Introduction to Machine Learning

WSS ML Workshop

Hosein Hasani

WSS 2024

Outline

- Introduction and Motivation
- ML Problems
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning
- Loss Function and Optimization
- Generalization and Overfitting

Machine Learning: Motivation

Extensive influence of Machine Learning across multitude of applications and everyday life:

- Computer Vision
- Signal Processing
- Audio and Speech Recognition
- Natural Language Processing
- Computational Social Science
- Control

Machine Learning: Motivation

Extensive influence of Machine Learning across multitude of applications and everyday life:

- Computational Biology and Bioinformatics
- Medicine, Diagnosis and Health Care
- Computational Neuroscience
- Brain-Computer Interface
- Financial Forecasting
- Recommender Systems

Machine Learning: Motivation

Why ML applications are growing?

- Improved machine learning algorithms
- Availability of data

  (Increased data capture, networking, …)
- Algorithms too complex to write by hand
  - Demand for complex systems

    (high-dimensional, multi-modal, …)
  - Demand for self-customization to user or environment

Machine Learning: Concept

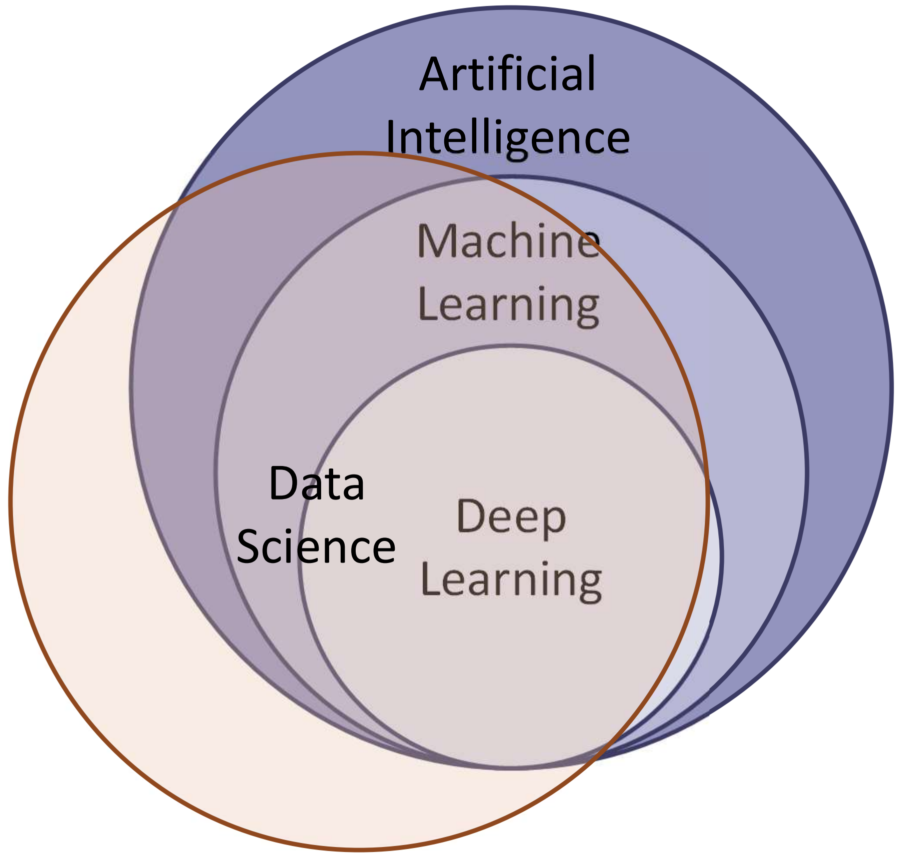
- Making machines learn!
- Using statistical models and algorithms to perform a specific task by learning data patterns, without being explicitly programmed
- **Generalization** to new unseen examples.

Machine Learning: Main Recipe

- A pattern exist …
- We do not know it mathematically!
- We have data on it :)

