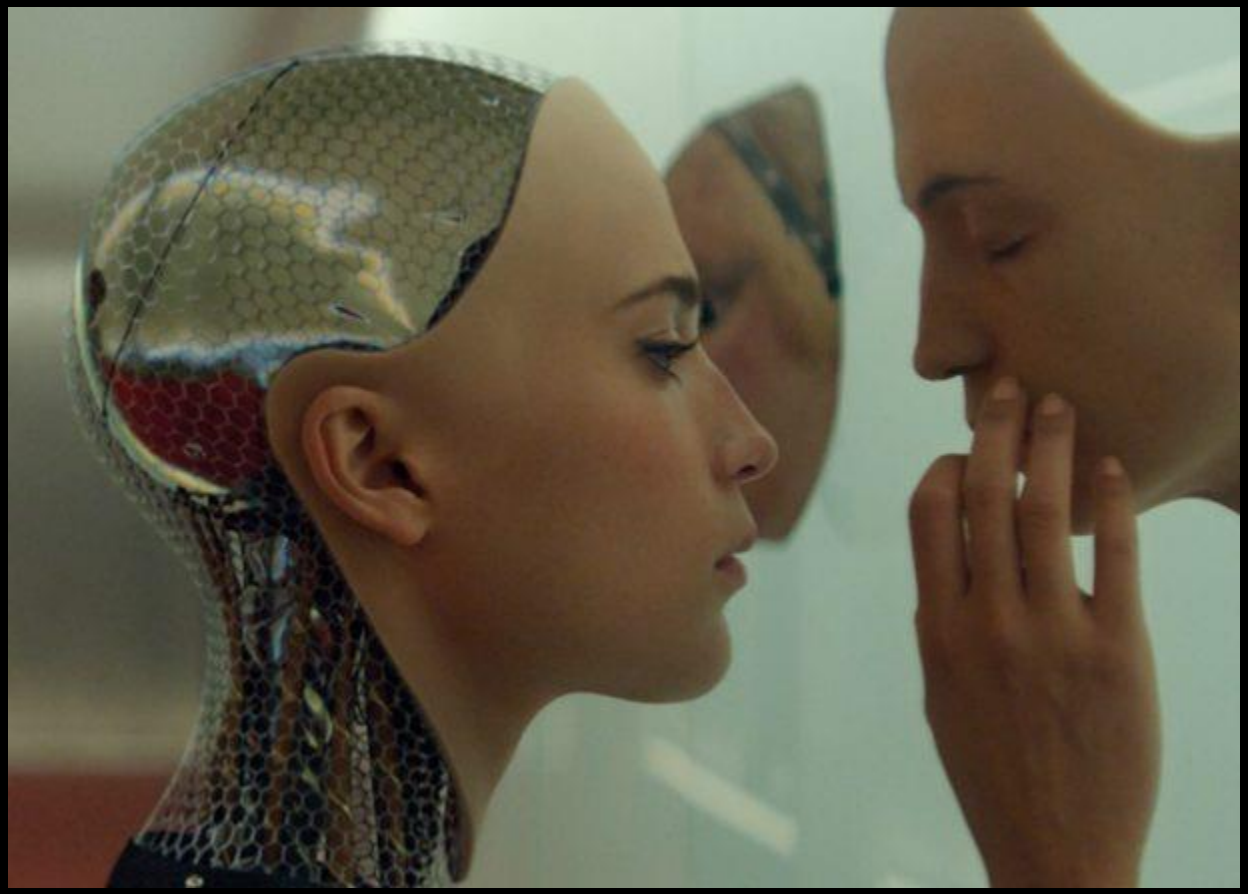
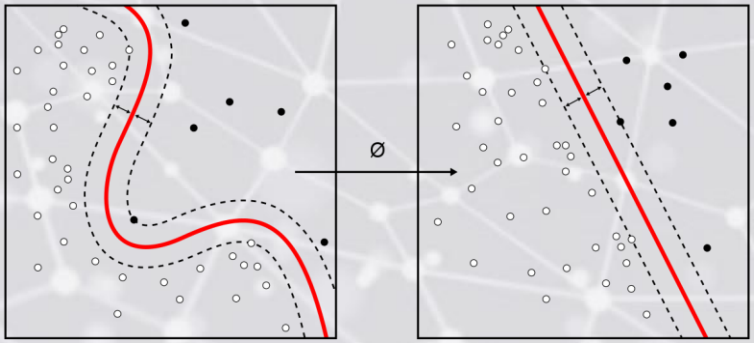
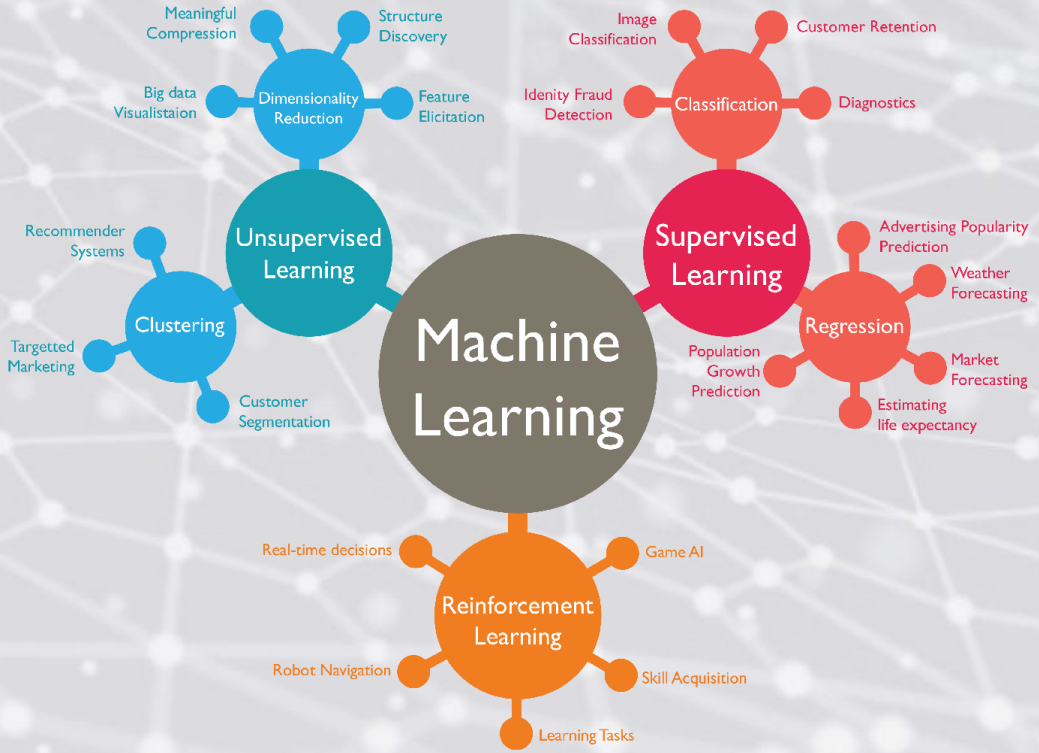


