT}\}$

Model

Model Params

Image

Class Label

CAT

DOG

DOG

CAT

CAT

DOG

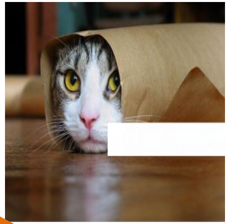
Machine Learning

- $M_\theta(I) = \{\text{DOG}, \text{CAT}\}$

Model — Model Params — Image — Class Label

Given $\boldsymbol{i}$ images with train labels

CAT



DOG

DOG

CAT

CAT

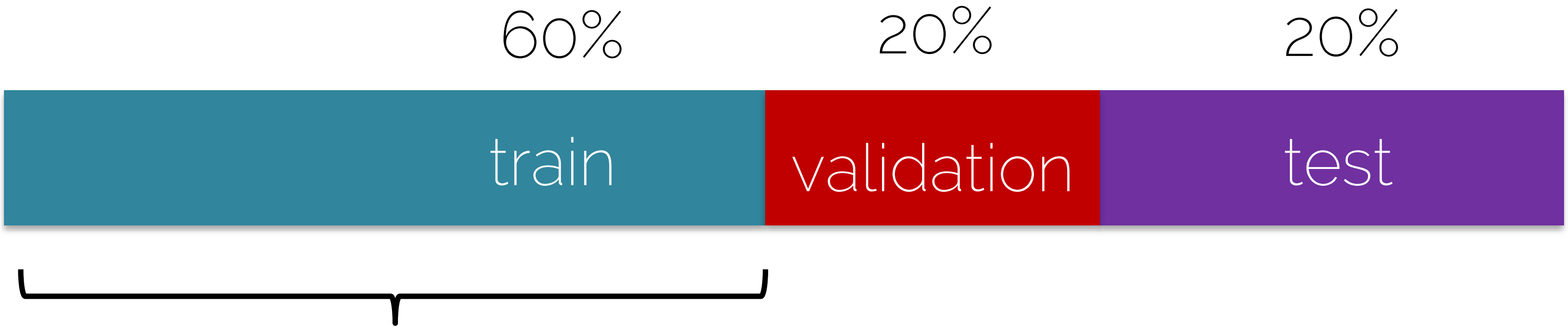
DOG

$$\theta^* = \underset{\theta}{\text{argmin}} \sum_i \text{D}(M_\theta(I_i) - Y_i)$$

"Distance" function — $\{\text{DOG}, \text{CAT}\}$

Basic Recipe for Machine Learning

- Split your data

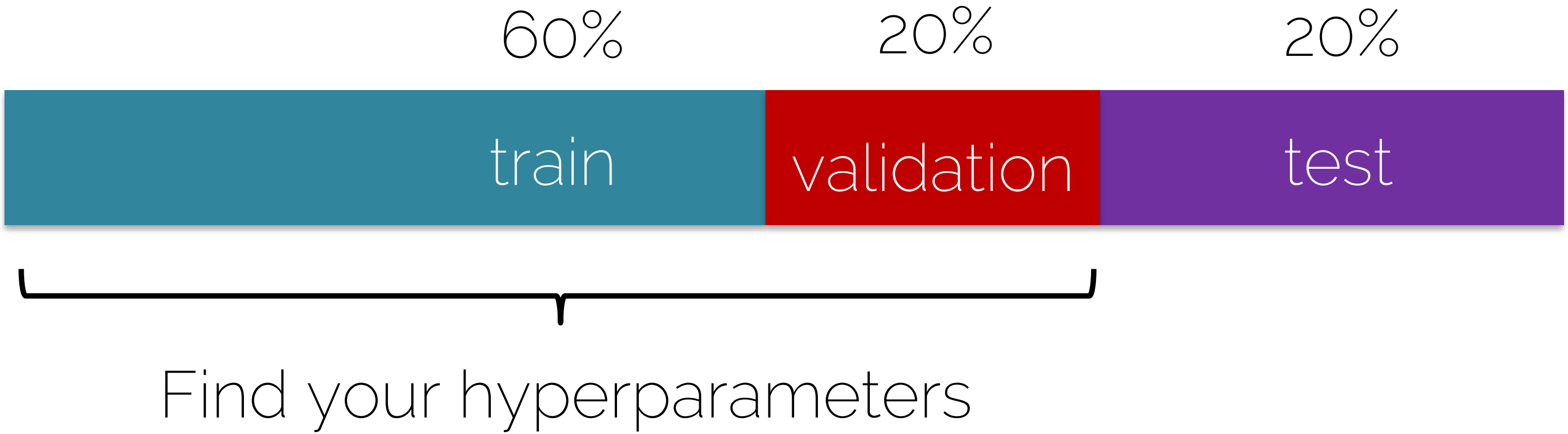


Find model params $\boldsymbol{\theta}$

Other splits are also possible (e.g., 80%/10%/10%)

Basic Recipe for Machine Learning

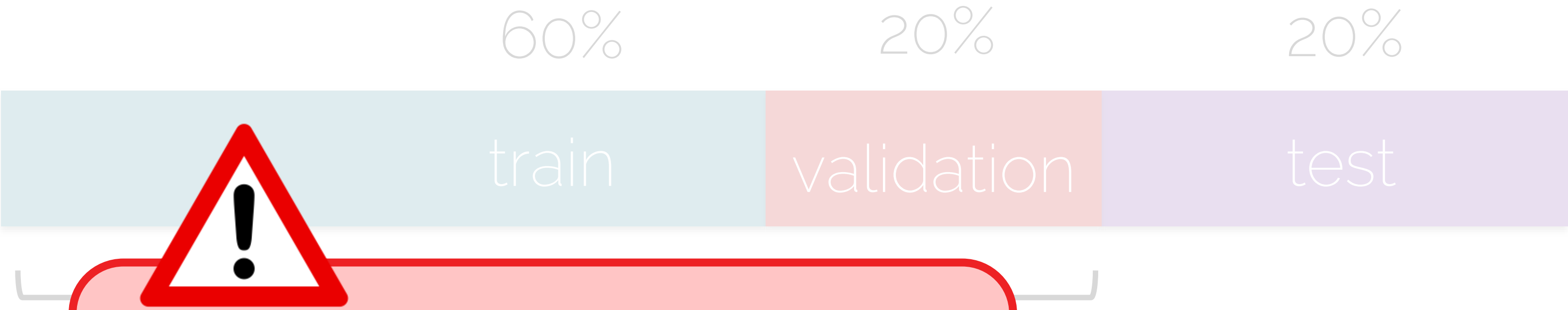
- Split your data



Other splits are also possible (e.g., 80%/10%/10%)

Basic Recipe for Machine Learning

- Split your data



Test set is only used once!

Machine Learning

- How can we learn to perform image classification?