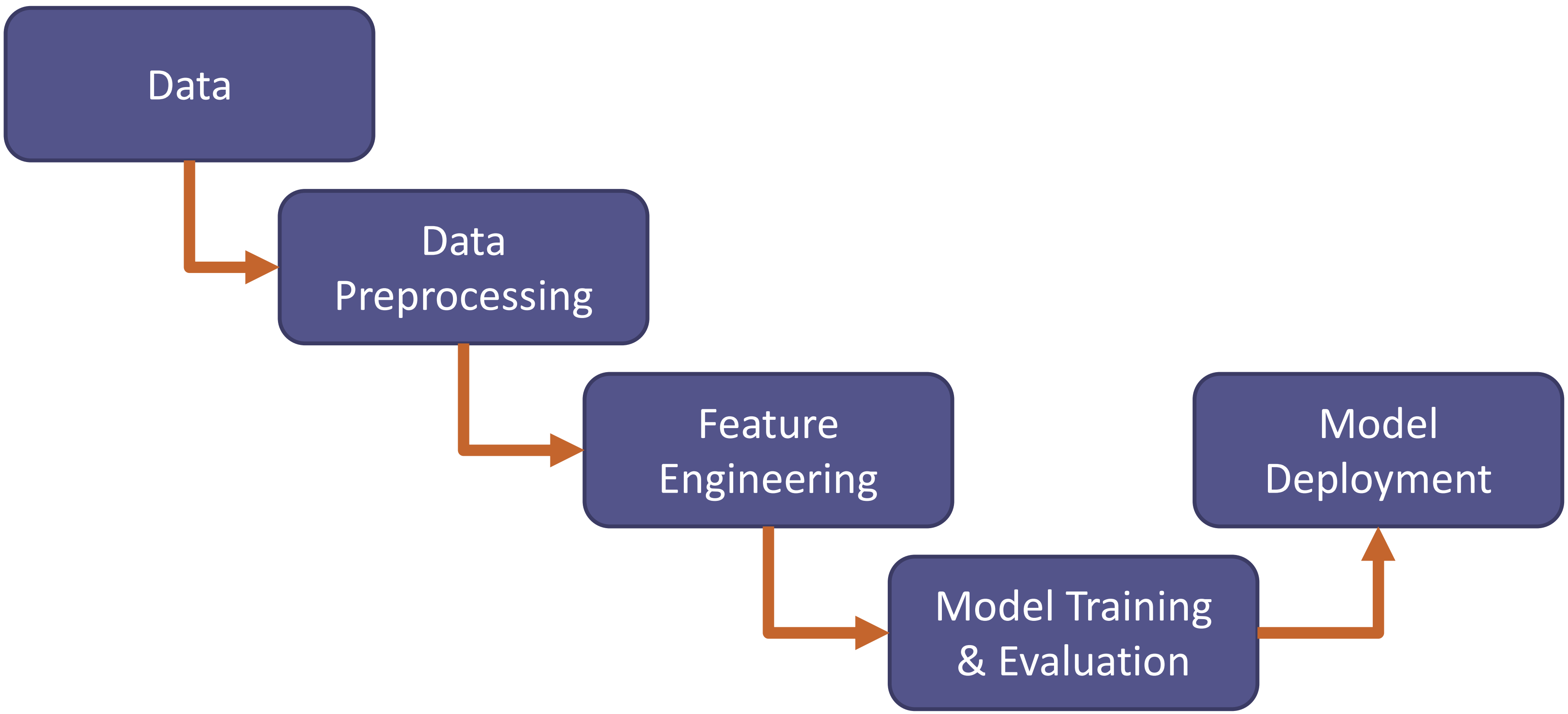
Machine Learning: Role of Data

What's the scope?



Machine Learning: Main Steps

- Typical steps:



Main ML Problems

- Supervised Learning
  - Classification
  - Regression
- Unsupervised Learning
  - Density Estimation
  - Generative Modeling
  - Clustering
  - Dimensionality Reduction
- Reinforcement Learning
  - Multi-armed Bandit

Supervised Learning vs. Unsupervised Learning

- Supervised learning

Given: Training set

Labeled set of N input-output pairs $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$

Goal: Learning a mapping from x to y

Supervised Learning vs. Unsupervised Learning

- Supervised learning

  Given: Training set

  Labeled set of N input-output pairs $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$

  Goal: Learning a mapping from x to y

- Unsupervised learning

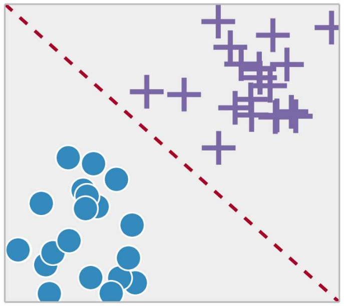
  Given: Training set

$$D = \{(x^{(i)})\}_{i=1}^{N}$$

  Goal: Revealing structure in the observed data and finding groups or intrinsic structures in the data

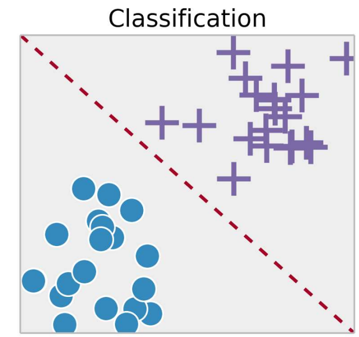
Supervised Learning: Classification vs. Regression

- Classification: predict a discrete target variable e.g. $y \in \{1, 2, \ldots, C\}$

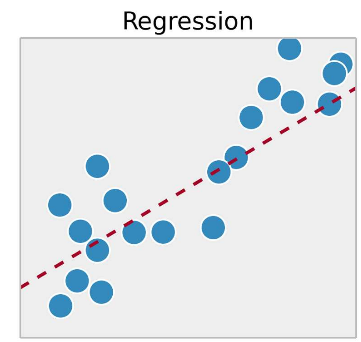


Supervised Learning: Classification vs. Regression

- Classification: predict a discrete target variable e.g. $y \in \{1, 2, \ldots, C\}$

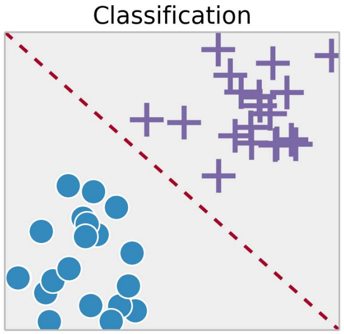


- Regression: predict a continuous target variable e.g. $y \in [0, 1]$

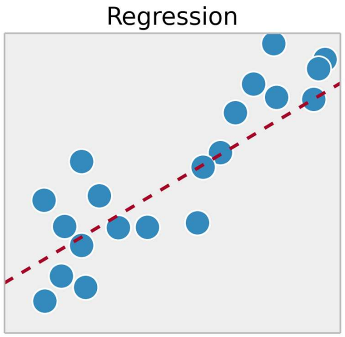


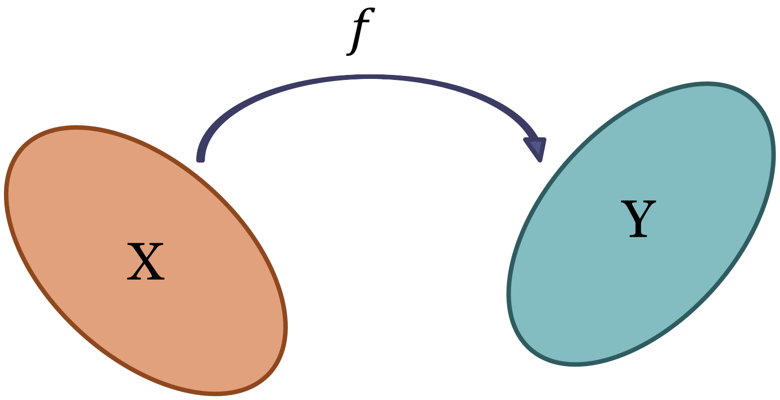
Supervised Learning: Classification vs. Regression

- Classification: predict a discrete target variable e.g. $y \in \{1, 2, \ldots, C\}$



- Regression: predict a continuous target variable e.g. $y \in [0, 1]$



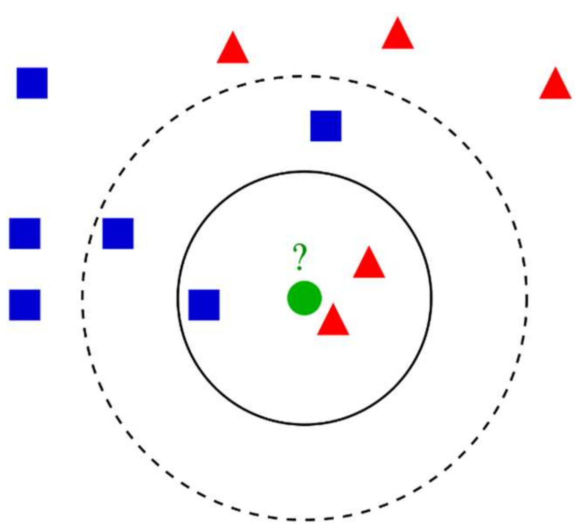


Classification

- A function $f\ : R^n \rightarrow \{1, \dots, k\}$ specifies which of k categories an input vector x belongs to.