A Brief Overview of General AI/ML Concepts

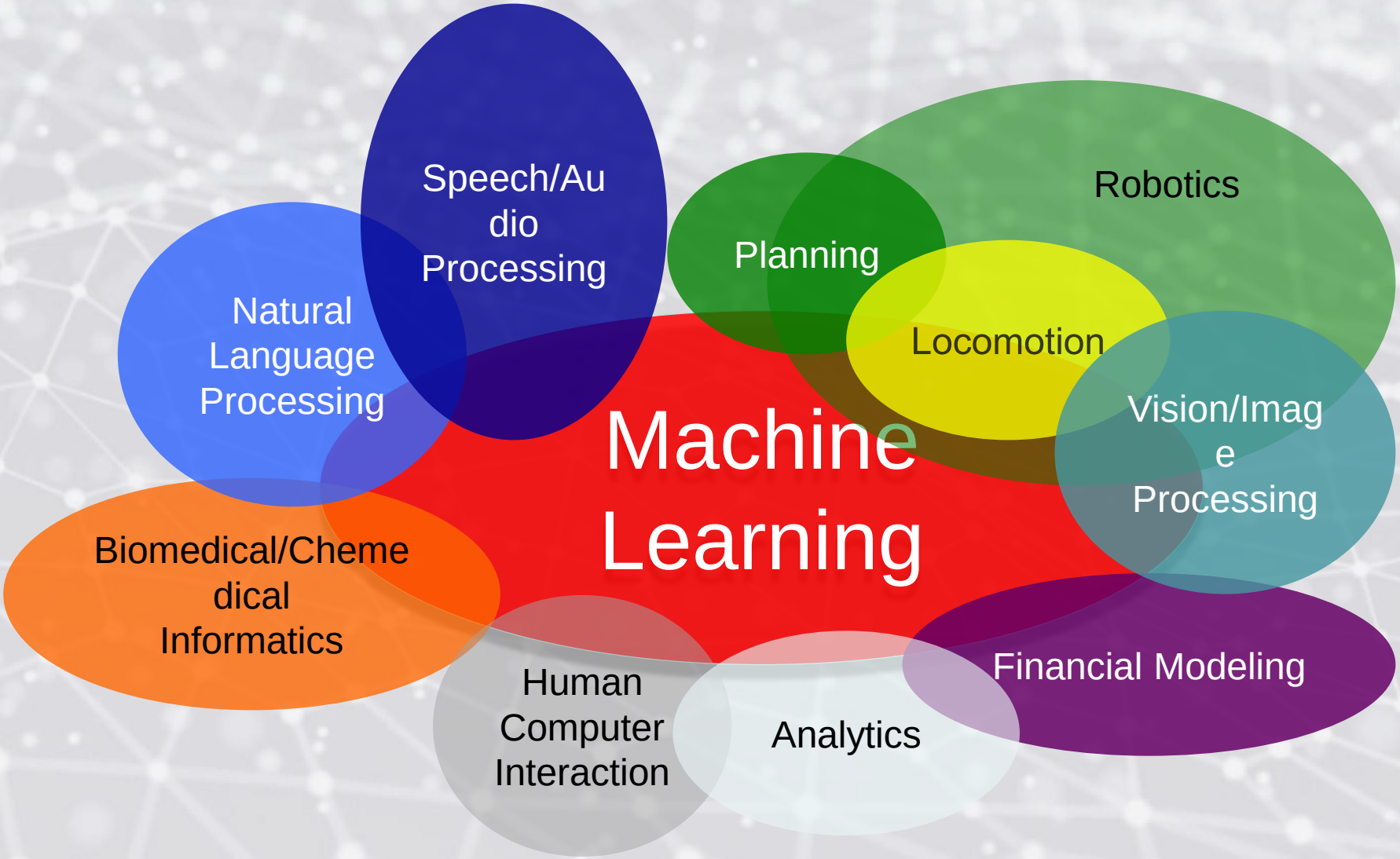
AI/ML Overview

- What is Machine Learning?
  - Detecting patterns and regularities with a good and generalizable approximation ("model" or "hypothesis").
  - Execution of a computer program to optimize the parameters of the model using training data or past experience.
  - Automatically identifying patterns in data.



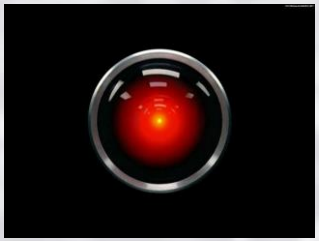


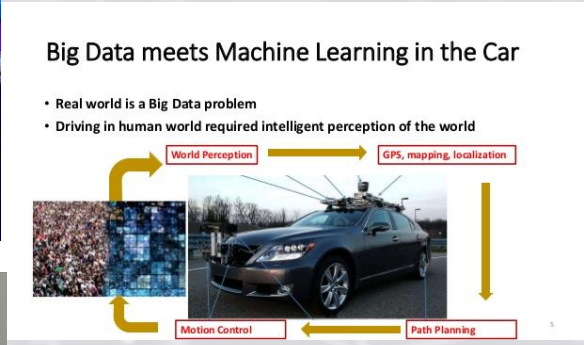
AI/ML Overview

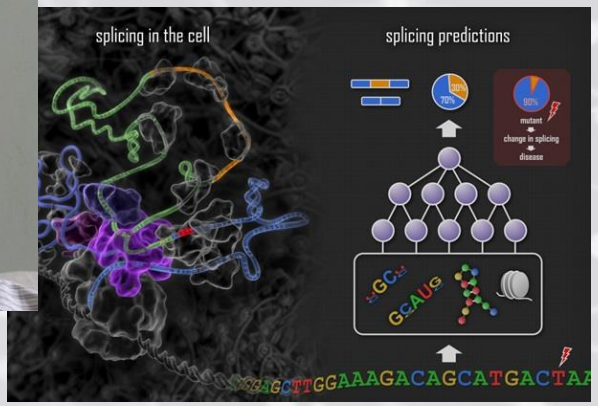


A Small Subset of Machine Learning Applications

(*) Speech Recognition
(*) NLP (natural language processing); machine translation.
(*) Computer Vision
(*) Medical Diagnosis
(*) Autonomous Driving
(*) Statistical Arbitrage
(*) Signal Processing
(*) Recommender Systems
(*) ~~World Domination~~
(*) Fraud Detection
(*) Social Media
(*) Data Security
(*) Search
(*) A.I. & Robotics
(*) Genomics
(*) Computational Creativity
(*) Hi Scores







A Small Subset of Machine Learning Applications

- https://www.youtube.com/watch?v=V1eYniJ0Rnk

- https://www.youtube.com/watch?v=SCE-QeDfXtA

AI/ML Overview

(2) General Classes of Problems in AI/ML:

1. Supervised Learning
2. Unsupervised Learning

Supervised Learning:

Goal is to learn a *mapping* from **inputs** (X) to **labels** (Y): $f : X \to Y$

With supervised learning we are given labels:

$$D = \left\{ \left(\mathbf{x}_i, y_i \right) \right\}_{i=1}^{n} \text{ where } \mathbf{x}_i \in \mathbb{R}^d$$

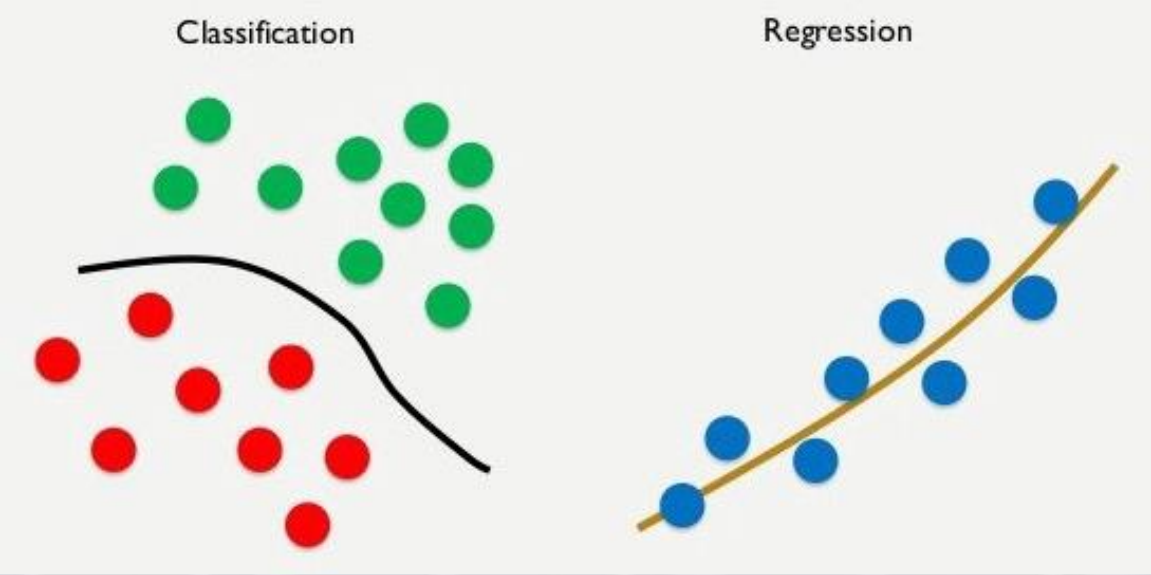
Commonly X denotes the **design matrix** (i.e. the matrix of data), where X is of dimension $n \times d$.

$$\mathbf{X} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & x_d^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^n & \cdots & \cdots & x_d^n \end{bmatrix}$$

AI/ML Overview

• When $y \in \mathbb{R}$ (i.e. the label is a real-value) the problem type is known as **regression** (this problem context is typically broader than say, *linear regression*). E.g., predict expected income from education level.