Task

Image classification

Performance measure

Accuracy

Experience

Data

Machine Learning

Unsupervised learning

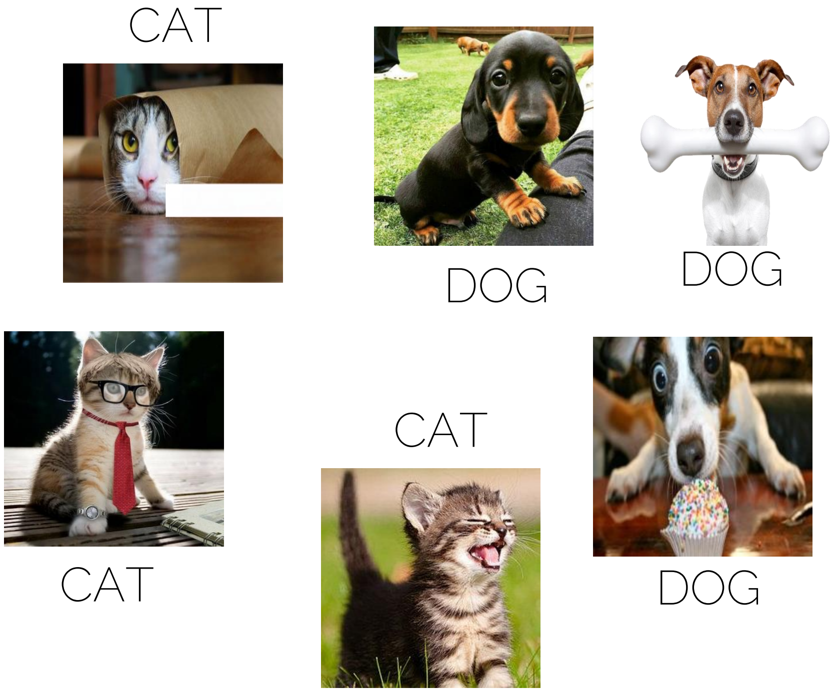
Supervised learning

- Labels or target classes

Machine Learning

Unsupervised learning

Supervised learning

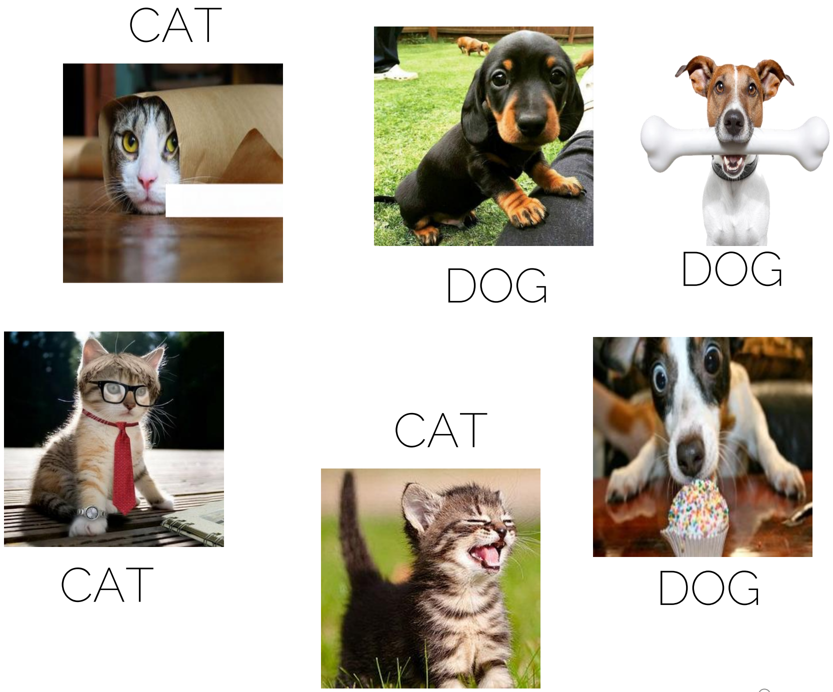


Machine Learning

Unsupervised learning

- No label or target class
- Find out properties of the structure of the data
- Clustering (k-means, PCA, etc.)