Classification

- A function $f : R^n \rightarrow \{1, \ldots, k\}$ specifies which of k categories an input vector x belongs to.
- Case Study: KNN (K Nearest Neighbors)
  - Stores all training cases and classify new cases based on similarity measure (like Euclidean distance)



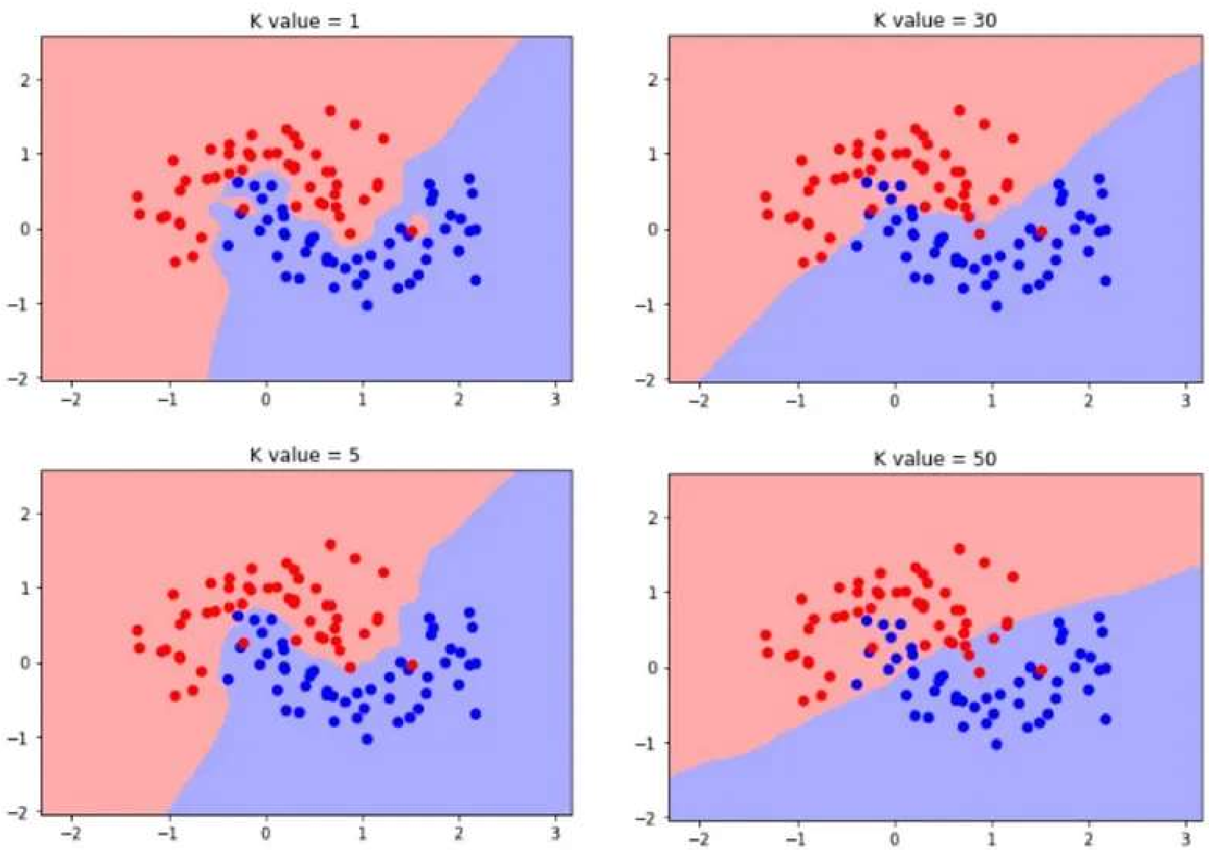
Classification: KNN

- Given
  - Training data $\{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, y^{(n)})\}$ are simply stored.
  - Test sample: $\boldsymbol{x}$

- To classify $\boldsymbol{x}$:
  - Find k nearest training samples to $\boldsymbol{x}$
  - Identify the number of samples k_j belonging to class $\mathcal{C}_j$
  - Assign $\boldsymbol{x}$ to the class $\mathcal{C}_{j^*}$ where $j^* = \underset{j=1,\ldots,c}{\operatorname{argmax}}\, k_j$