• On the other hand, when $y \in \{1, \ldots, K\}$ (i.e. the label is categorical) we say that the problem type is **classification**. E.g., predict where image contains a pedestrian (binary classification).



AI/ML Overview

• General goal in ML is to learn the "true" mapping *f*:

$$y = f(\mathbf{x})$$

• Usually, with real-world applications, we can at best approximate the true mapping:

$$y = \underbrace{\hat{f}(\mathbf{x})}_{\text{approximate map}} + \underbrace{\varepsilon}_{\text{irreducible error}}$$

• Why do we bother approximating *f* ? Two basic reasons: (1) Prediction; (2) Inference.

• In general, we want: $\hat{f} \approx f$ so that our model makes reliable predictions on all domain-related data.

AI/ML Overview

• We can quantify the proximity of our approximation $\hat{f} \approx f$ through the use of a **loss function**.

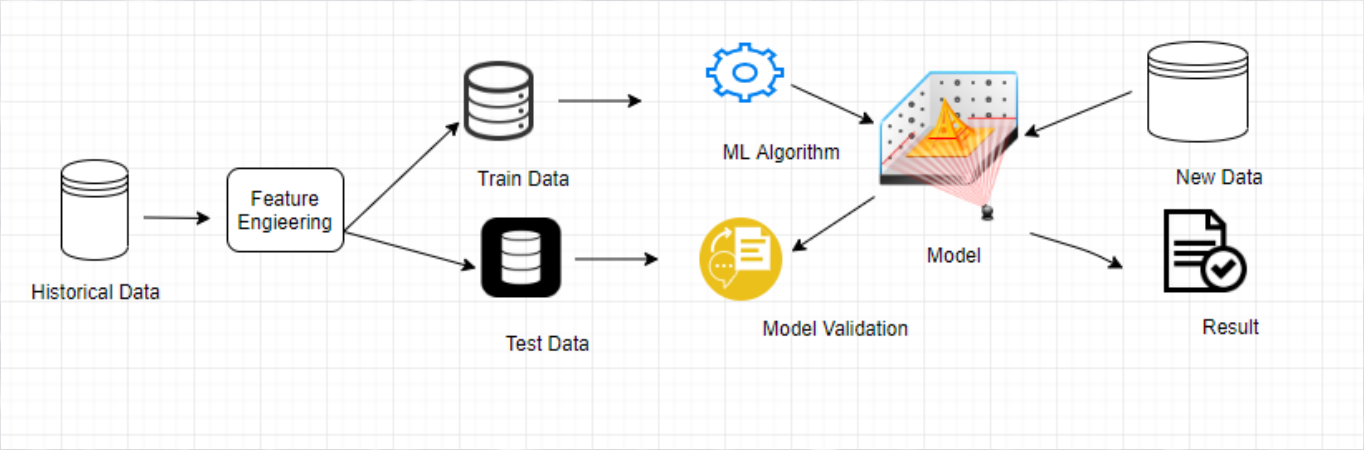
• Two of the most common loss functions used across ML are the **0-1** and **Quadratic** Loss:

0-1 Loss (Binary Classification): $$L\left(f(x), \hat{f}(x)\right) = \begin{cases} 0 \text{ if } f(x) = \hat{f}(x) \\ 1 \text{ if } f(x) \neq \hat{f}(x) \end{cases}$$

Quadratic Loss: $$L\left(f(x), \hat{f}(x)\right) = \left(f(x) - \hat{f}(x)\right)^2$$

AI/ML Overview

Machine Learning Workflow:



1. Collect data: $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, partition data into **training** and **test** sets:

$$D_{train} \subset D,\ D_{test} \subset D$$

$$D_{train} \cup D_{test} = D,\ D_{train} \cap D_{test} = \varnothing$$

2. Train model $\hat{f}$ (e.g. regression, NN) using D_{train}.

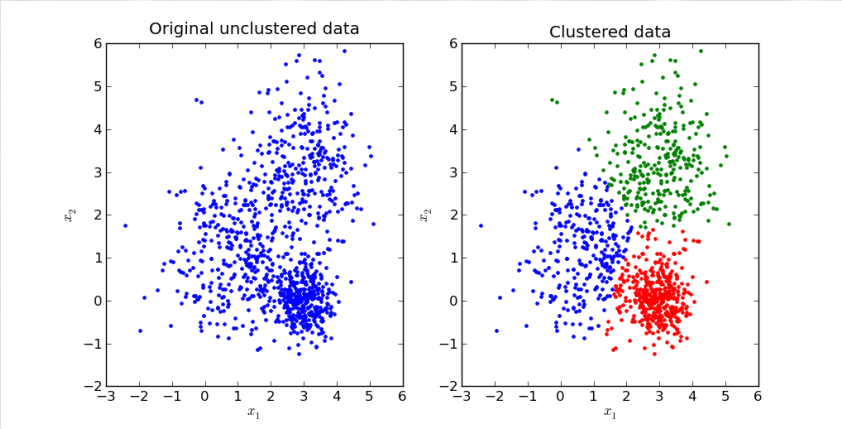
3. Evaluate model with loss function on D_{test}.

*Big Idea: The smaller the (total) loss on the test set, the better the model (ideally). We use the results on the test set to approximate how well the model will generalize to new data.

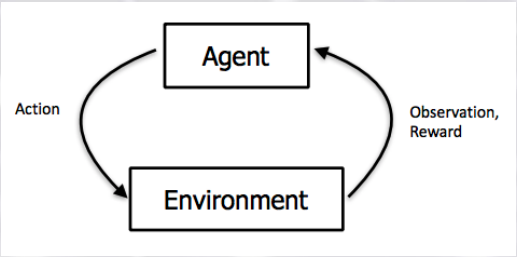
AI/ML Overview

• With **unsupervised learning** we are given data without labels.

• In this case we aim to discover "interesting structure" in the data; this is sometimes called knowledge discovery or cluster analysis.



*Note: **Reinforcement Learning** offers a third problem class in AI/ML, where an "agent" learns how to act or behave when given occasional reward or punishment signals (e.g. *Atari w/ Deep Q-Learning* (2014), *AlphaGo* (2016)).



AI/ML Overview

Parametric Models vs non-Parametric Models:

- Parametric models consist of a finite (and fixed) number of parameters:

$$\boldsymbol{\theta} = \langle \theta_1, \ldots, \theta_N \rangle$$

*Idea: With an ML algorithm, we learn to "tune" these parameters.

Ex. Fit a polynomial curve to a data set (e.g. using OLS).

