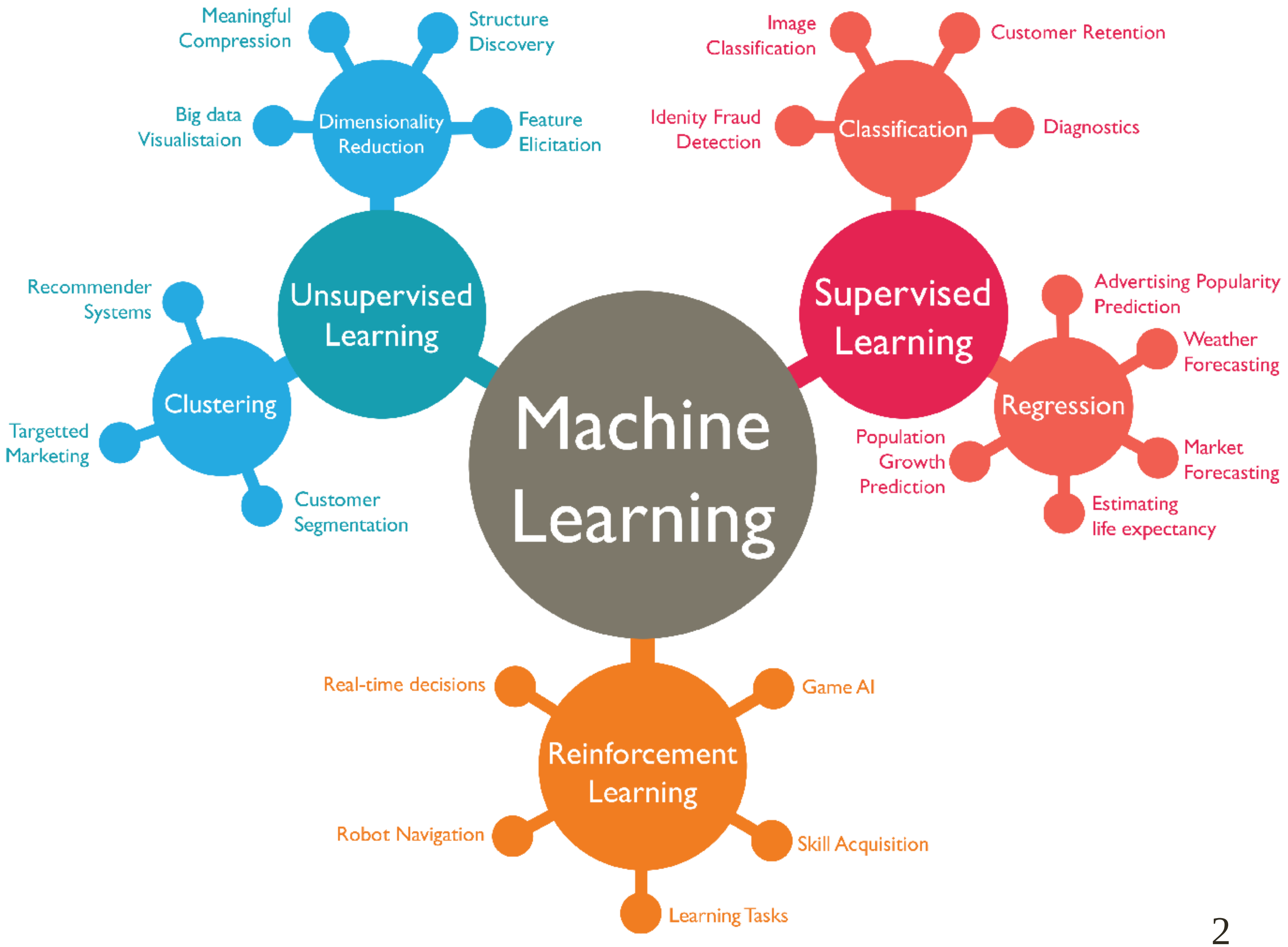
Electrical and Computer Engineering Department
Tarbiat Modares University

Machine Learning Basics

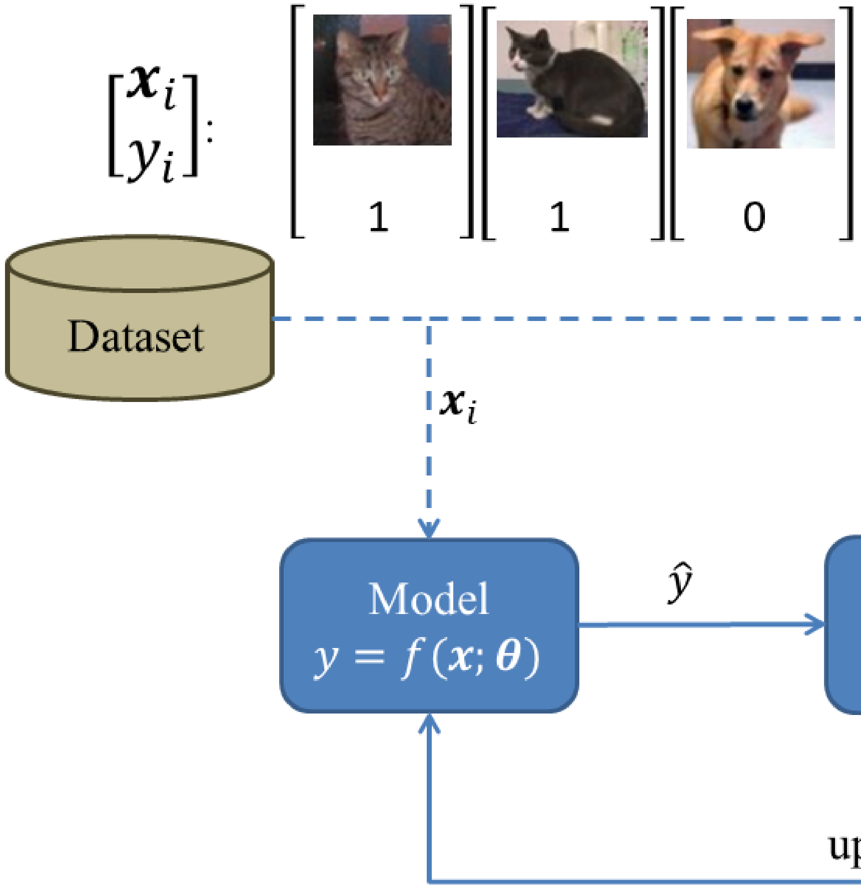
Foad Ghaderi, PhD

Branches of machine learning



Branches of machine learning
Supervised learning

- The most common case
- Learning to map input data to known targets given a set of examples