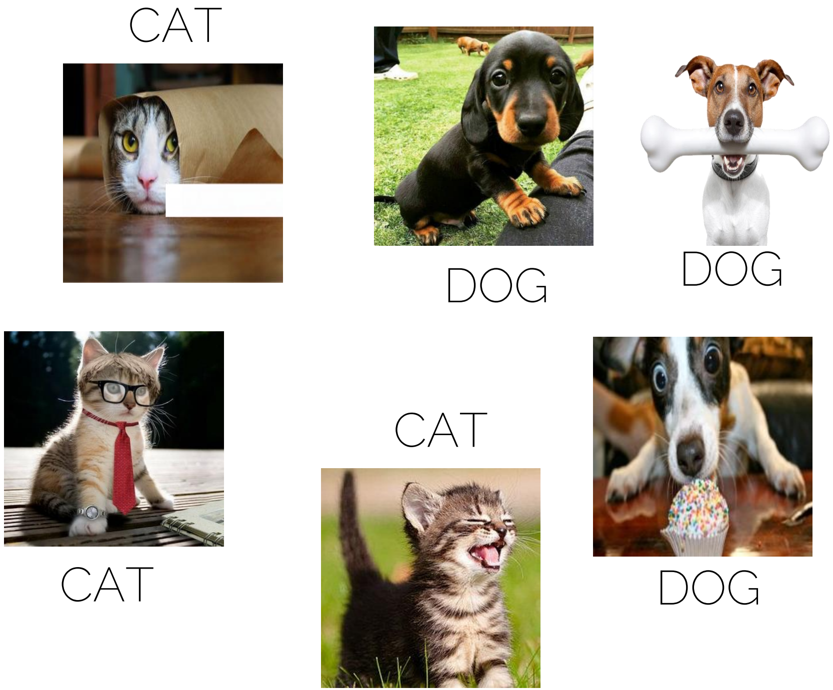
Supervised learning



Machine Learning

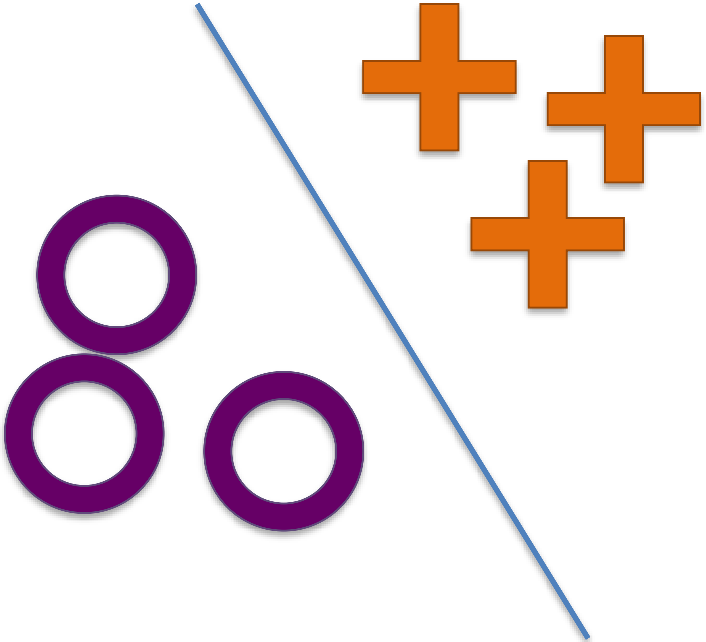
Unsupervised learning

Supervised learning

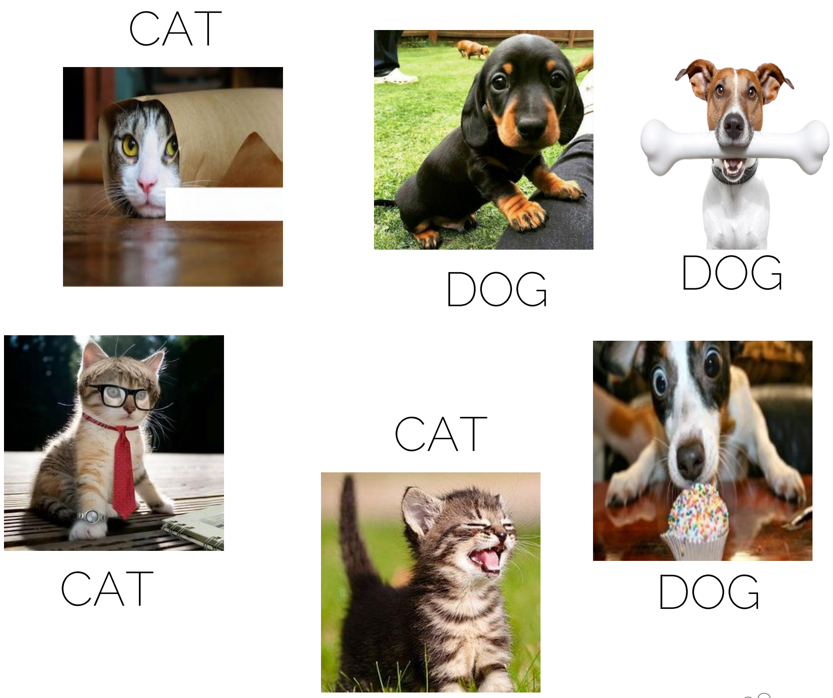


Machine Learning

Unsupervised learning



Supervised learning



Machine Learning

Unsupervised learning

Supervised learning

Reinforcement learning

Machine Learning

Unsupervised learning

Supervised learning

Reinforcement learning

Machine Learning

Unsupervised learning

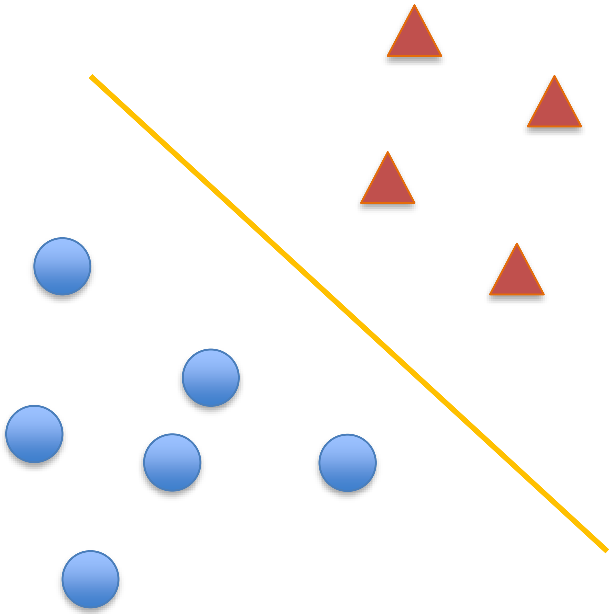
Supervised learning

Reinforcement learning

Agents ← reward ← Environment

Linear Decision Boundaries

Let's start with a simple linear model!

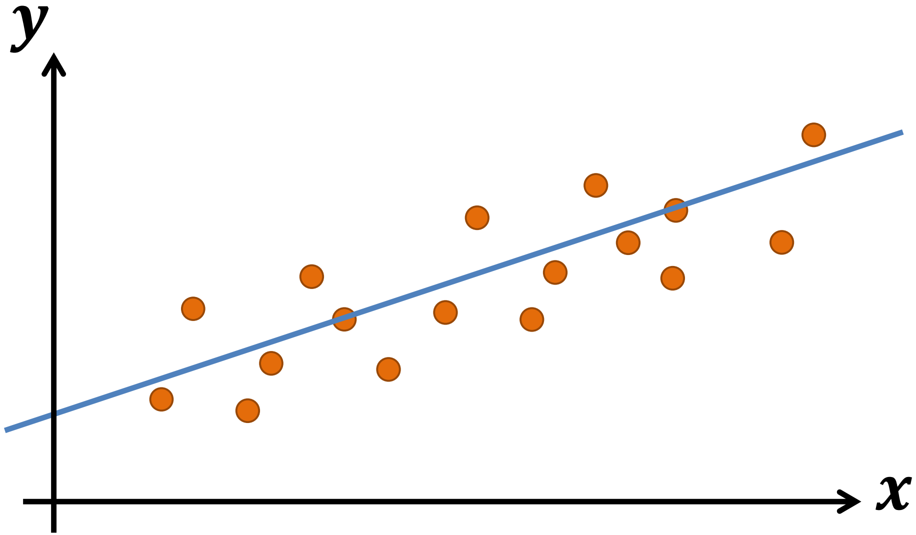


What are the pros and cons for using linear decision boundaries?