Linear Regression: $\hat{f}(x) = \theta_0 + \theta_1 x$

Quadratic Regression: $\hat{f}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

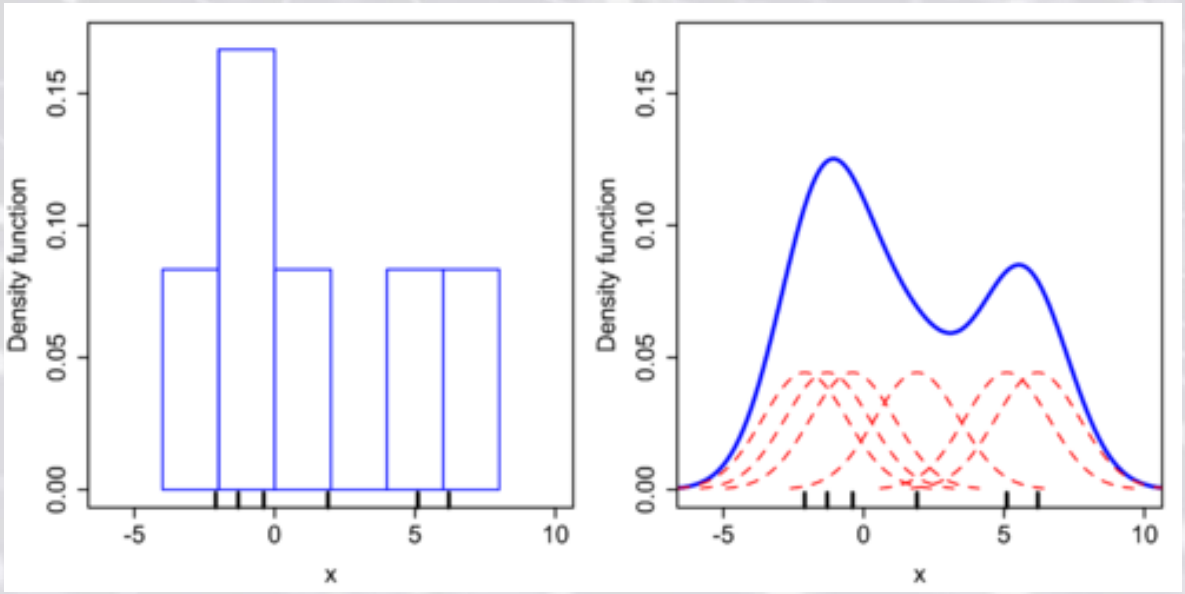
Polynomial Regression: $\hat{f}(x) = \sum_{i=0}^{d} \theta_i x^i$ (d+1 parameters: $\boldsymbol{\theta} = \langle \theta_0, \ldots, \theta_d \rangle$)

AI/ML Overview

Parametric Models vs non-Parametric Models:

• A **non-parametric model** contains either an infinite number of parameters (e.g. *Gaussian Process*) or a variable number of parameters (e.g. *kernel density estimation*) --typically the number of parameters scales with size of the data.

Histograms (left) and kernel density estimation (right) represent examples of non-parametric models, as each model becomes more complex/refined as the size of the dataset grows.



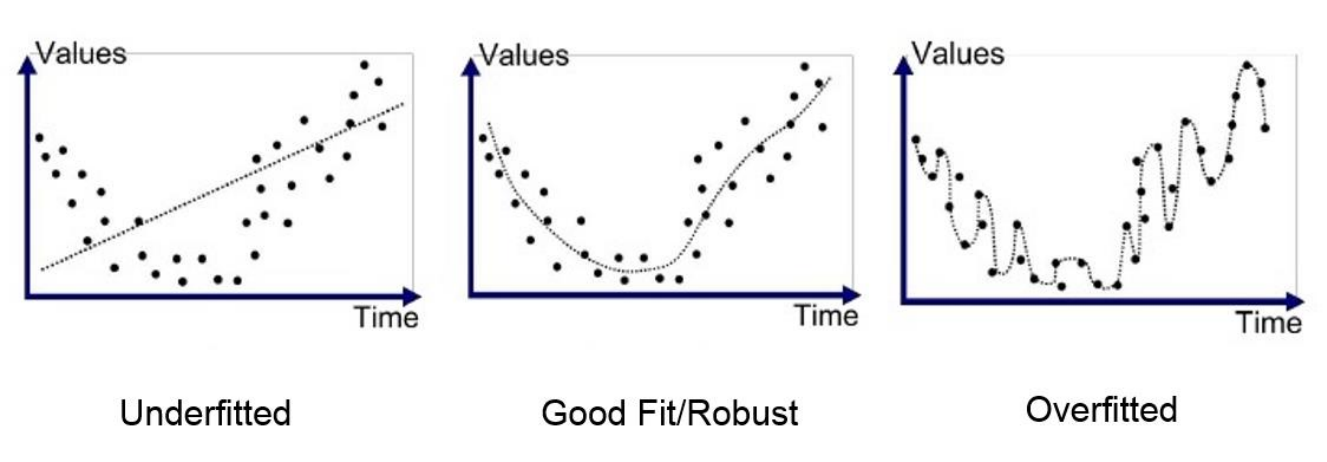
AI/ML Overview

Parametric Models vs non-Parametric Models:

*Note: If we use a model with a small number of parameters, it is usually easier to train (requires less time and data). However, a low dimensional model might not be sufficiently complex to capture all of the interesting and useful patterns in our data! (This phenomenon is called **underfitting**)

• Conversely, a large dimensional/complex model requires more computation and time on average; moreover, an excessively complex model will be "over tuned" to the training data – this is called **overfitting**.

Conclusion: There is "no free lunch" in ML!



AI/ML Overview

Parametric Models vs non-Parametric Models:

- How do we know when we get it "right" with respect to fitting a model?

Unfortunately, there is no general-purpose answer – this is the nature of the "art" of ML.

In general, however, we can assess our model accuracy with a loss function:

Quadratic Loss: $\underset{\text{mean-squared error}}{\text{MSE}} = \frac{1}{n}\underbrace{\sum_{i=1}^{n}\left(y_i - \hat{f}\left(x_i\right)\right)^2}_{\text{over test data}}$

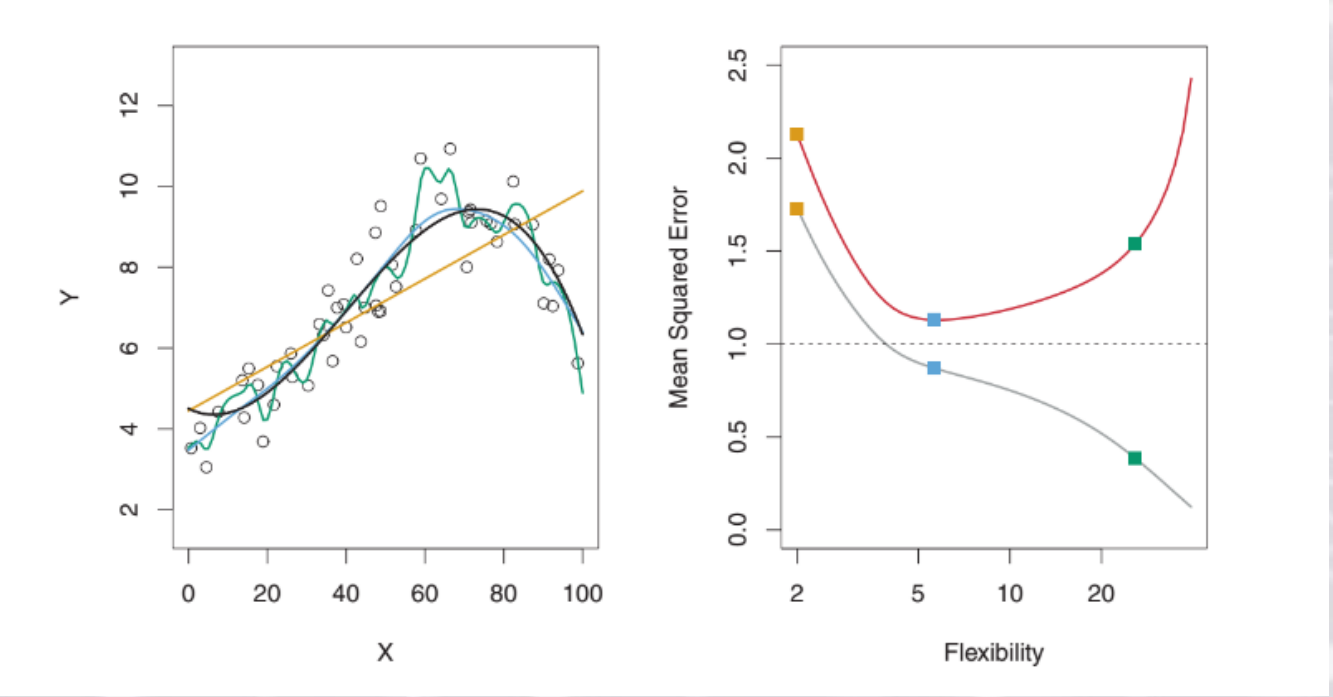
0-1 Loss: $\frac{1}{n}\underbrace{\sum_{i=1}^{n} L\left(y_i, \hat{f}\left(x_i\right)\right)}_{\text{counts \# of "mistakes"}}$