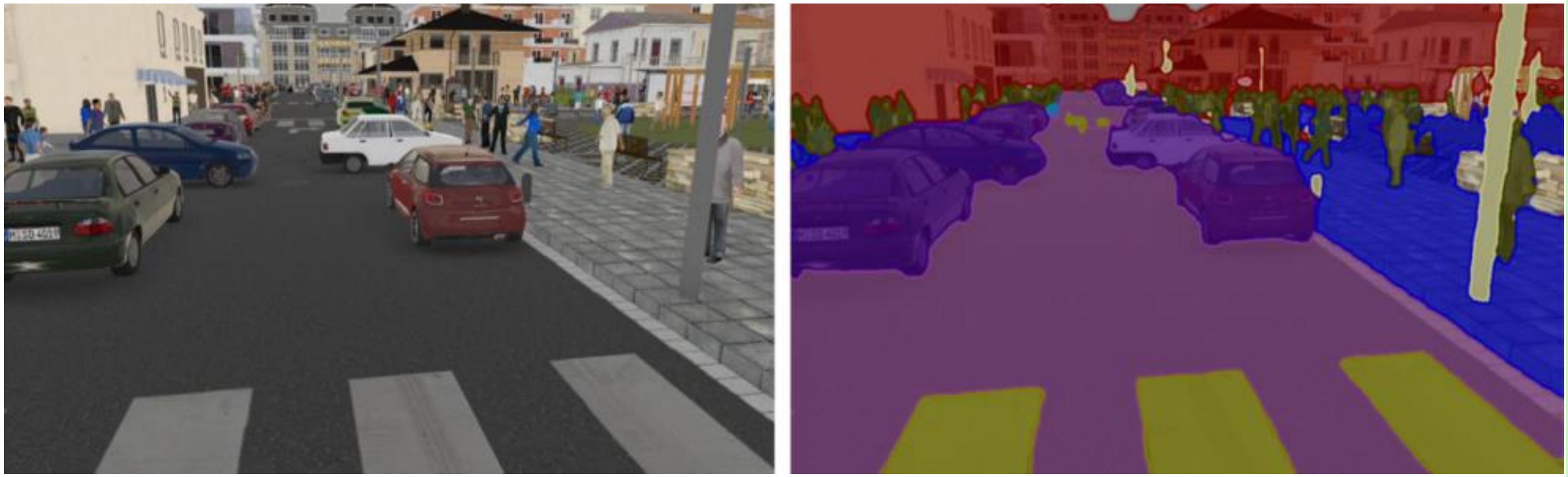
$\begin{bmatrix} \boldsymbol{x}_i \\ y_i \end{bmatrix}$:
1
1
0
Dataset
$\boldsymbol{x}_i$
y_i
Model
$y = f(\boldsymbol{x}; \boldsymbol{\theta})$
$\hat{y}$
Loss function
$\mathcal{L}$
Optimizer
updated $\boldsymbol{\theta}$ value

Example applications:
- ✓ Optical character recognition,
- ✓ Speech recognition,
- ✓ Image classification,
- ✓ Language translation
- ✓ …

Branches of machine learning
Unsupervised learning

Often a necessary step in better understanding a dataset. Finding interesting transformations of the input data without the help of any targets, for the purposes of

- ❑ Data visualization,
- ❑ Data compression,
- ❑ Data denoising, or
- ❑ To better understand the correlations present in the data at



Branches of machine learning
Reinforcement learning

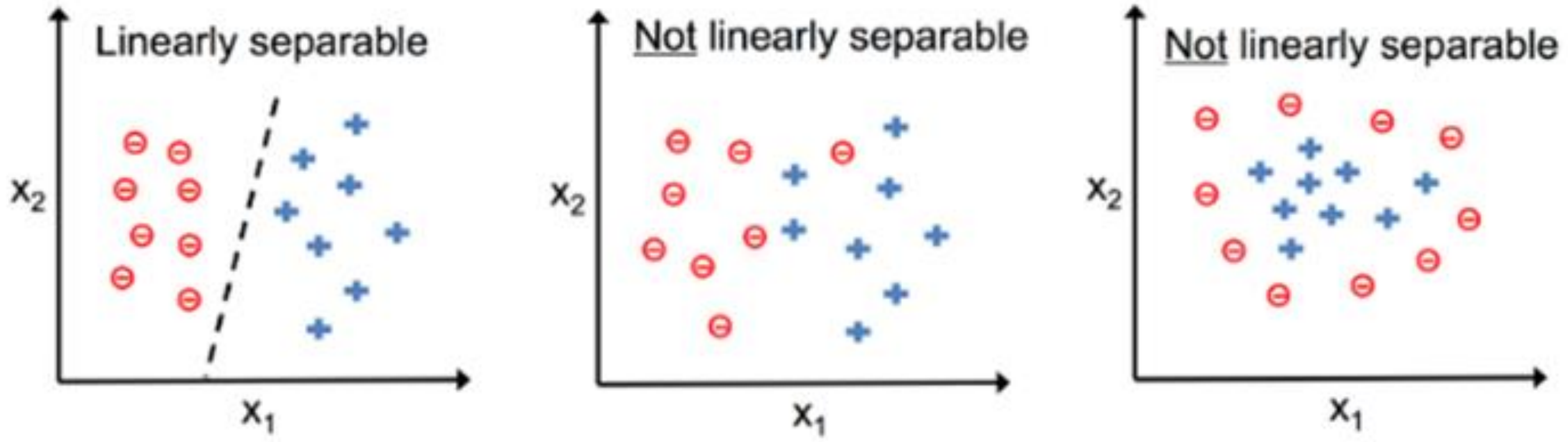
An *agent* receives information about its environment and learns to choose actions that will maximize some reward.

Mostly a research area and hasn't yet had significant practical successes beyond games (Atari, Go).

Possible real world applications:

- self-driving cars,
- robotics,
- resource management,
- education,
- …

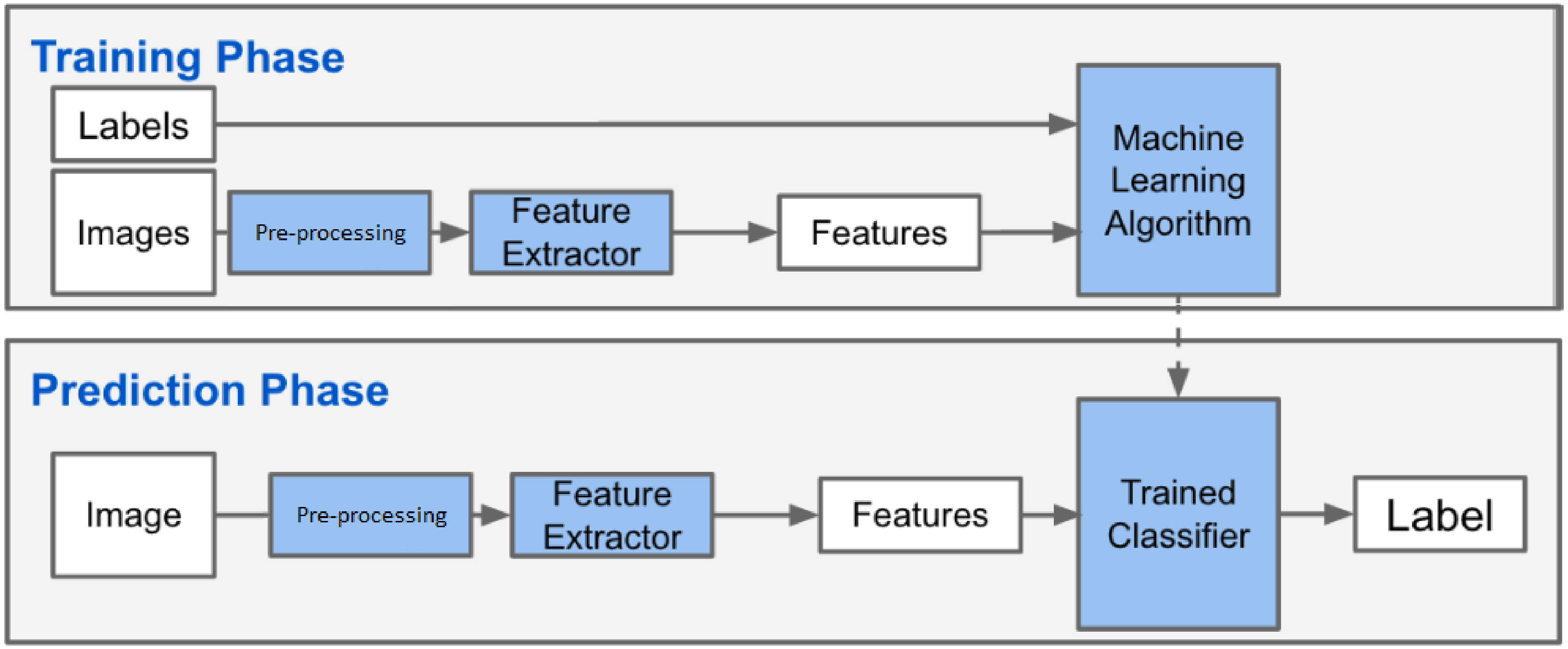
Classification



Classification

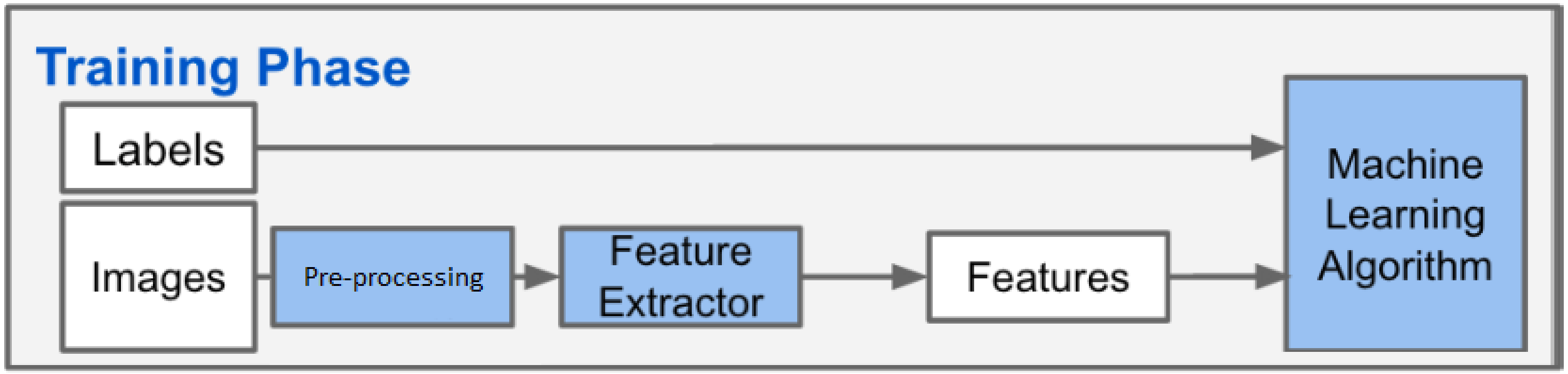
Classification has 2 phases:

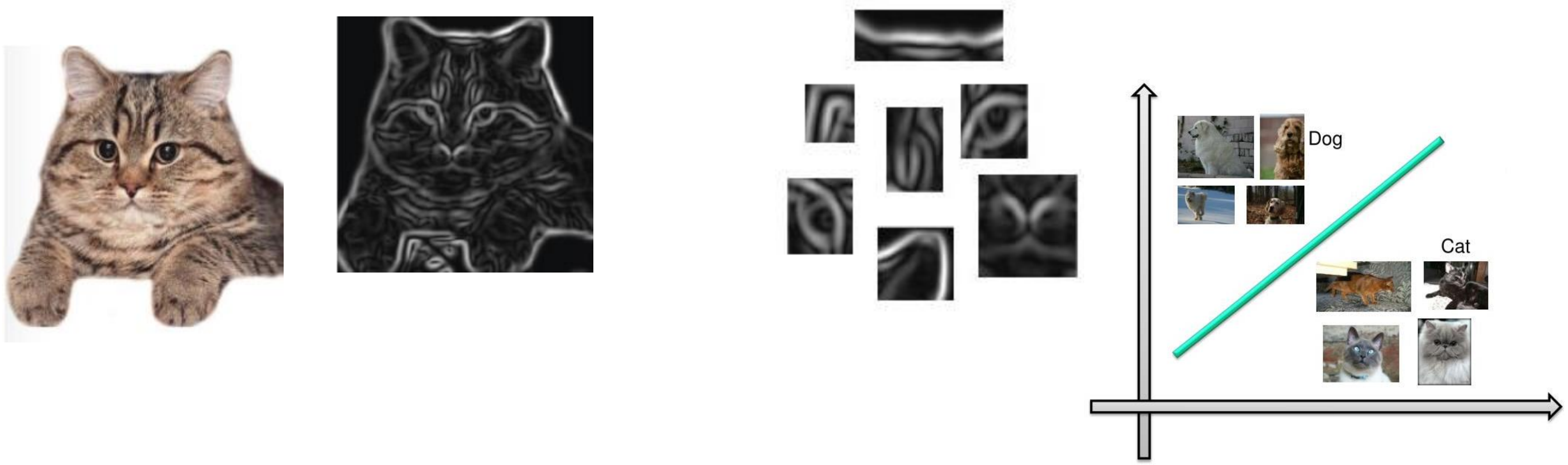
- ❑ **Training phase:**
  A machine learning algorithm is trained using a dataset comprised of the data samples and their corresponding labels.
- ❑ **Prediction phase:**
  The trained model is utilized to predict labels of unseen samples.