Linear Regression

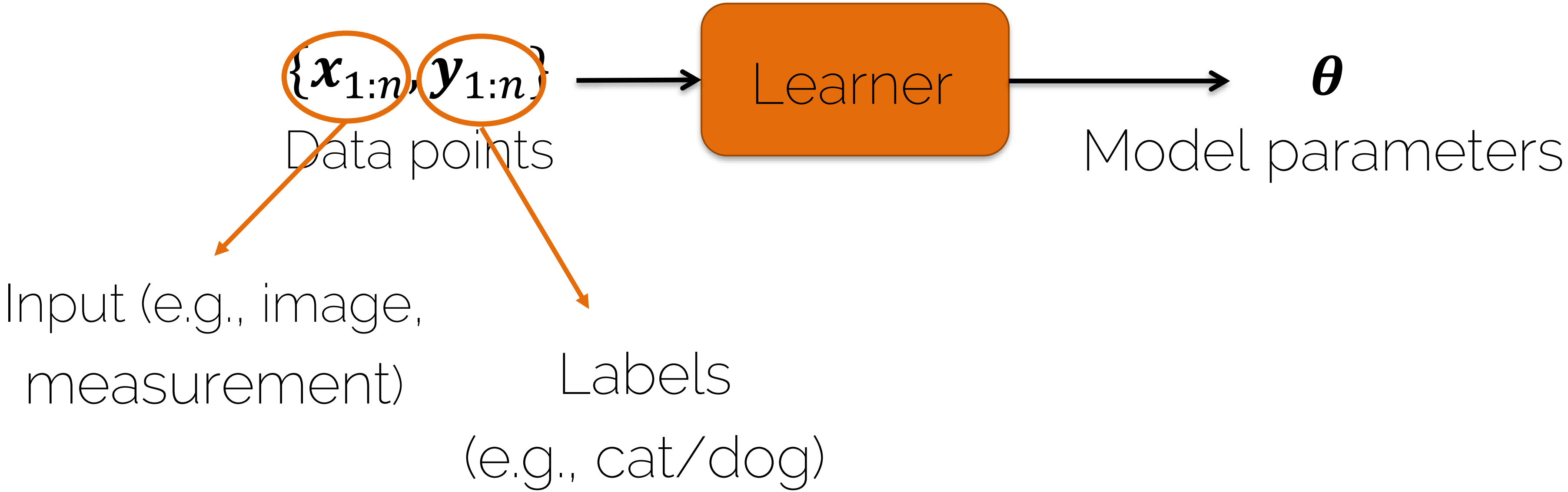
Linear Regression

- Supervised learning
- Find a linear model that explains a target $\boldsymbol{y}$ given inputs $\boldsymbol{x}$



Linear Regression

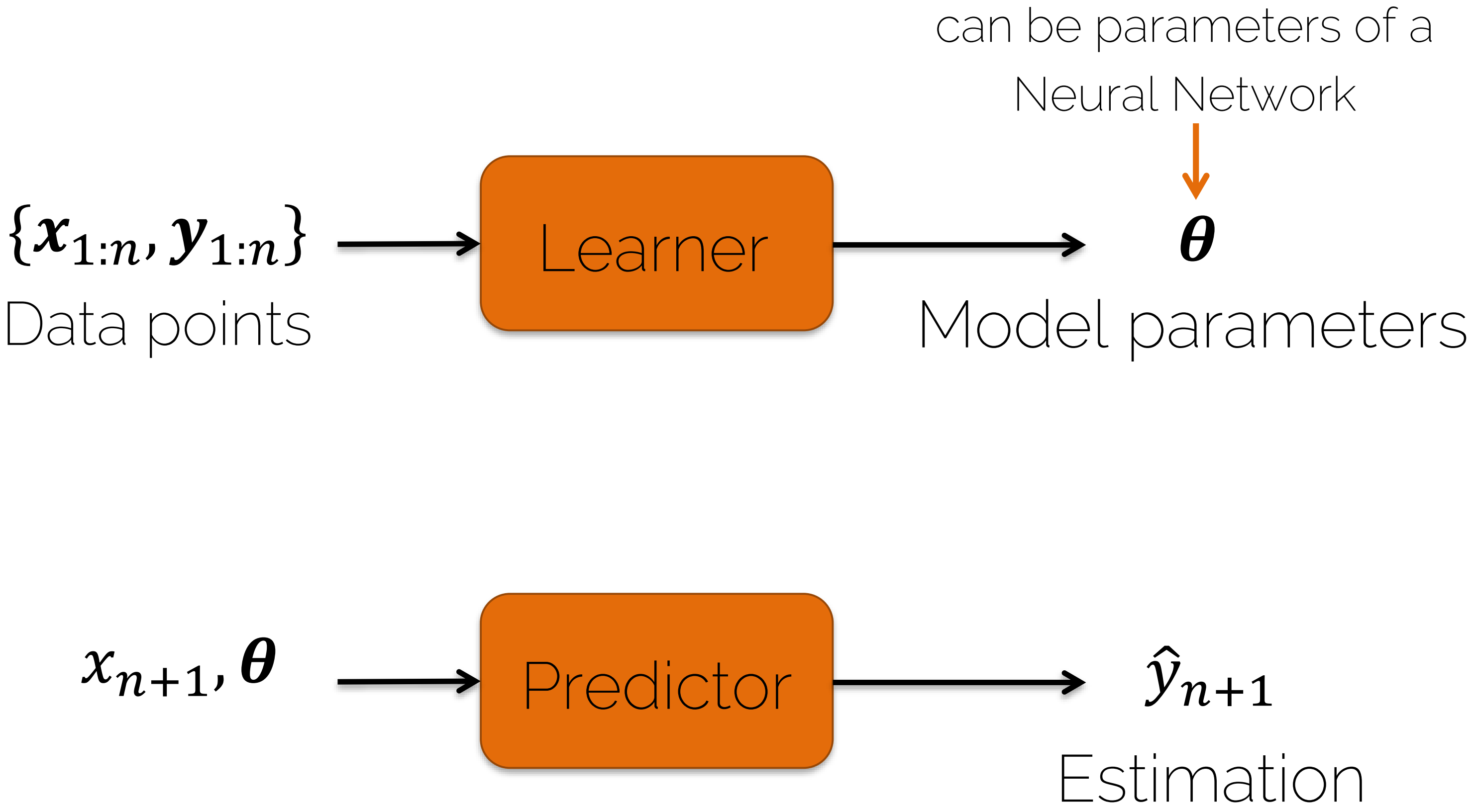
Training



Linear Regression

Training

Testing



Linear Prediction

- A linear model is expressed in the form

$$\hat{y}_i = \sum_{j=1}^{d} x_{ij}\theta_j$$

input dimension

weights (i.e., model parameters)