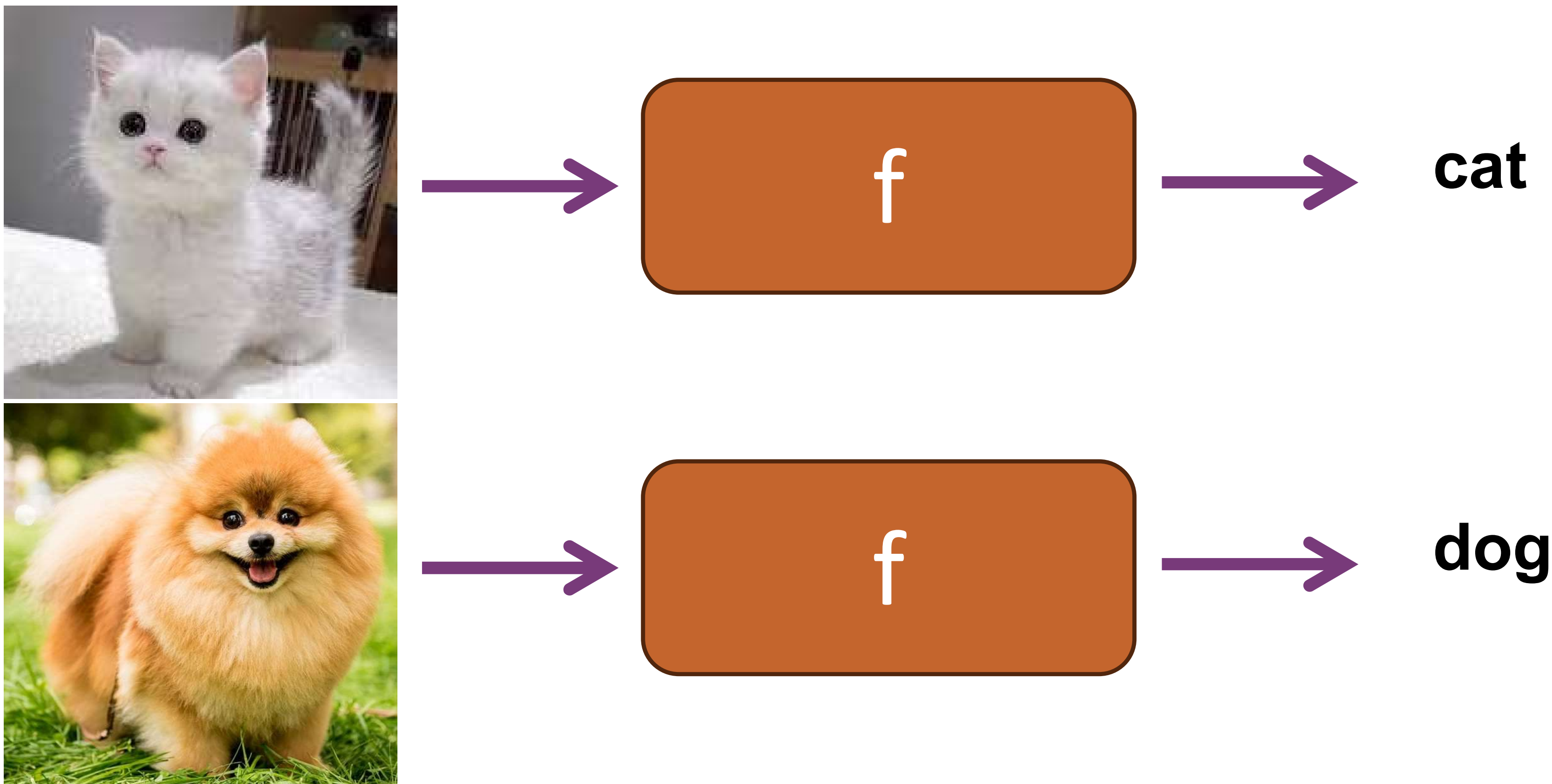
Classification: KNN

- Effect of K on decision boundaries



Classification

More advanced applications:



Object Recognition

Regression

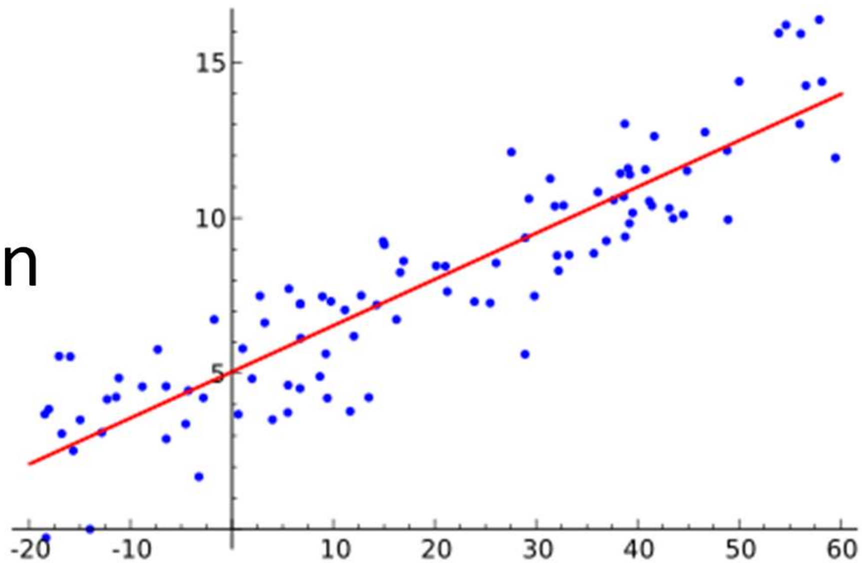
- A function $f : R^n \rightarrow R$ that maps an input vector x to a continues value y.

Regression

- A function $f: R^n \rightarrow R$ that maps an input vector x to a continues value y.