AI/ML Overview

Parametric Models vs non-Parametric Models:

*Note: Unfortunately, having a low training error (e.g. MSE) does not guarantee low test error in general.

• One common remedy for parametric models: train several models of varying complexity (e.g. linear regression, quadratic, cubic regression), compute MSE for each test set, choose the model with the lowest MSE.



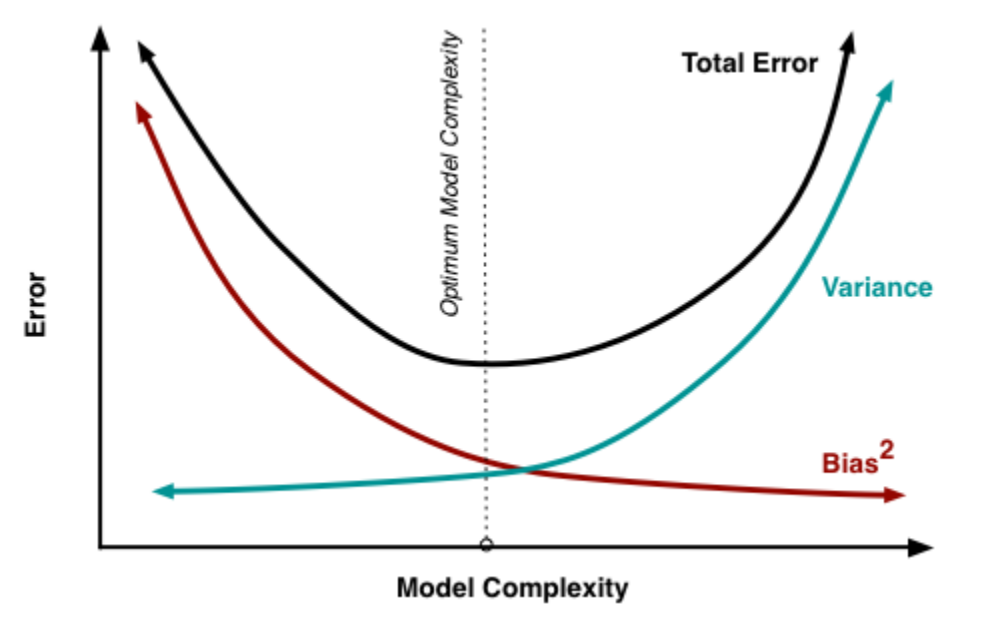
AI/ML Overview

Bias-Variance Tradeoff:

• The "U-shape" phenomenon in the test MSE is indicative of two competing properties of learned models: Bias and Variance.

Low-Dimensional (simple models): High Bias & Low Variance

High-Dimensional (complex/flexible models): Low Bias & High Variance



AI/ML Overview

Bias-Variance Tradeoff:

• More concretely, the **expected Test MSE** with respect to the parameter estimate can always be decomposed into the sum (2) fundamental quantities: Bias and Variance.

$$
\begin{aligned}
\mathrm{MSE}(\hat{\theta}) \equiv \mathbb{E}((\hat{\theta}-\theta)^2) &= \mathbb{E}\left[\left(\hat{\theta}-\mathbb{E}(\hat{\theta})+\mathbb{E}(\hat{\theta})-\theta\right)^2\right] \\
&= \mathbb{E}\left[\left(\hat{\theta}-\mathbb{E}(\hat{\theta})\right)^2+2\left((\hat{\theta}-\mathbb{E}(\hat{\theta}))(\mathbb{E}(\hat{\theta})-\theta)\right)+\left(\mathbb{E}(\hat{\theta})-\theta\right)^2\right] \\
&= \mathbb{E}\left[\left(\hat{\theta}-\mathbb{E}(\hat{\theta})\right)^2\right]+2\mathbb{E}\left[(\hat{\theta}-\mathbb{E}(\hat{\theta}))(\mathbb{E}(\hat{\theta})-\theta)\right]+\mathbb{E}\left[\left(\mathbb{E}(\hat{\theta})-\theta\right)^2\right] \\
&= \mathbb{E}\left[\left(\hat{\theta}-\mathbb{E}(\hat{\theta})\right)^2\right]+2(\mathbb{E}(\hat{\theta})-\theta)\overbrace{\mathbb{E}(\hat{\theta}-\mathbb{E}(\hat{\theta}))}^{=\mathbb{E}(\theta)-\mathbb{E}(\theta)=0}+\mathbb{E}\left[\left(\mathbb{E}(\hat{\theta})-\theta\right)^2\right] \\
&= \mathbb{E}\left[\left(\hat{\theta}-\mathbb{E}(\hat{\theta})\right)^2\right]+\mathbb{E}\left[\left(\mathbb{E}(\hat{\theta})-\theta\right)^2\right] \\
&= \mathrm{Var}(\hat{\theta})+\mathrm{Bias}(\hat{\theta},\theta)^2
\end{aligned}
$$

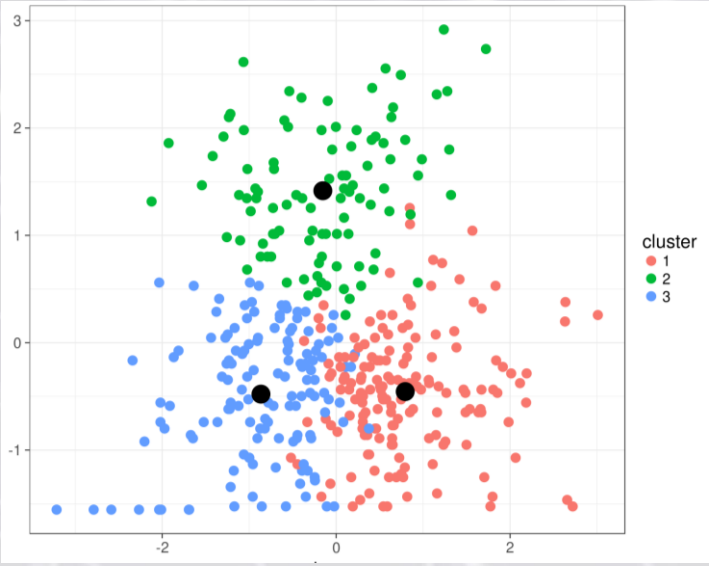
*From above, we see that the ideal model will simultaneously achieve low Variance and low Bias.

AI/ML Overview

Unsupervised Learning:

- Suppose we have $D = \{(\mathbf{x}_i)\}_{i=1}^{n}$ with no class labels (i.e. no y values).

- We will use a clustering method to first cluster the data (let k represent the number of clusters), then classify a new datum based on a nearest centroid criterion – this algorithm is called **k-means**.