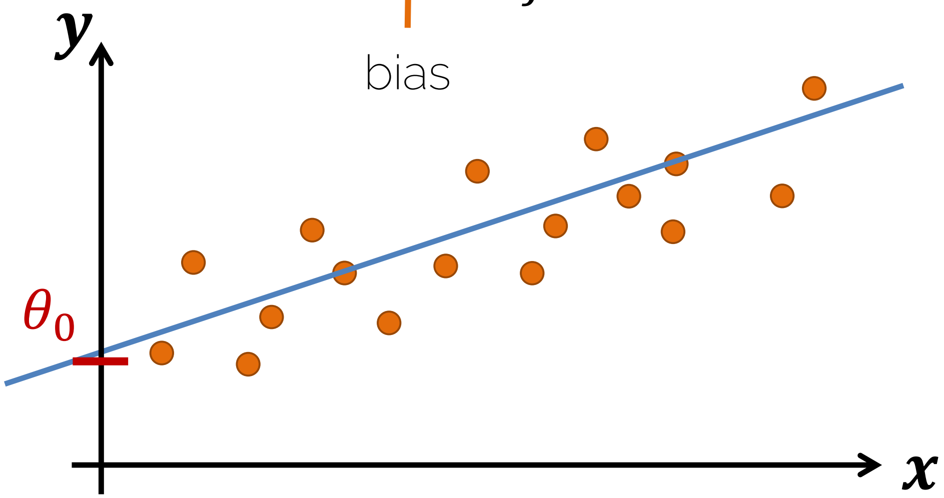
Input data, features

Linear Prediction

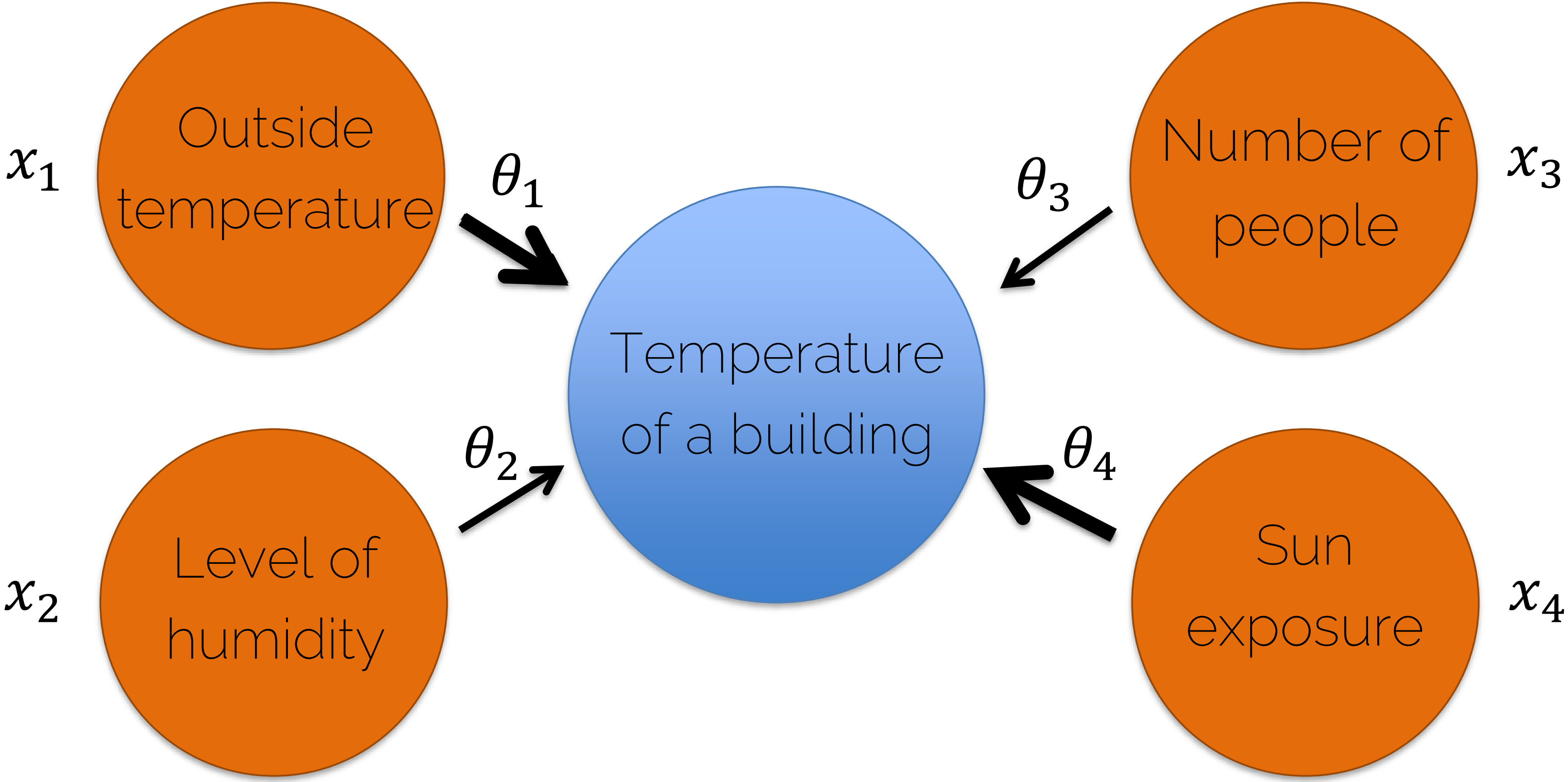
- A linear model is expressed in the form

$$\hat{y}_i = \theta_0 + \sum_{j=1}^{d} x_{ij}\theta_j = \theta_0 + x_{i1}\theta_1 + x_{i2}\theta_2 + \cdots + x_{id}\theta_d$$

bias



Linear Prediction



Linear Prediction

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \theta_0 + \begin{bmatrix} x_{11} & \cdots & x_{1d} \\ x_{21} & \cdots & x_{2d} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nd} \end{bmatrix} \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix}$$

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nd} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \Rightarrow \quad \hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$$

Linear Prediction

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$$

Prediction

Input features

(one sample has $\boldsymbol{d}$ features)

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nd} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$$

Model parameters

($\boldsymbol{d}$ weights and $\mathbf{1}$ bias)

Linear Prediction

Temperature of the building

Bias, Outside temperature, Humidity, Number people, Sun exposure (%)

MODEL

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} 1 & 25 & 50 & 2 & 50 \\ 1 & -10 & 50 & 0 & 10 \end{bmatrix} \cdot \begin{bmatrix} 0.2 \\ 0.64 \\ 0 \\ 1 \\ 0.14 \end{bmatrix}$$

Linear Prediction

Temperature of the building

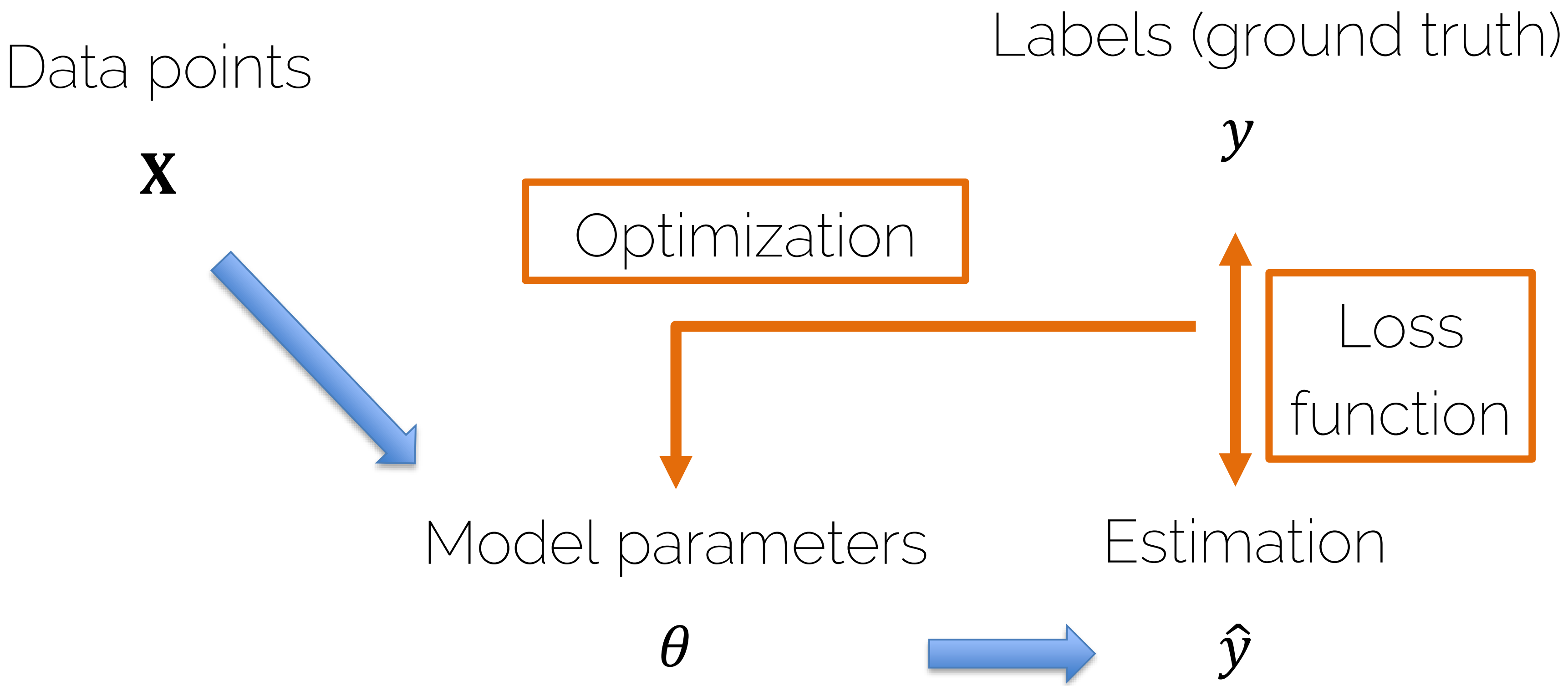
Bias | Outside temperature | Humidity | Number people | Sun exposure (%)