- Case study: Linear Regression

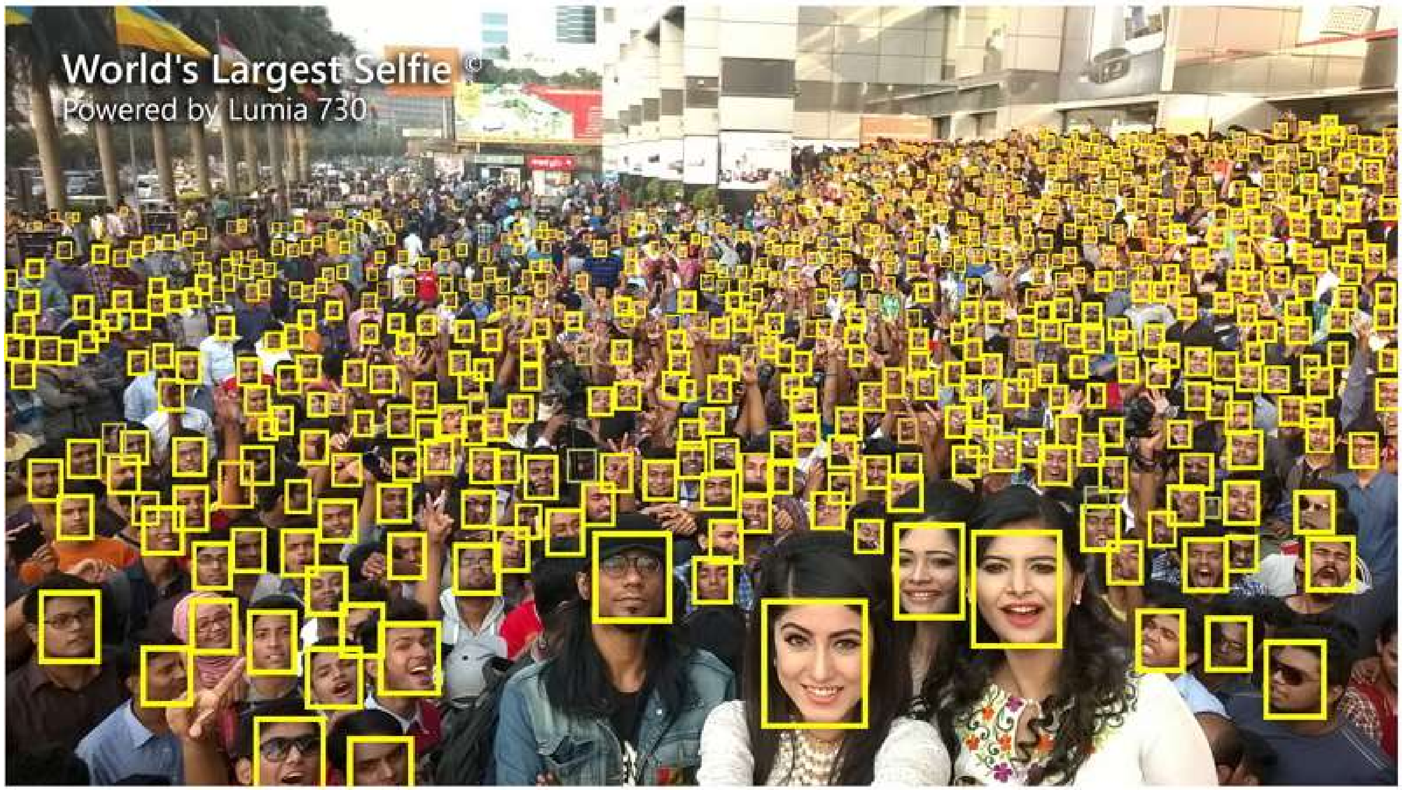
$$f(x;w) = \mathrm{w}_0 + \mathrm{w}_1\mathrm{x}$$



$w = [w_0, w_1]$: Parameters that be estimated during optimization

Regression

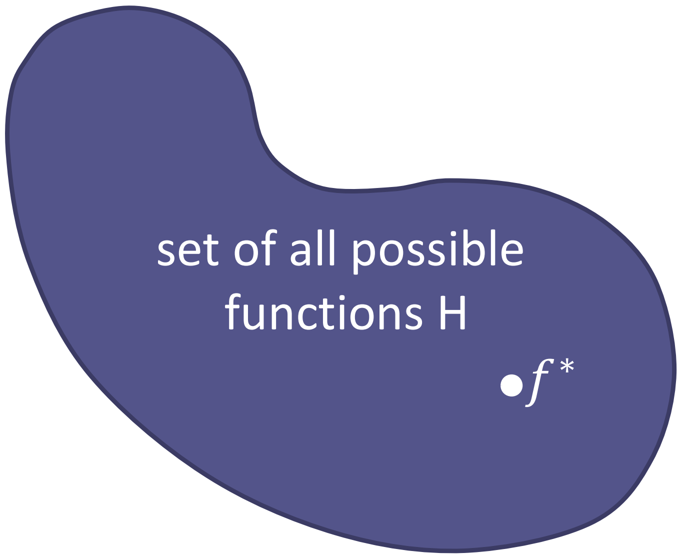
- More advanced applications:



Object Detection

Hypothesis Class and Inductive Bias

- The aim of supervised learning is to find f^* (best solution) from **hypothesis space** (e. g. the set of all possible functions)

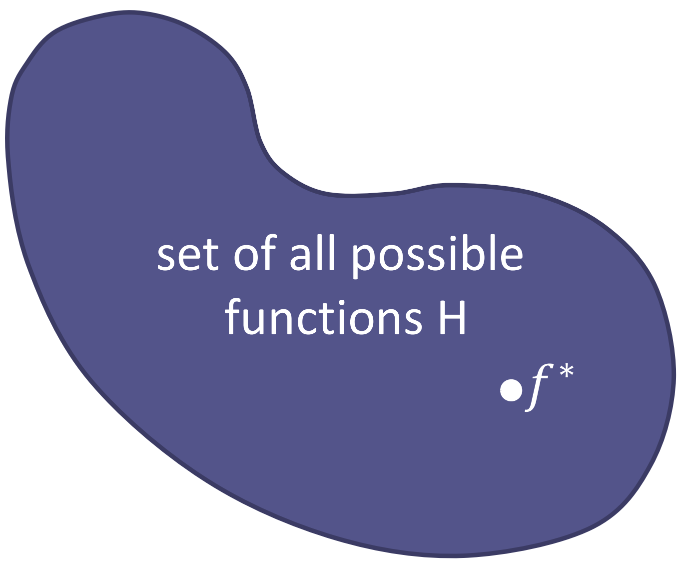


Hypothesis Class and Inductive Bias

- The aim of supervised learning is to find f^* (best solution) from **hypothesis space** (e. g. the set of all possible functions)
- Example:

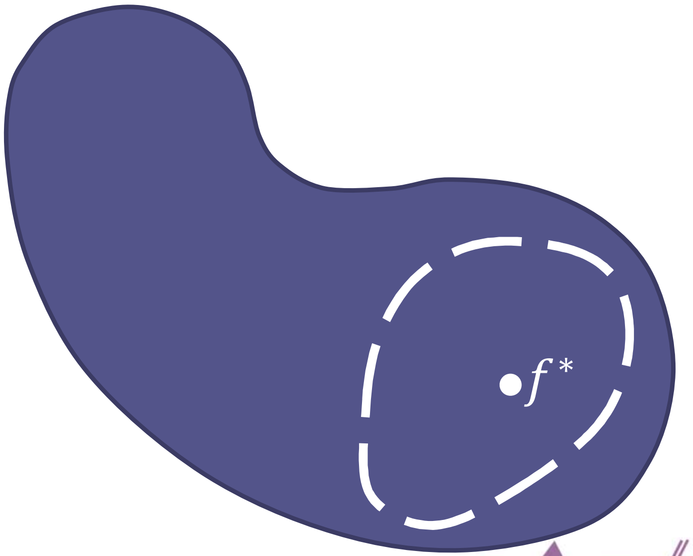
In linear regression the hypothesis space include all possible functions with the form of

$$f(x; w_0, w_1) = w_0 + w_1 x$$



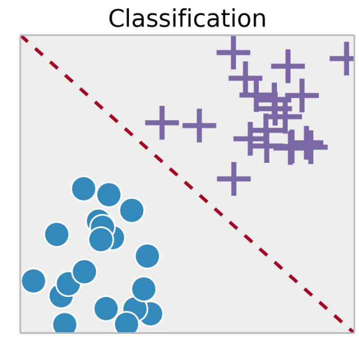
Hypothesis Class and Inductive Bias

- **Inductive bias** is the set of assumptions that a learner uses to predict outputs of given inputs.
- Some times we use our knowledge about the nature of data to **restrict** the hypothesis space.