*In this case, inference for a new datum $\mathbf{x}^*$ is performed by identifying the cluster c with the minimum distance from the class centroid ($\boldsymbol{\mu}$).

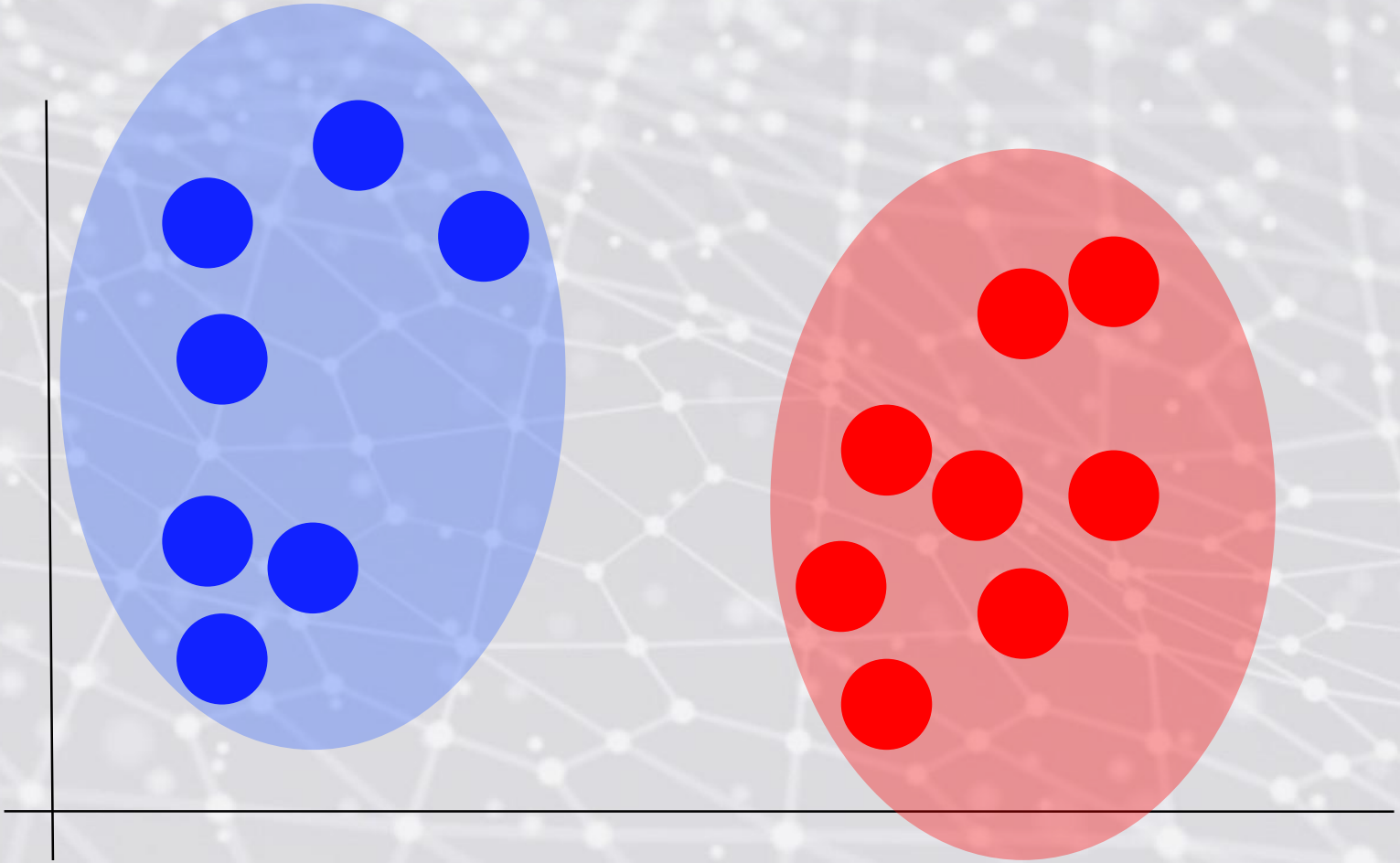
$$y^* = \underset{c \in C}{\operatorname{argmin}} \left\| \mathbf{x}^* - \boldsymbol{\mu}_c \right\|$$