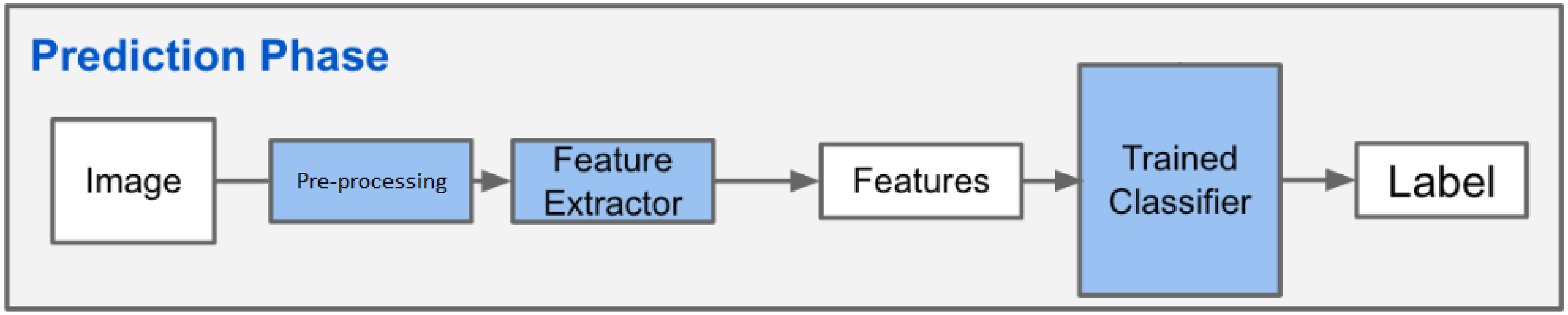
Machine Learning Phases

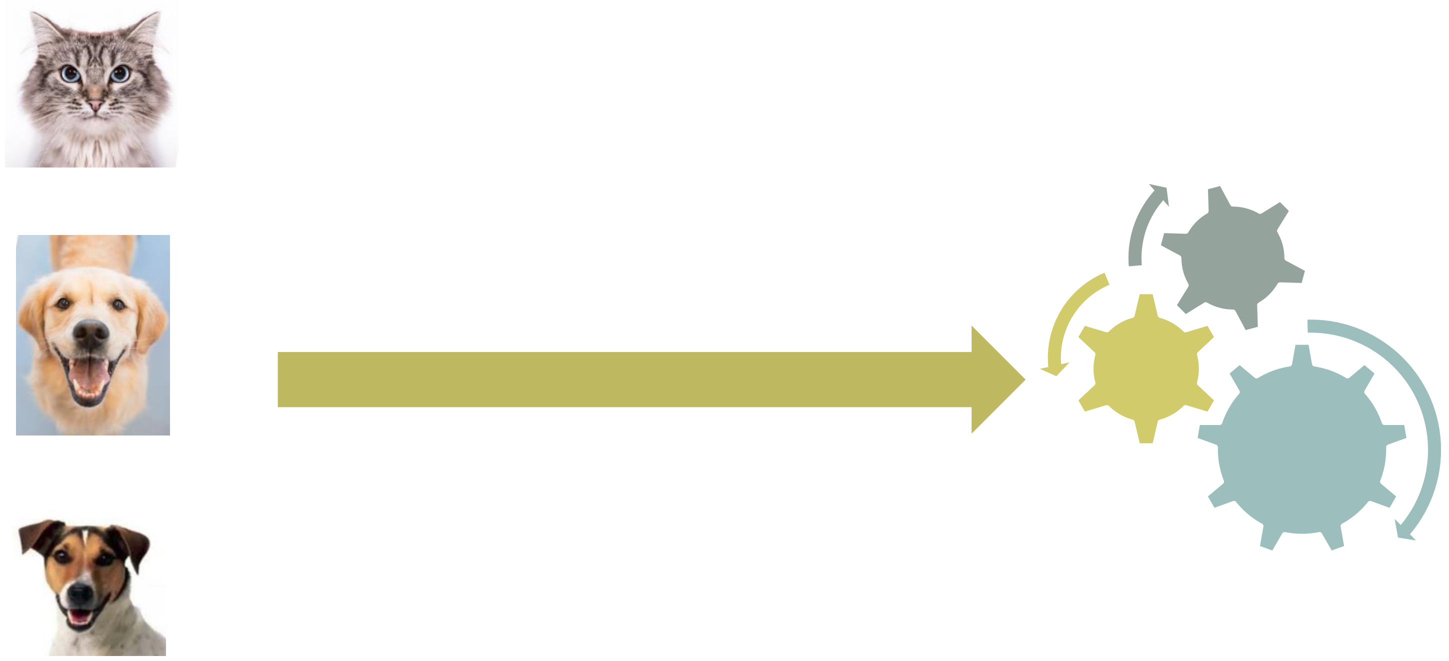
Classification





Classification





Capacity, Overfitting and Underfitting

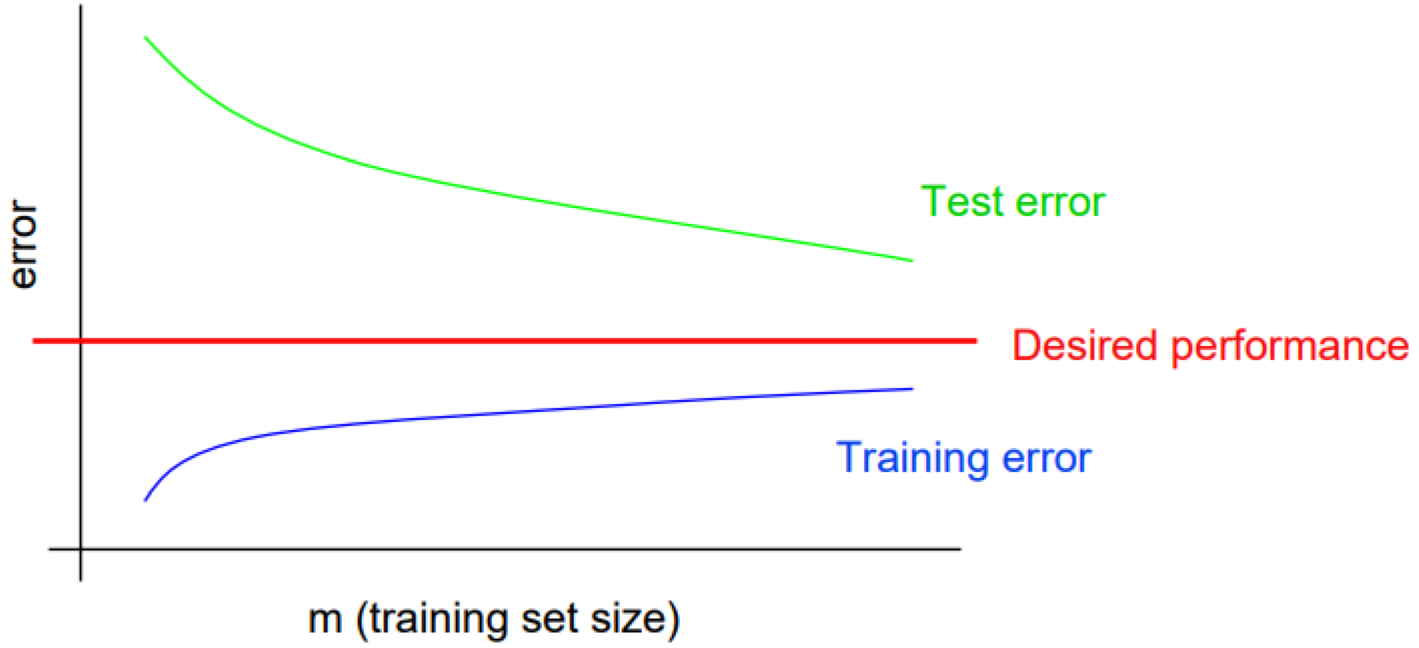
The central challenge in machine learning is that we must perform well on ***new, previously unseen*** inputs—not just those on which our model was trained.

The ability to perform well on previously unobserved inputs is called ***generalization***.

Training error: the measured error on the training set

Generalization (test) error: the expected value of the error on a new input.