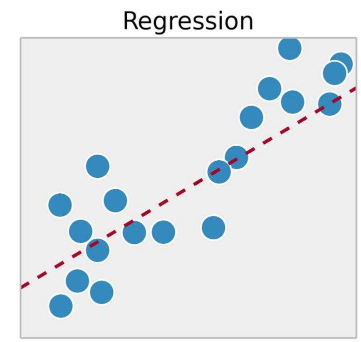


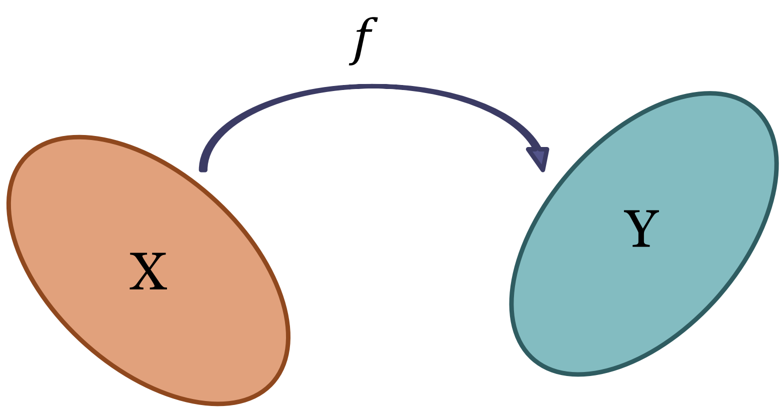
Supervised Learning (Recap)

- Classification: predict a discrete target variable e.g. $y \in \{1, 2, \ldots, C\}$



- Regression: predict a continuous target variable e.g. $y \in [0, 1]$





Unsupervised Learning

Unsupervised learning

Given: Training set

$$D = \{(x^{(i)})\}_{i=1}^{N}$$

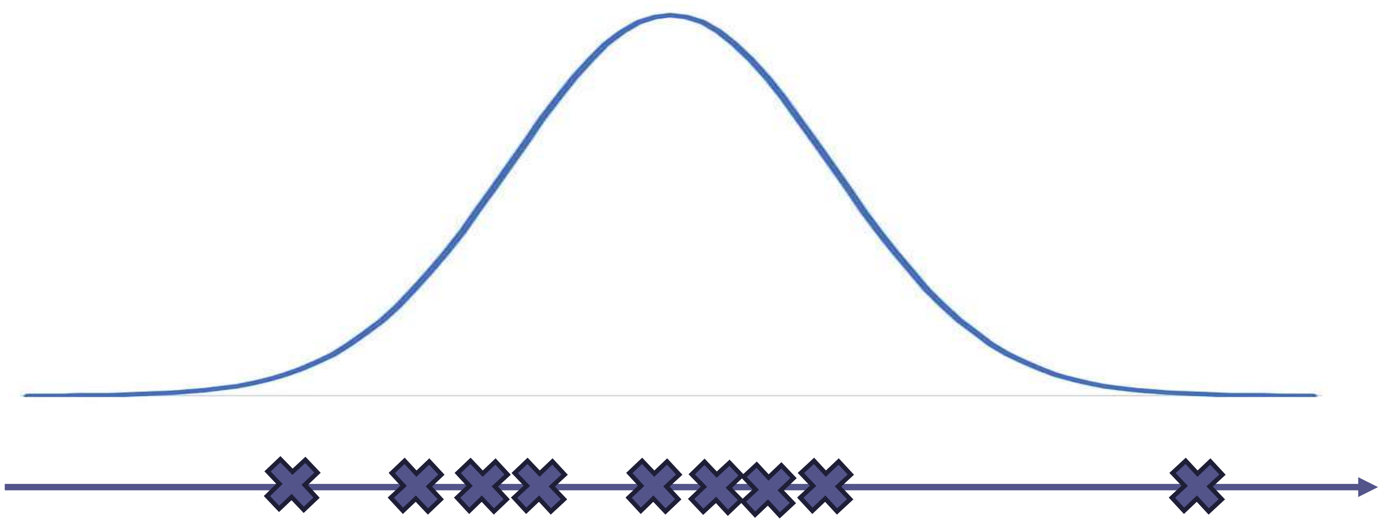
Goal: Revealing structure in the observed data and finding groups or intrinsic structures in the data

Main Approaches:

- Density Estimation
- Generative Modelling
- Clustering
- Dimensionality Reduction

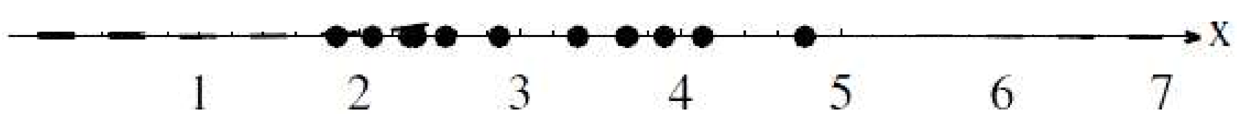
Density Estimation

- Estimating the probability density function $p(\boldsymbol{x})$, given a set of data points $\left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{N}$ drawn from it.