MODEL

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} 1 & 25 & 50 & 2 & 50 \\ 1 & -10 & 50 & 0 & 10 \end{bmatrix} \cdot \begin{bmatrix} 0.2 \\ 0.64 \\ 0 \\ 1 \\ 0.14 \end{bmatrix}$$

How do we obtain the model?

How to Obtain the Model?

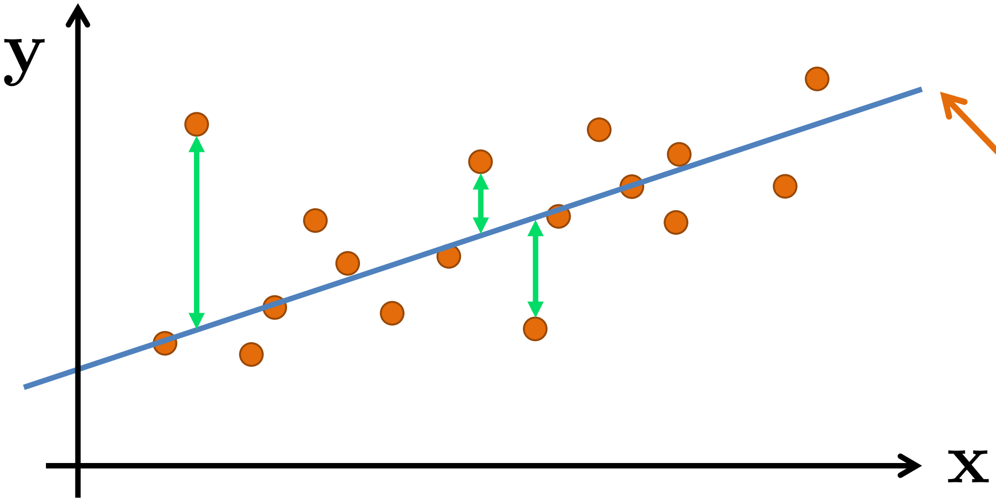


How to Obtain the Model?

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

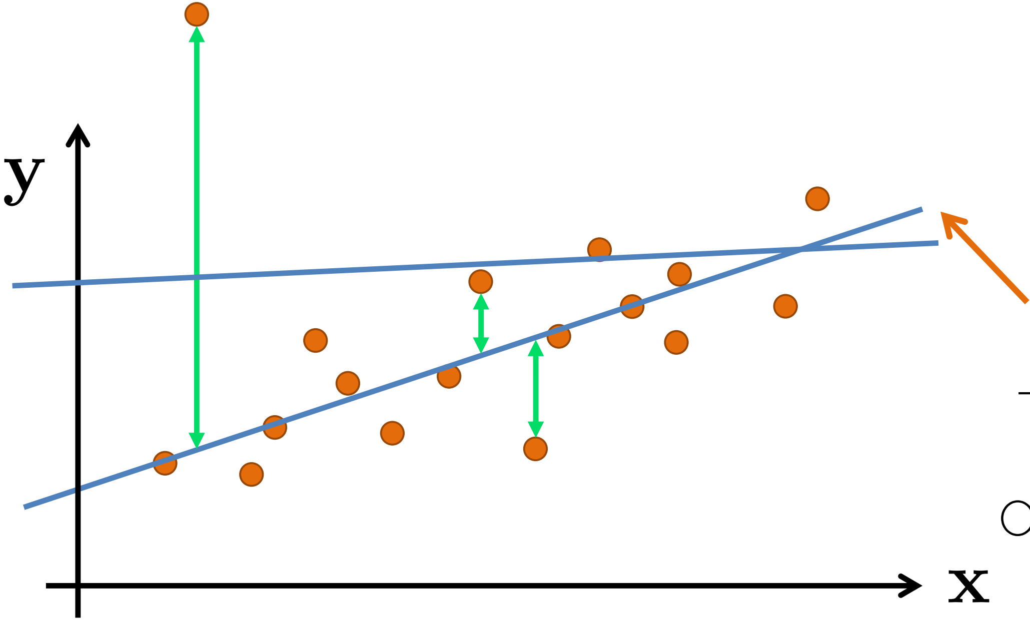
- **Optimization:** changes the model in order to improve the loss function (i.e., to improve my estimation).