Capacity, Overfitting and Underfitting

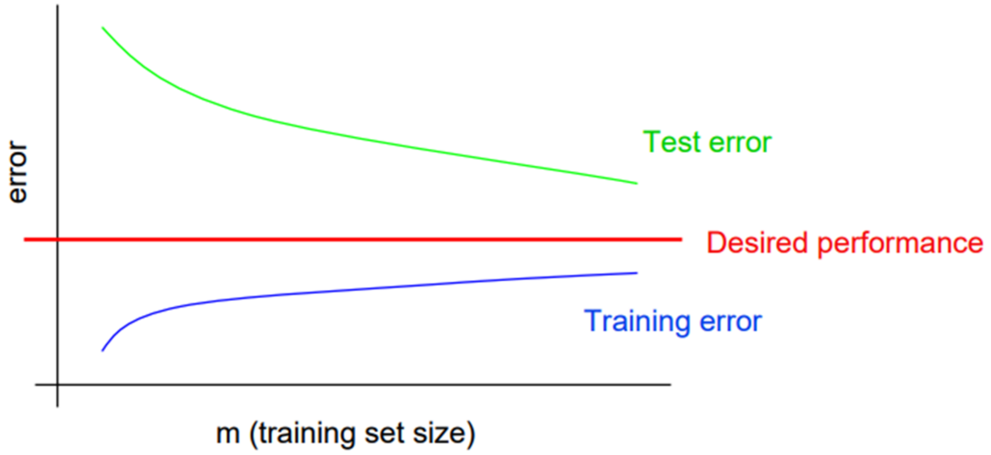


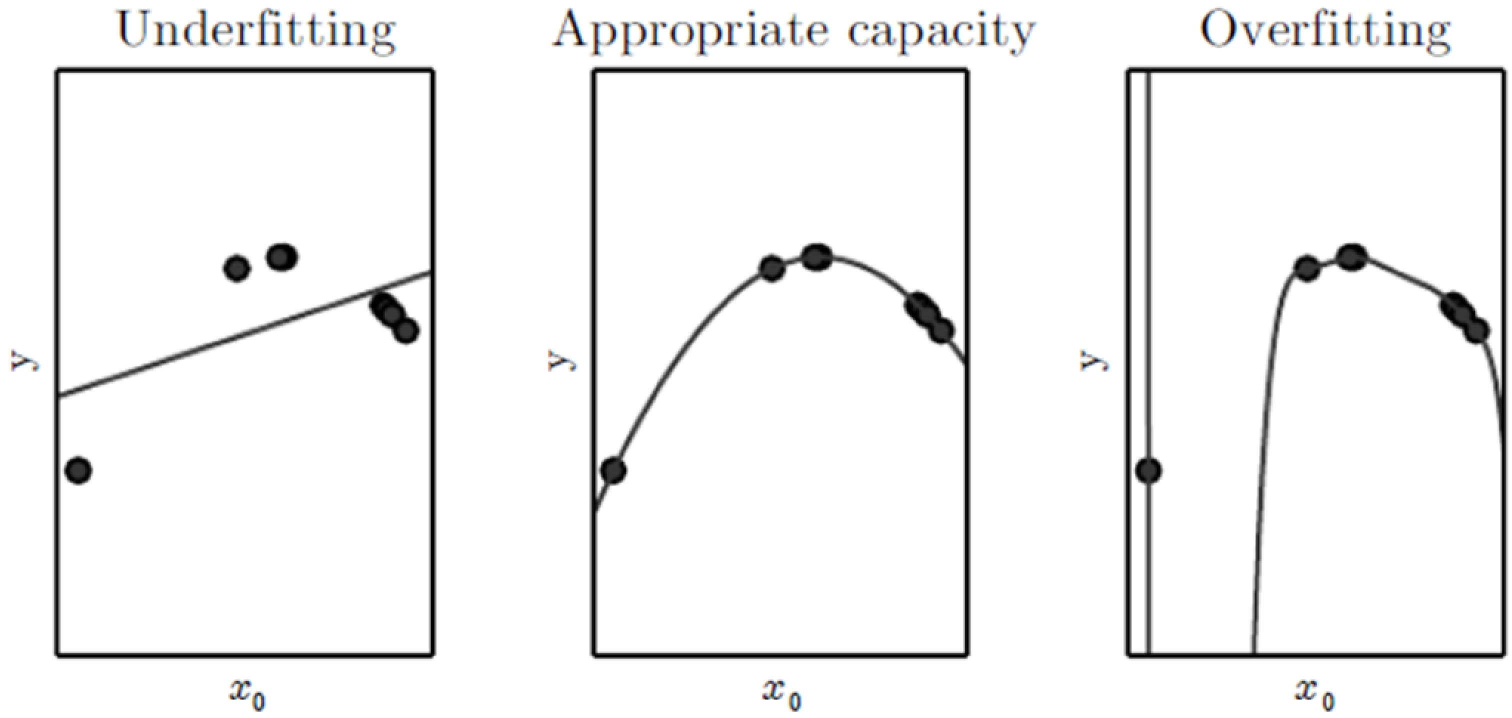
Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set.

Overfitting occurs when the gap between the training error and test error is too large.

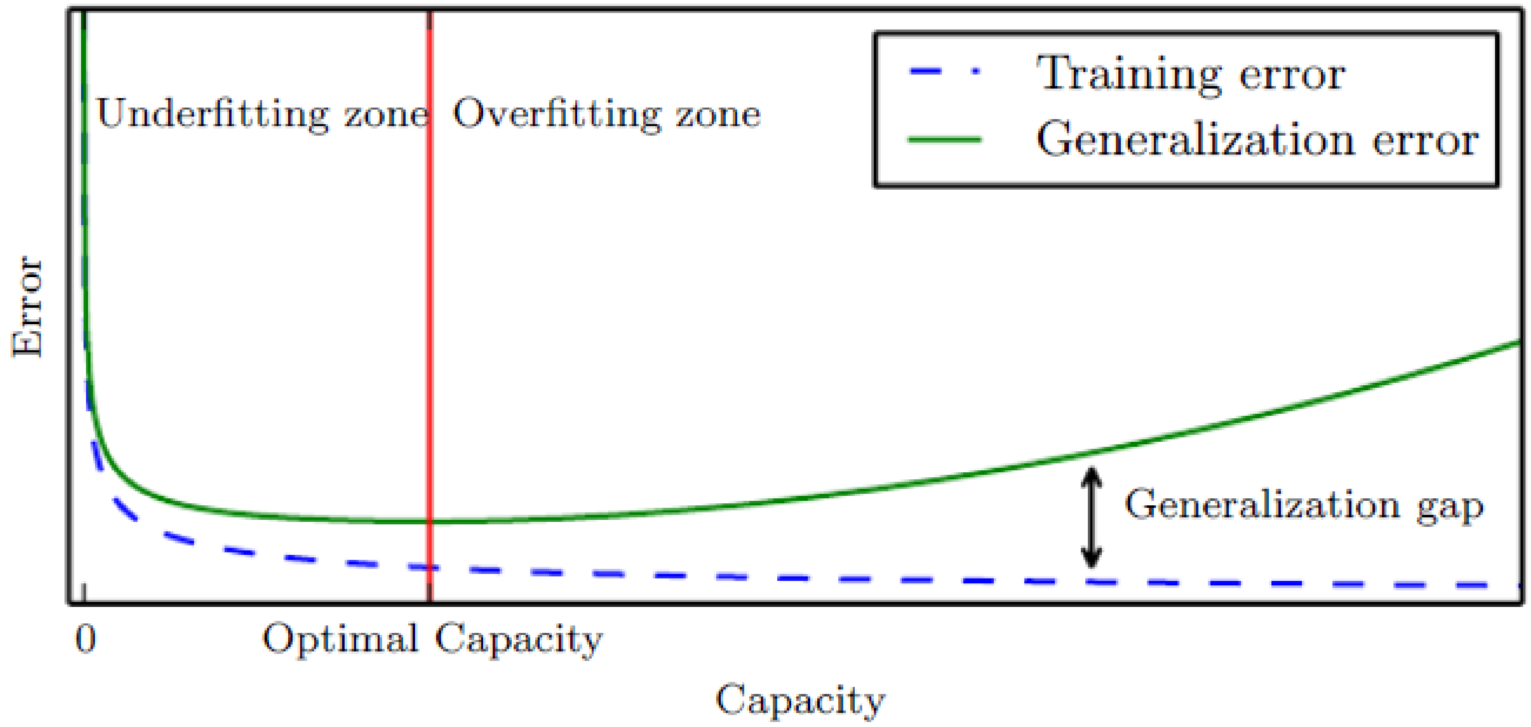
Capacity: is the ability of the model to fit a wide variety of functions.

Capacity, Overfitting and Underfitting





Capacity, Overfitting and Underfitting



Typical relationship between capacity and error.

Regularization

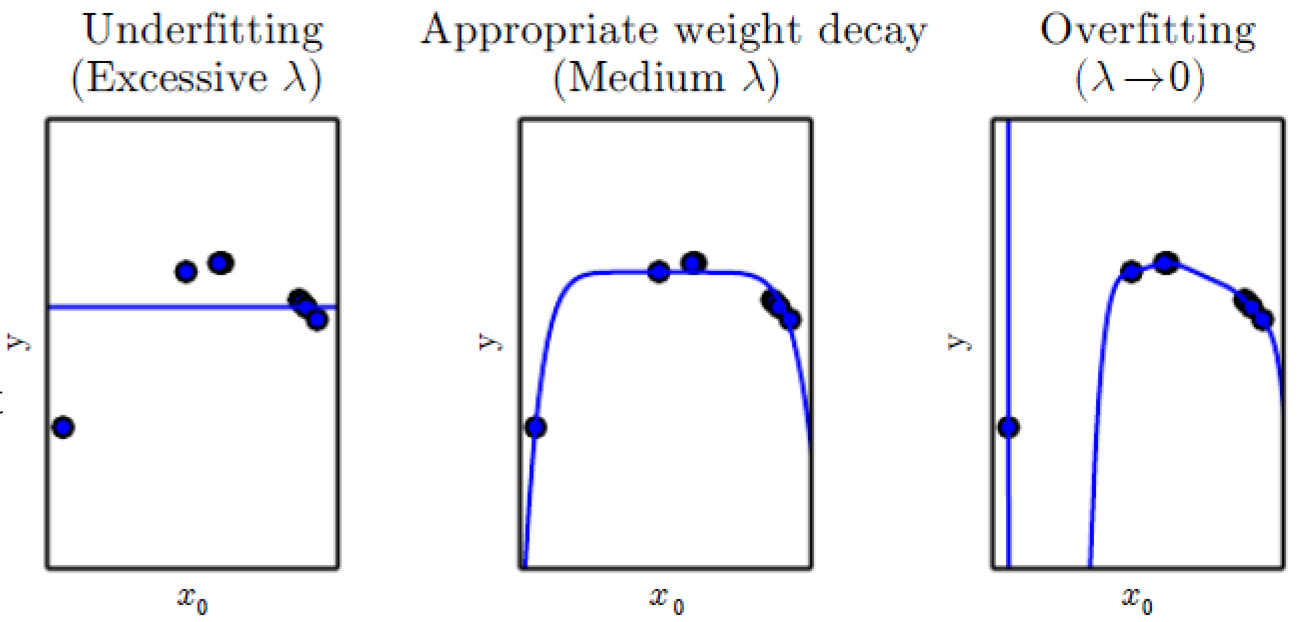
We can give a learning algorithm a preference for one solution in its hypothesis space to another. This means that both functions are eligible, but one is preferred. The unpreferred solution can be chosen only if it fits the training data significantly better than the preferred solution.

Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

Example: linear regression

$$j(w) = MSE_{train} + \lambda w^T w$$

Solutions that have smaller slope, or put weight on fewer of the features.



Hyperparameters

Most machine learning algorithms have several settings that we can use to control the behavior of the learning algorithm. These settings are called *hyperparameters*.

Validation set

We split the training data into two disjoint subsets.

- One of these subsets is used to learn the parameters.
- The other subset is our **validation set**, used to estimate the generalization error during or after training, allowing for the hyperparameters to be updated accordingly.

No example from the test set can be used in the validation set.

Cross-Validation

A small test set implies statistical uncertainty around the estimated average test error, making it difficult to claim that algorithm A works better than algorithm B on the given task. (When the dataset has hundreds of thousands of examples or more, this is not a serious issue.)

Repeating the training and testing computation on different randomly chosen subsets or splits of the original dataset. The most common of these is the k-fold cross-validation procedure

Cross-Validation

```
Define KFoldXV($\mathbb{D}, A, L, k$):
Require: $\mathbb{D}$, the given dataset, with elements $\boldsymbol{z}^{(i)}$
Require: $A$, the learning algorithm, seen as a function that takes a dataset as
    input and outputs a learned function
Require: $L$, the loss function, seen as a function from a learned function $f$ and
    an example $\boldsymbol{z}^{(i)} \in \mathbb{D}$ to a scalar $\in \mathbb{R}$
Require: $k$, the number of folds
    Split $\mathbb{D}$ into $k$ mutually exclusive subsets $\mathbb{D}_i$, whose union is $\mathbb{D}$.
    for $i$ from 1 to $k$ do
        $f_i = A(\mathbb{D} \backslash \mathbb{D}_i)$
        for $\boldsymbol{z}^{(j)}$ in $\mathbb{D}_i$ do
            $e_j = L(f_i, \boldsymbol{z}^{(j)})$
        end for
    end for
    Return $\boldsymbol{e}$
```

Estimators, Bias and Variance

Point Estimation

The attempt to provide the single "best" prediction of some quantity of interest (a single parameter or a vector of parameters).

Let $\{x^{(1)}, \ldots, x^{(m)}\}$ be a set of *m* independent and identically distributed (i.i.d.) data points. A *point estimator* or *statistic of a quantity* is any function of the data:

$$\hat{\boldsymbol{\theta}}_m = g(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)})$$

Estimators, Bias and Variance

Bias

The bias of an estimator is defined as:

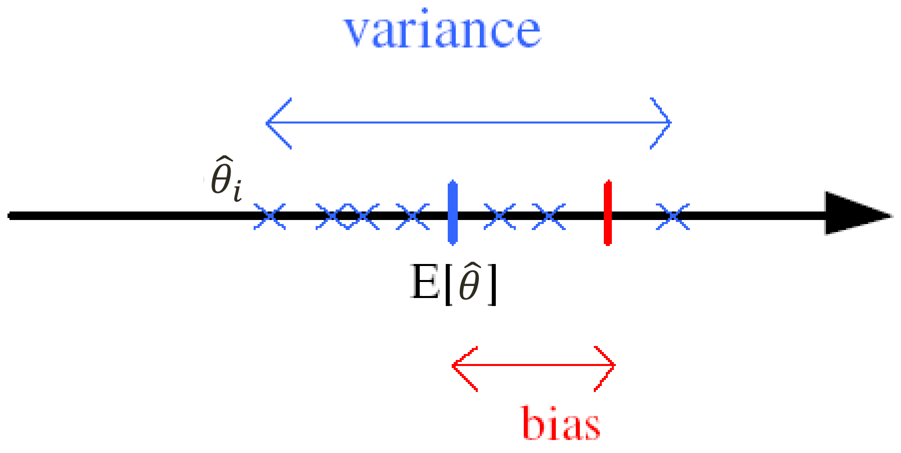
$$\text{bias}(\hat{\boldsymbol{\theta}}_m) = \mathbb{E}(\hat{\boldsymbol{\theta}}_m) - \boldsymbol{\theta}$$

An estimator $\hat{\theta}_m$ is said to be *unbiased* if $\text{bias}(\hat{\theta}_m) = 0$.
An estimator $\hat{\theta}_m$ is said to be *asymptotically unbiased* if $\lim_{m \to \infty} \text{bias}(\hat{\theta}_m) = 0$.

Estimators, Bias and Variance

Variance

Another property of the estimator that we might want to consider is how much we expect it to vary as a function of the data sample.

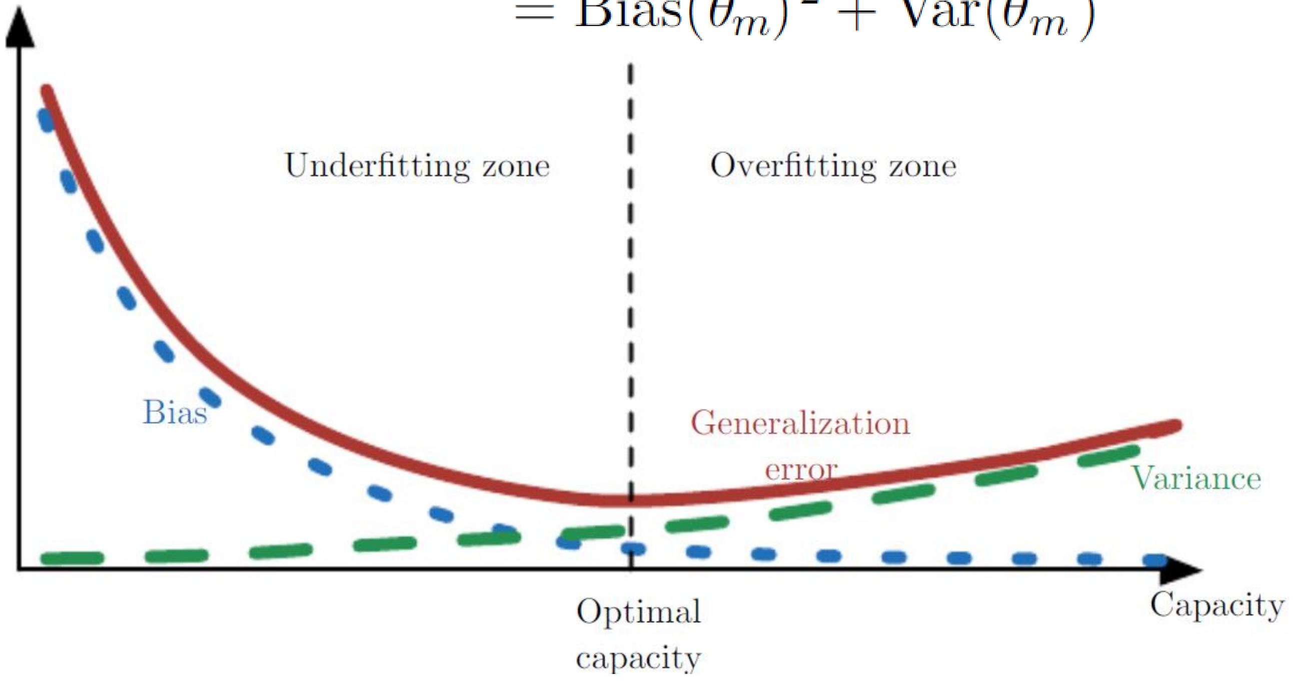


Estimators, Bias and Variance

How do we choose between different estimators?