Density Estimation

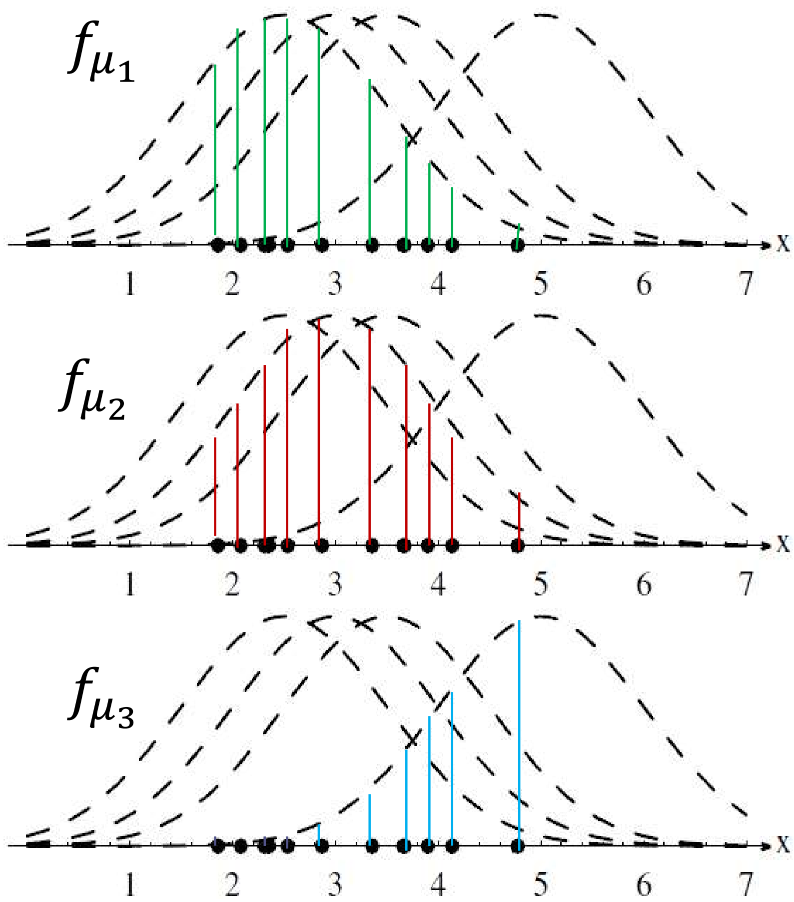
Example: Normal distribution



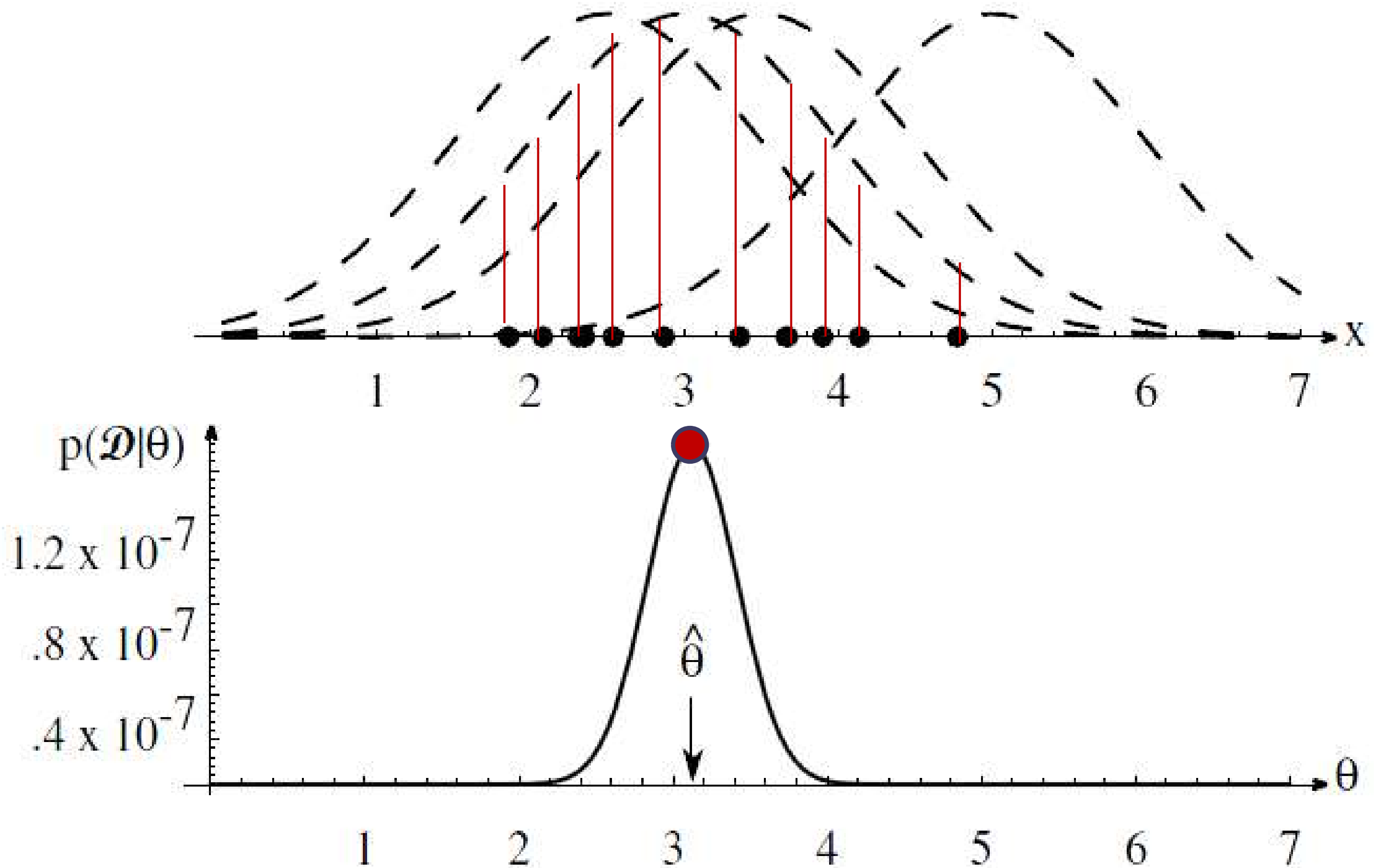
$$p_\mu(x) = N(x; \mu, 1)$$

$$p_\mu(x^{(1)}, x^{(2)}, \dots, x^{(N)}) = \prod_{i=1}^{N} p_\mu(x^{(i)})$$

Density Estimation: Normal distribution