Linear Regression: Loss Function



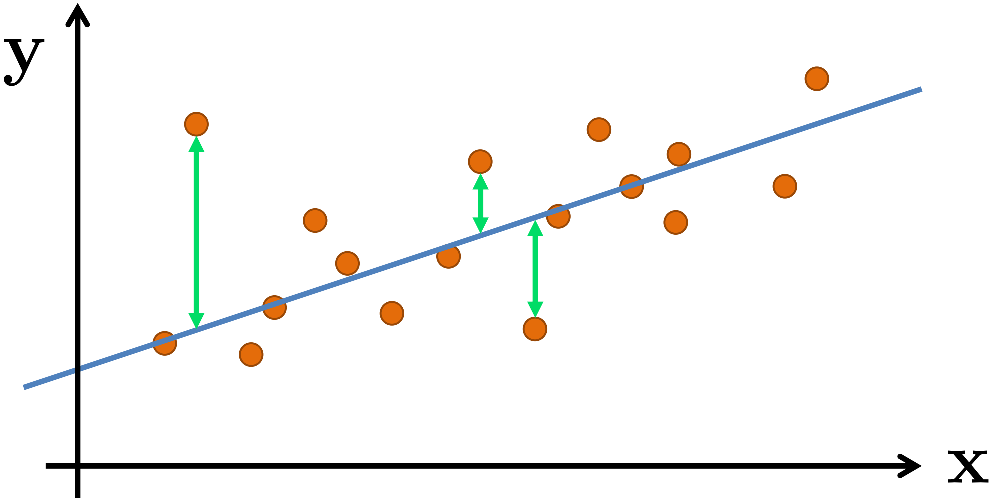
Prediction:
Temperature
of the building

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 linear model to the data

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

- 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

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

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

Matrix notation

$\boldsymbol{n}$ training samples, each input vector has size $\boldsymbol{d}$

$\boldsymbol{n}$ labels

Optimization: Linear Least Squares

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

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

Matrix notation

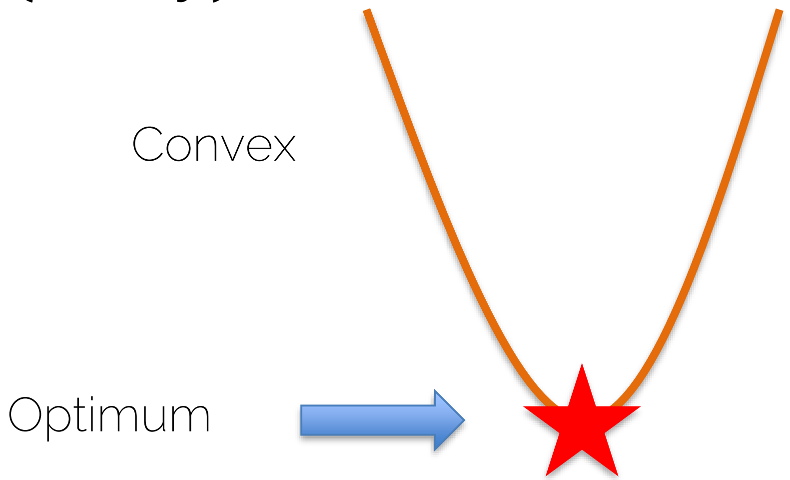
More on matrix notation in the next exercise session

Optimization: Linear Least Squares

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

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

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



Optimization

Details in the exercise session!

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

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

We have found an analytical solution to a convex problem

Inputs: Outside temperature, number of people, ...

True 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 parameter(s)

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.

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

Maximum Likelihood Estimate

- MLE assumes that the training samples are **independent** and **generated by the same probability distribution**

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

"i.i.d." assumption

Maximum Likelihood Estimate

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

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

Logarithmic property $\log ab = \log a + \log b$

Back to Linear Regression

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

What shape does our probability distribution have?

Back to Linear Regression

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

What shape does our probability distribution have?

Back to Linear Regression

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

Gaussian / 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

Gaussian:

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

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

Back to Linear Regression

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

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

mean

Gaussian:

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

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

Back to Linear Regression

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

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

mean

Gaussian:

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

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

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

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

Back to Linear Regression

$$\sum_{i=1}^{n} \log \left[(2\pi\sigma^2)^{-\frac{1}{2}} e^{-\frac{1}{2\sigma^2}(y_i - \boldsymbol{x_i\theta})^2} \right]$$

Canceling $\log$ and e

$$\sum_{i=1}^{n} -\frac{1}{2}\log\,(2\pi\sigma^2) + \sum_{i=1}^{n}\left(-\frac{1}{2\sigma^2}\right)(y_i - \boldsymbol{x_i\theta})^2$$

Matrix notation

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

Back to Linear Regression

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

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

Details in the exercise session!

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

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

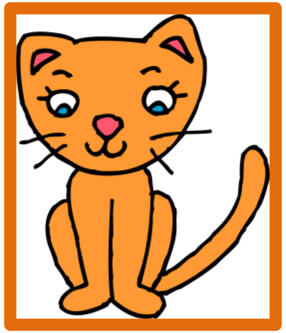
How can we find the estimate of theta?

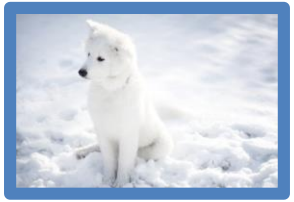
Linear Regression

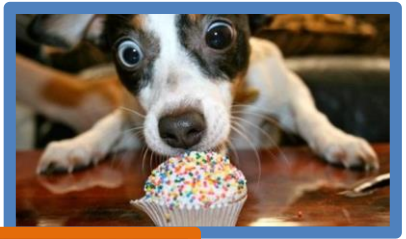
- Maximum Likelihood Estimate (MLE) with a Gaussian assumption leads to the Least Squares Estimation