AI/ML Overview

Clustering Example

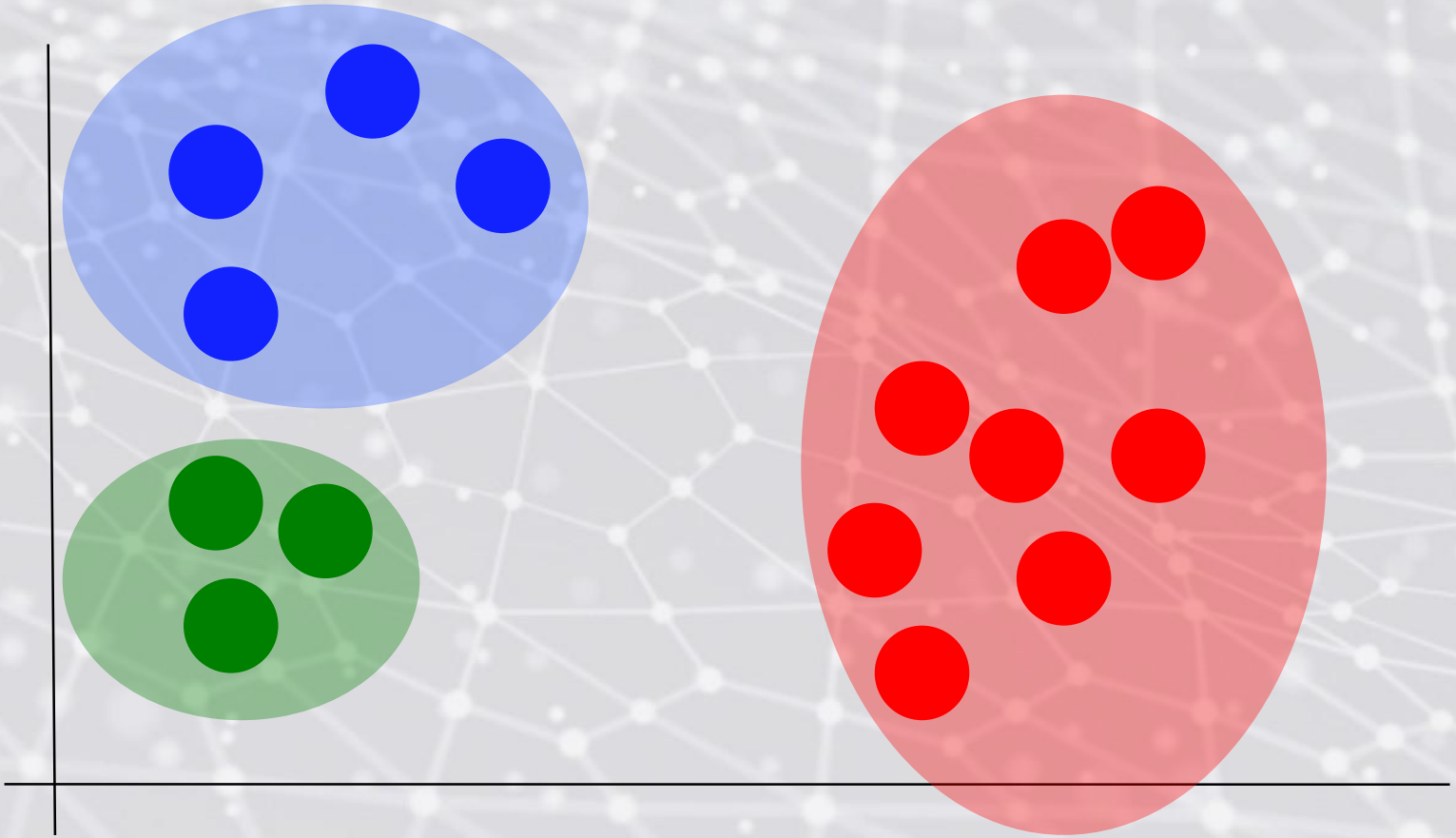
AI/ML Overview

Clustering Example



AI/ML Overview

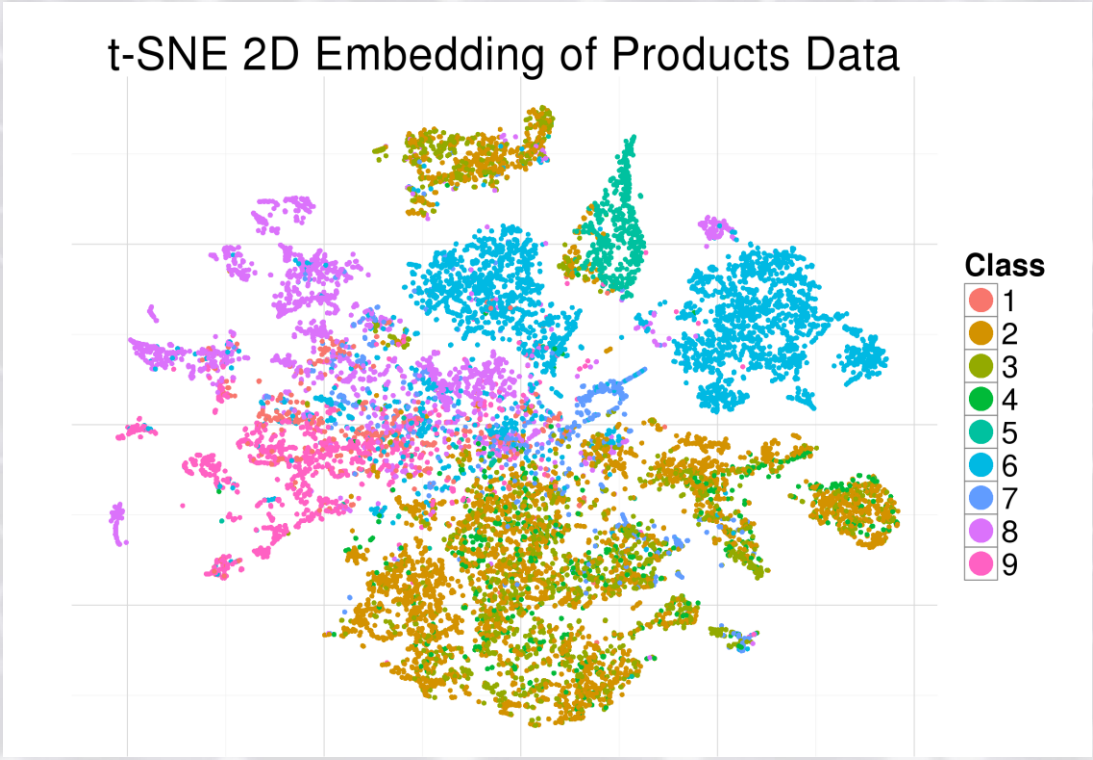
Clustering Example



AI/ML Overview

MNIST Classification

- 60k training/10k test images
- LeCun, Bengio, *et al.* (1998) used SVMs to get error rate of 0.8%.
- More recent research using CNNs (a type of neural network) yields 0.23% error.

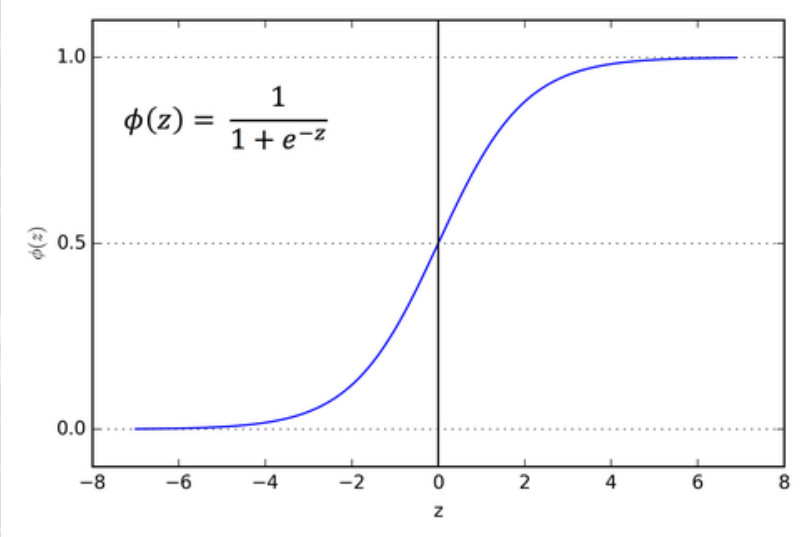


AI/ML Overview

Logistic Regression:

• **Logistic Regression** is a standard parametric (binary) classification model in ML.

• Logistic regression makes use of a *logistic* (i.e. **sigmoid φ(z)**) function that is common to many different ML models (in particular, sigmoids are often used as *activation functions* in NNs).



In general, a *multi-variate sigmoid function* is defined:

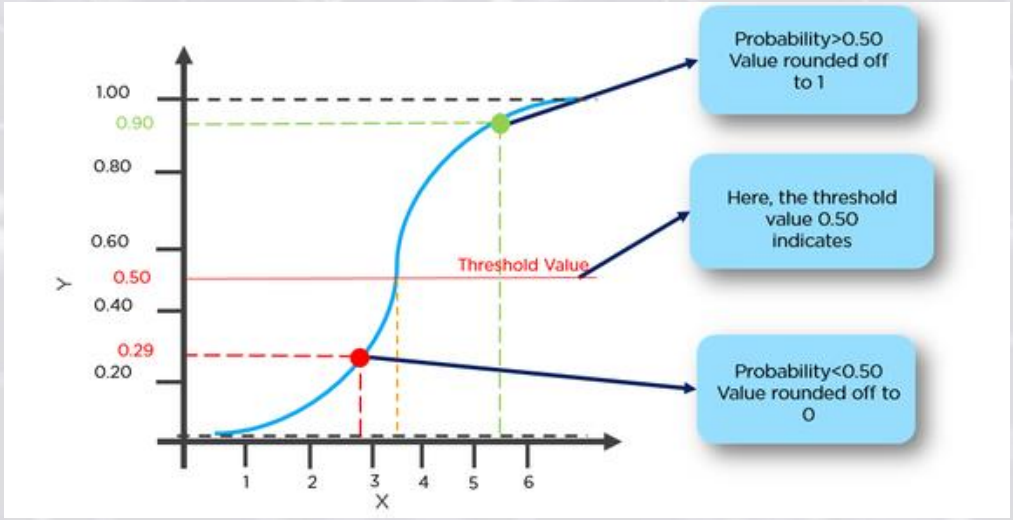
$$\phi\left(\underbrace{\mathbf{x}}_{\substack{\text{input}\\\text{datum}}}, \underbrace{\boldsymbol{\theta}}_{\substack{\text{model}\\\text{parameters}}}\right) = \frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{x}}}$$

AI/ML Overview

Logistic Regression:

$$p\left(\underbrace{y}_{\text{binary class}} \mid \underbrace{\mathbf{x}}_{\text{input datum}}, \underbrace{\boldsymbol{\theta}}_{\text{model parameters}}\right) = \underset{\text{Bernoulli}}{Ber}\left(y \mid \underbrace{sigm\left(\boldsymbol{\theta}^T \mathbf{x}\right)}_{\text{sigmoid}}\right), \quad y \in \{0,1\}$$

- Steps to train and evaluate a logistic regression model:
  1. Using training data, "tune" model parameters: $\boldsymbol{\theta} = \langle \theta_1, \ldots, \theta_N \rangle$
  2. Inference (i): Pass test datum $\mathbf{x}^*$ through sigmoid $\phi(\mathbf{x}^*, \boldsymbol{\theta}) = \dfrac{1}{1+e^{-\boldsymbol{\theta}^T \mathbf{x}^*}}$
  3. Inference (ii): Apply a *decision rule* (i.e. threshold):