- ✓ The most common way to negotiate this trade-off is to use cross-validation.
- ✓ Alternatively, we can also compare the *mean squared error* (MSE) of the estimates:

$$\begin{aligned}\mathrm{MSE} &= \mathbb{E}[(\hat{\theta}_m - \theta)^2] \\ &= \mathrm{Bias}(\hat{\theta}_m)^2 + \mathrm{Var}(\hat{\theta}_m)\end{aligned}$$



Maximum Likelihood Estimation

We would like to have some principle from which we can derive specific functions that are good estimators for different models.

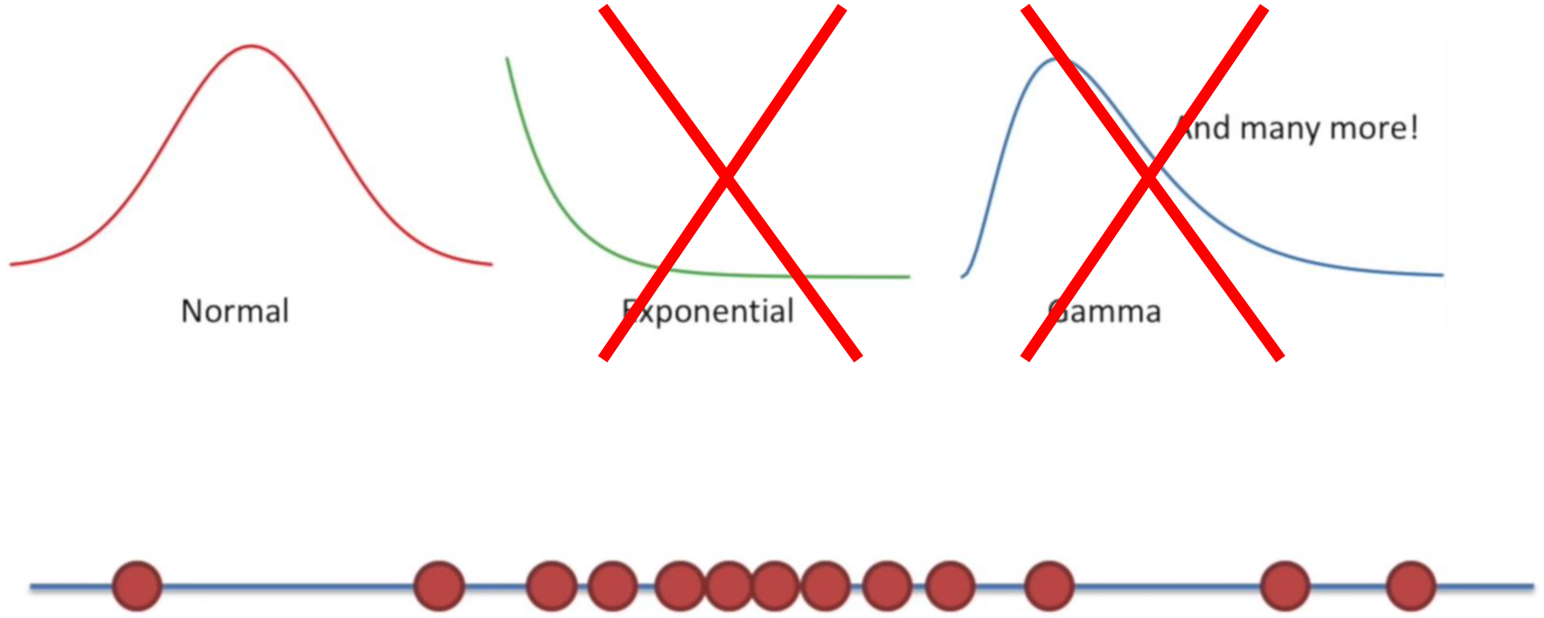
Consider a set of m examples $\mathbb{X} = \{x^{(1)}, \dots, x^{(m)}\}$ drawn independently from the true but unknown data generating distribution $p_{\text{data}}(\boldsymbol{x})$.

Let $p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta})$ be a parametric family of probability distributions over the same space indexed by $\boldsymbol{\theta}$.

In other words, $p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta})$ maps any configuration $\boldsymbol{x}$ to a real number estimating the true probability $p_{\text{data}}(\boldsymbol{x})$.