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

Image Classification





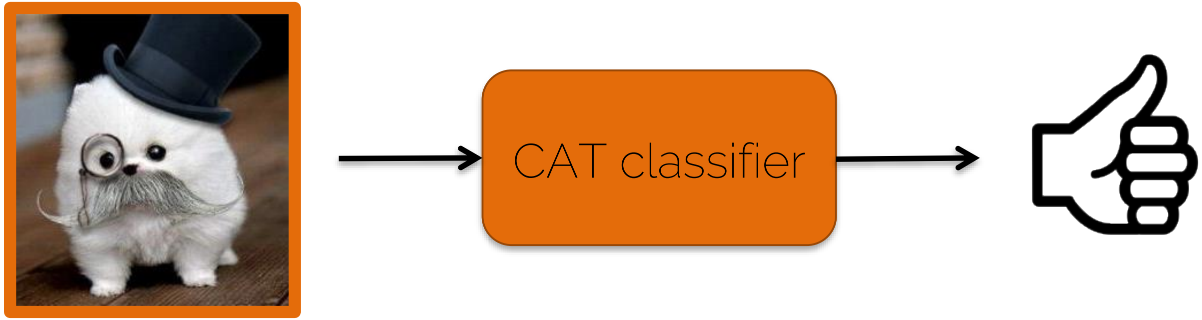


Regression vs Classification

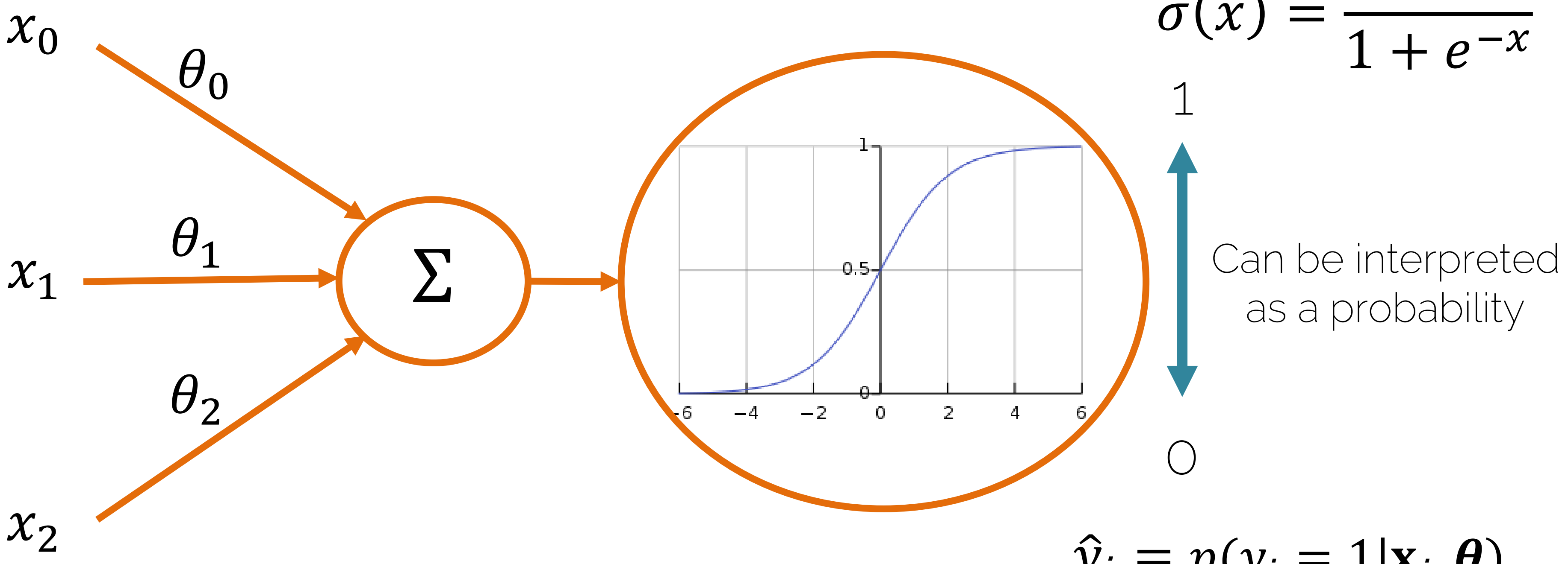
- Regression: predict a continuous output value (e.g., temperature of a room)

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

Logistic Regression



Sigmoid for Binary Predictions

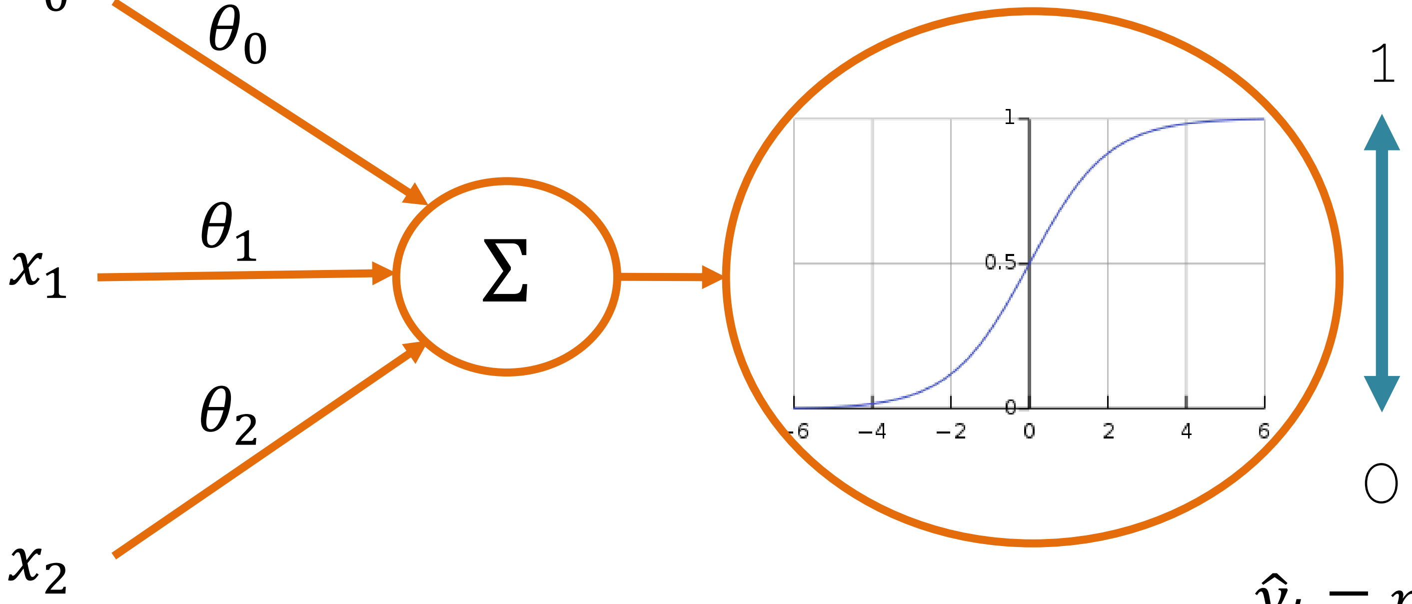


$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

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

Spoiler Alert: 1-Layer Neural Network

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

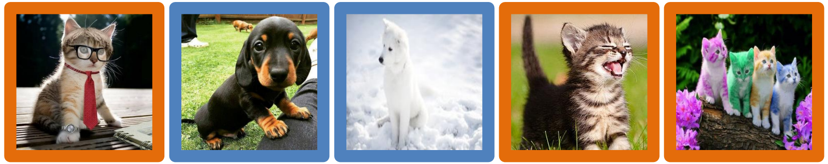


Can be interpreted as a probability

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

Logistic Regression: Max. Likelihood

- Probability of a binary output



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

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

- Maximum Likelihood Estimate

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

Logistic Regression: Loss Function

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

$$\log p(\mathrm{y}|\mathbf{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \log\left(\hat{y}_i^{y_i}(1 - \hat{y}_i)^{(1-y_i)}\right)$$

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

(given that we have two classes)

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 *binary cross-entropy* loss (BCE)

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

Logistic Regression: Optimization

- Loss for each training sample:

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

- Overall loss

$$C(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\hat{y}_i, y_i)$$

Minimization

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

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

Logistic Regression: Optimization

- No closed-form solution

- Make use of an iterative method → gradient descent

Gradient descent –
later on!

Insights from the first lecture

- We can learn from experience
  -> Intelligence, certain ability to infer the future!

- Even linear models are often pretty good for complex phenomena: e.g., weather:
  - Linear combination of day-time, day-year etc. is often pretty good

Next Lectures

- Next exercise session: Math Recap II

- Next Lecture: Lecture 3:
  - Jumping towards our first Neural Networks and Computational Graphs

References for further Reading

- Cross validation:
  - https://medium.com/@zstern/k-fold-cross-validation-explained-5aeba90ebb3
  - https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

- General Machine Learning book:
  - Pattern Recognition and Machine Learning. C. Bishop.

See you next week ☺