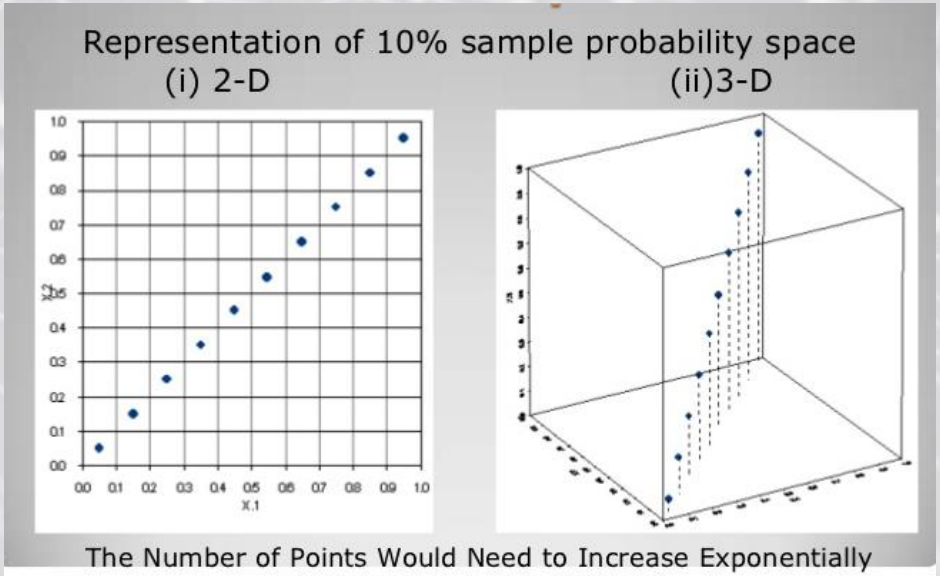


$$y(\mathbf{x}^*) = 1 \leftrightarrow p(y = 1 \mid \mathbf{x}^*, \boldsymbol{\theta}) \geq 0.5$$

AI/ML Overview

The Curse of Dimensionality:

- In ML we are faced with a fundamental dilemma: to maintain a given model accuracy in higher dimensions we need a huge amount of data!
- An exponential increase in data required to densely populate space as the dimension increases.
- Points are equally far apart in high dimensional space (this is counter-intuitive).

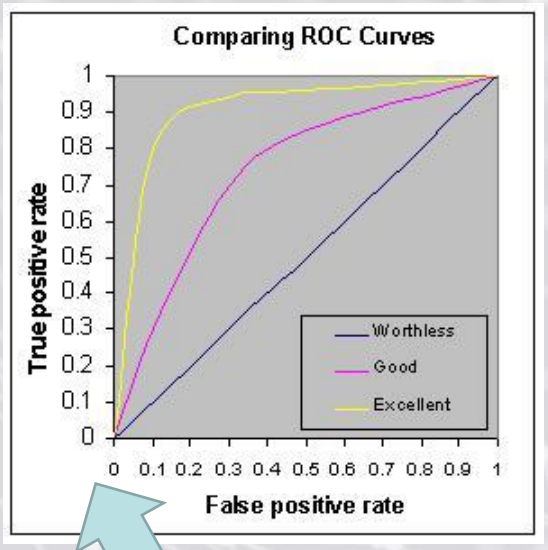


AI/ML Overview

Confusion Matrix, ROC curves, etc.:

• A **Confusion Matrix** is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

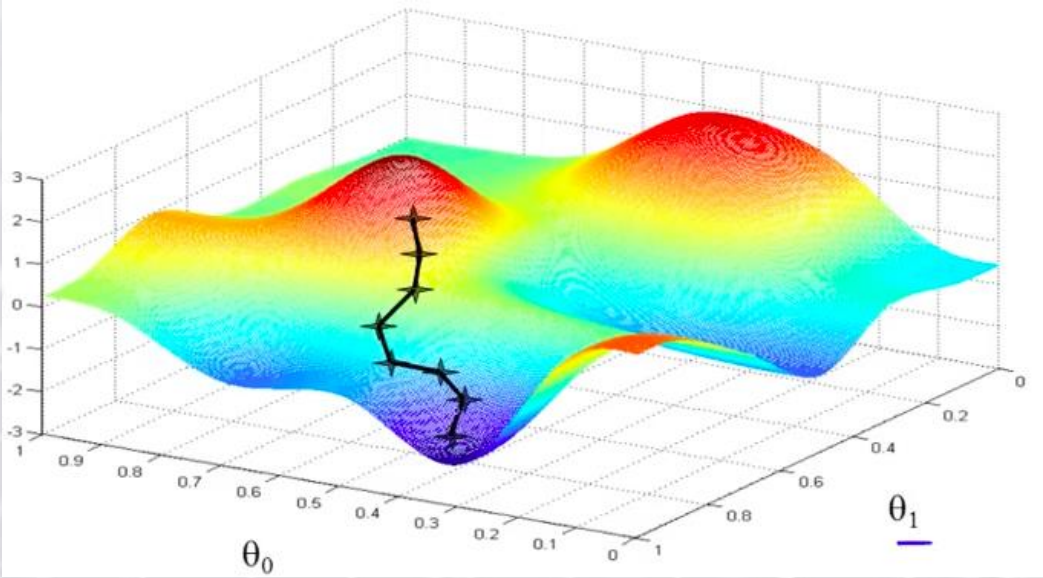


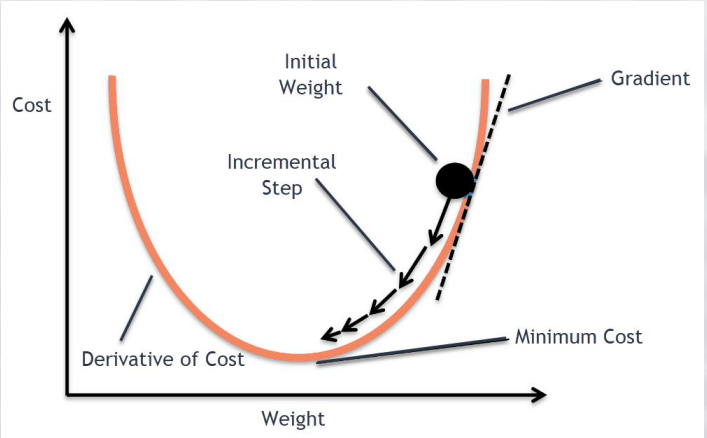
Measure	Formula
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Misclassification rate (1 – Accuracy)	$\frac{FP + FN}{TP + TN + FP + FN}$
Sensitivity (or Recall)	$\frac{TP}{TP + FN}$
Specificity	$\frac{TN}{TN + FP}$
Precision (or Positive Predictive Value)	$\frac{TP}{TP + FP}$

Area under (the) ROC curve (AUC) is a common metric used to assess/compare classifiers.

AI/ML Overview

Gradient Descent (the workhorse of ML):





General formula for Gradient Descent:

$$\underbrace{\boldsymbol{\theta}_{n+1}}_{\text{model parameter estimate}} = \boldsymbol{\theta}_n - \underbrace{\eta}_{\text{"learning rate"}} \cdot \underbrace{\nabla F(\boldsymbol{\theta}_n)}_{\text{gradient of loss function}}$$

Idea: We incrementally update the estimate of our model parameters by "walking" downhill in the parameter space.

• The step-size of the parameter updates is modulated by the **learning rate parameter** (η); a large value for η can lead to a faster convergence of the model parameters – however, we then risk settling into a local minimum. Ideally η should be set to balance speed of convergence with achieving a satisfactory approximation of the global minimum of the loss function (F).

Fin