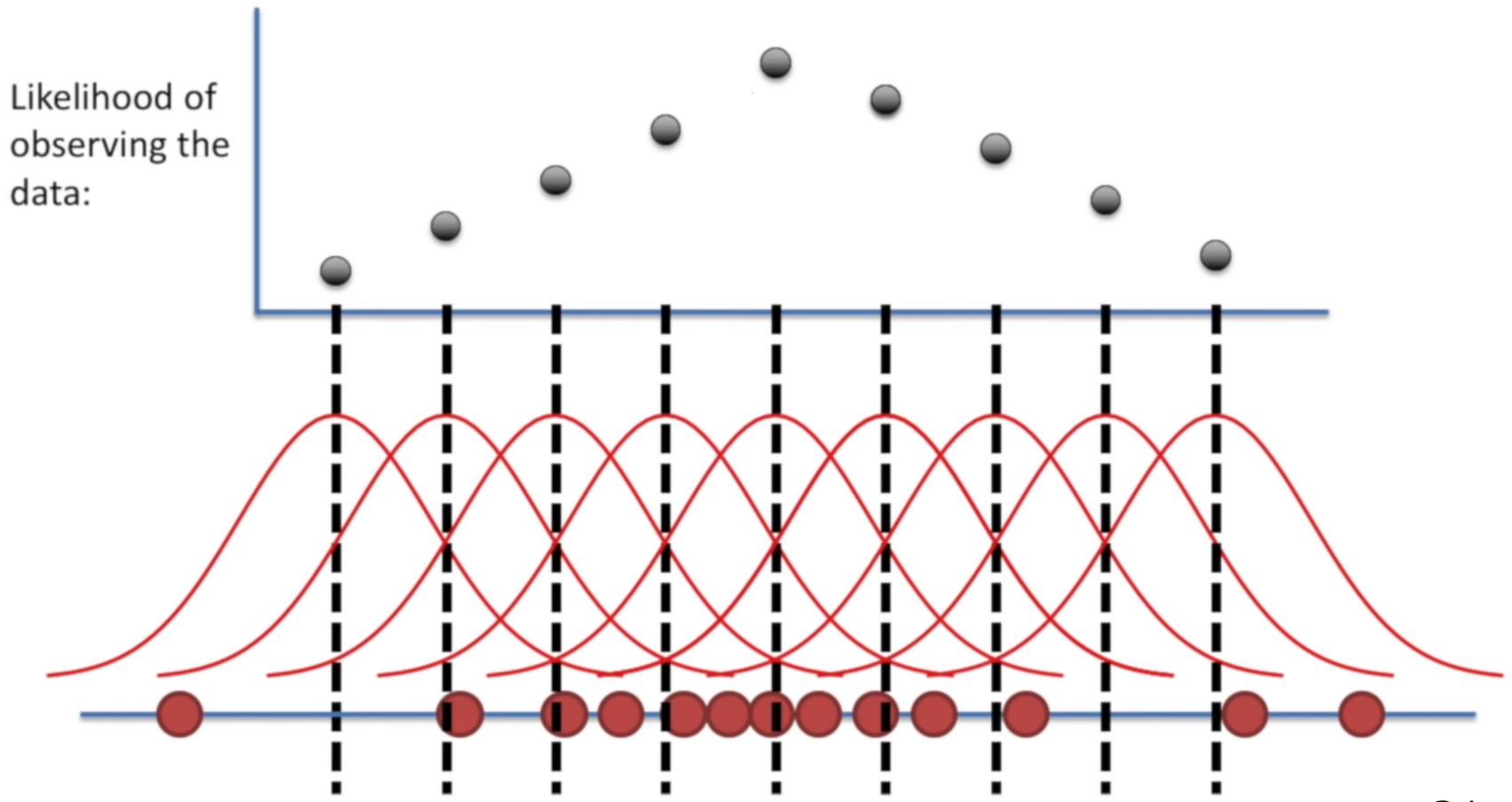
Maximum Likelihood Estimation

Example



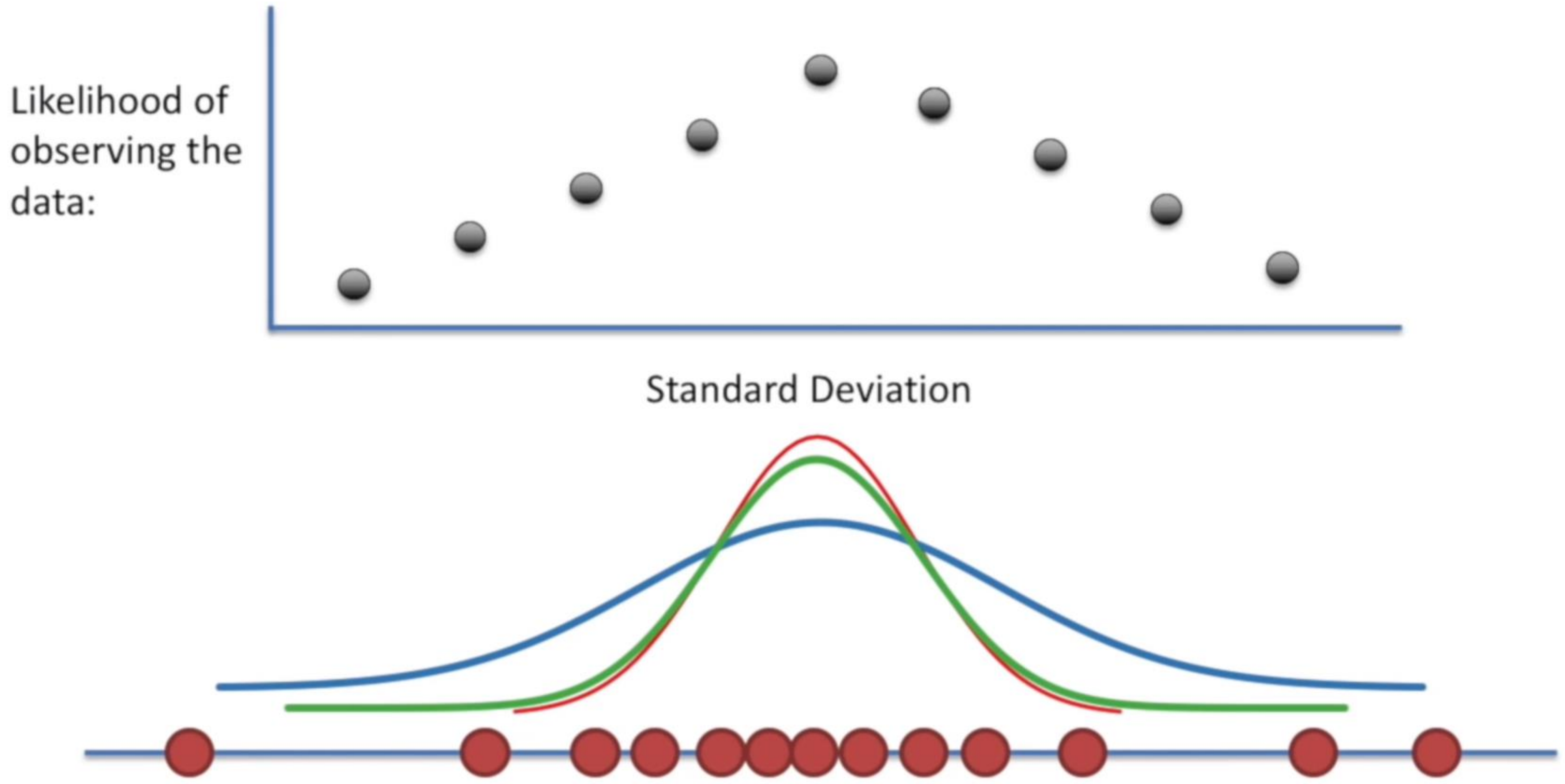
Maximum Likelihood Estimation

Example



Maximum Likelihood Estimation

Example



Maximum Likelihood Estimation

The maximum likelihood estimator for ***θ*** is then defined as

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\mathrm{model}}(\mathbb{X}; \boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\mathrm{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

To obtain a more convenient but equivalent optimization problem

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\mathrm{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

we can divide by *m* to obtain a version of the criterion that is expressed as an expectation

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathrm{data}}} \log p_{\mathrm{model}}(\boldsymbol{x}; \boldsymbol{\theta})$$

Maximum Likelihood Estimation

Conditional Log-Likelihood and Mean Squared Error

The maximum likelihood estimator can readily be generalized to the case where our goal is to estimate a conditional probability $P(y|\mathrm{x};\theta)$ in order to predict y given x.

This is actually the most common situation because it forms the basis for most supervised learning.

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{Y} \mid \boldsymbol{X}; \boldsymbol{\theta})$$

X and ***Y*** represents all the inputs and observed targets, respectively.

If the examples are assumed to be i.i.d.

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log P(\boldsymbol{y}^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

Maximum Likelihood Estimation

Conditional Log-Likelihood and Mean Squared Error
Example: Linear Regression as Maximum Likelihood
We revisit linear regression from the point of view of maximum likelihood estimation, i.e., instead of producing a single prediction $\hat{y}$, we now think of the model as producing a conditional distribution $p(y|\boldsymbol{x})$.

We might see several training examples with the same input value $\boldsymbol{x}$ *but different values of* y.

The goal of the learning algorithm is now to fit the distribution $p(y|\boldsymbol{x})$ to all of those different y values that are all compatible with $\boldsymbol{x}$.

Maximum Likelihood Estimation

Conditional Log-Likelihood and Mean Squared Error

Example: Linear Regression as Maximum Likelihood

$$p(y \mid \boldsymbol{x}) = \mathcal{N}(y; \hat{y}(\boldsymbol{x}; \boldsymbol{w}), \sigma^2)$$

the function $\hat{y}(\boldsymbol{x}; \boldsymbol{w})$ gives the prediction of the mean of the Gaussian.

we assume that the variance is fixed to some constant σ^2 chosen by the user.

$$\sum_{i=1}^{m} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$= -m \log \sigma - \frac{m}{2} \log(2\pi) - \sum_{i=1}^{m} \frac{\|\hat{y}^{(i)} - y^{(i)}\|^2}{2\sigma^2}$$

where $\hat{y}^{(i)}$ is the output of the linear regression on the *i*-th input $\boldsymbol{x}^{(i)}$ and *m* is the number of the training examples.

Maximum Likelihood Estimation

Conditional Log-Likelihood and Mean Squared Error
Example: Linear Regression as Maximum Likelihood

$$-m \log \sigma - \frac{m}{2} \log(2\pi) - \sum_{i=1}^{m} \frac{||\hat{y}^{(i)} - y^{(i)}||^2}{2\sigma^2}$$

Comparing the log-likelihood with the mean squared error,

$$\text{MSE}_{\text{train}} = \frac{1}{m} \sum_{i=1}^{m} ||\hat{y}^{(i)} - y^{(i)}||^2$$

Maximizing the log-likelihood with respect to ***w*** yields the same estimate of the parameters ***w*** as does minimizing the mean squared error.