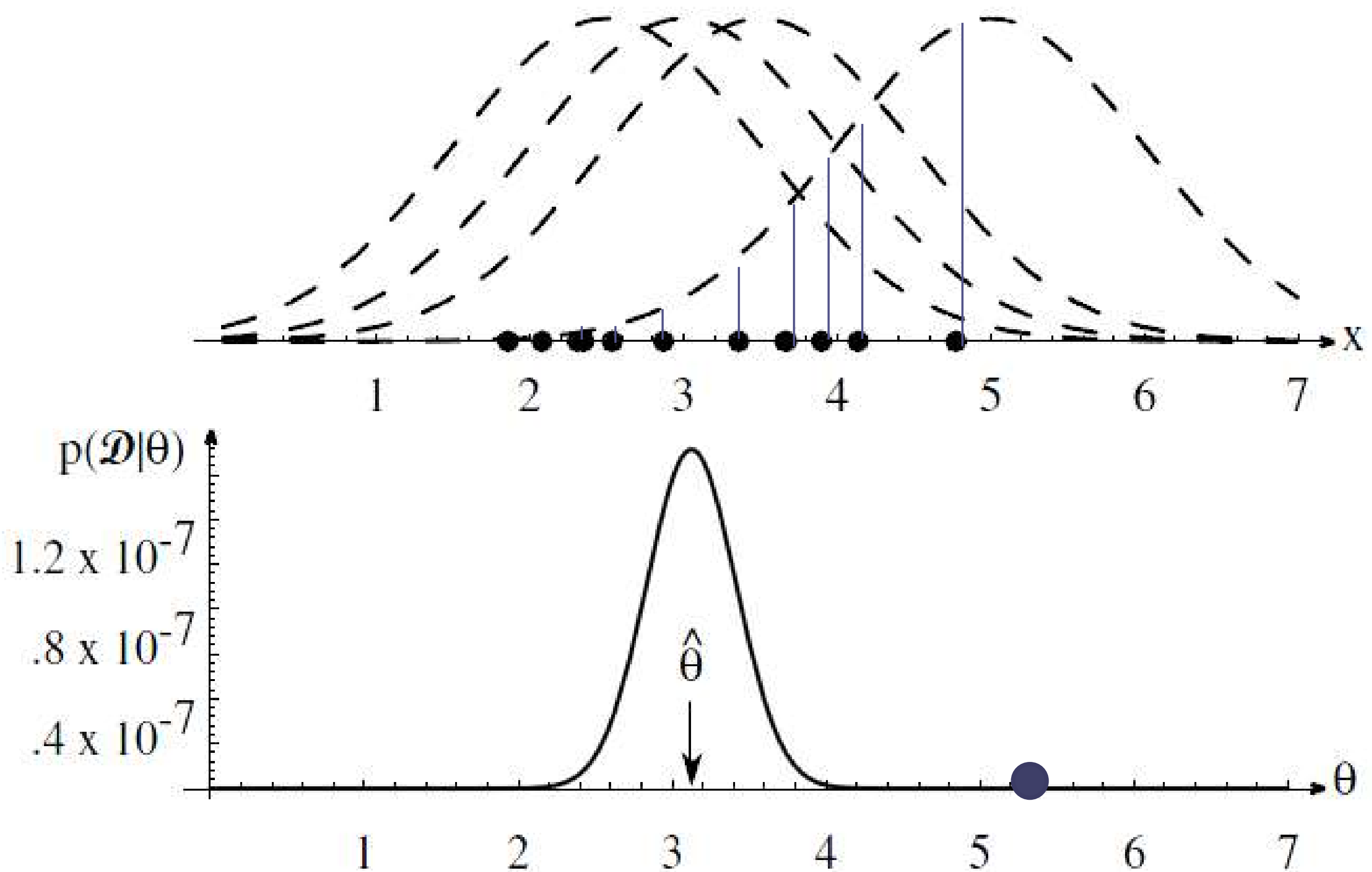


Density Estimation: Normal distribution



$\hat{\theta}$ best agrees with the observed samples

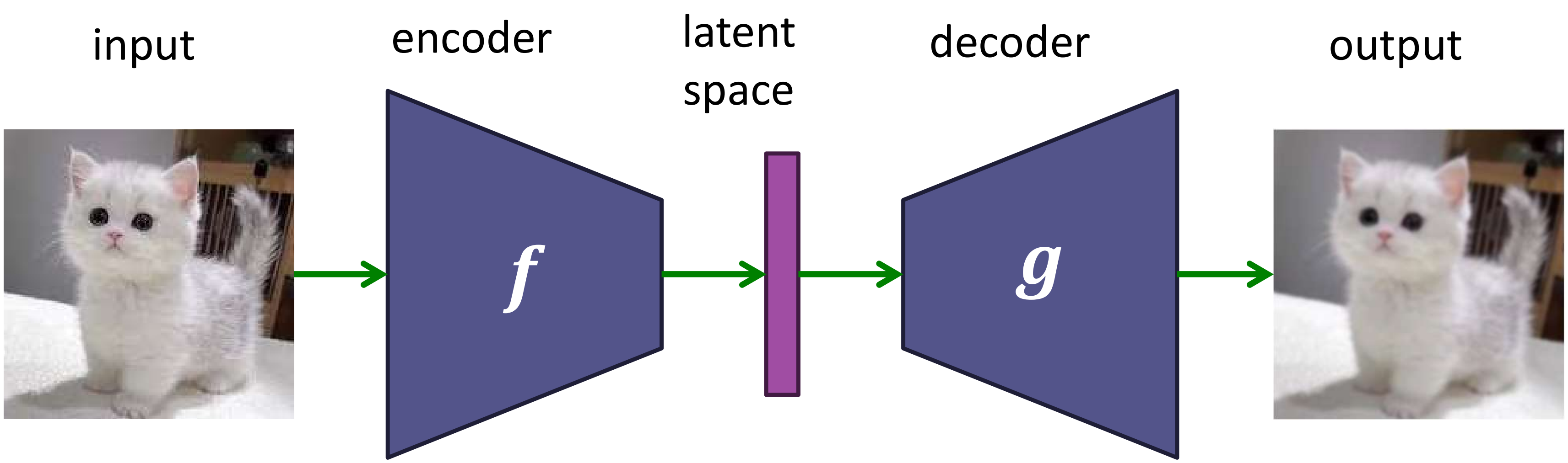
Density Estimation: Normal distribution



$\hat{\theta}$ best agrees with the observed samples

Density Estimation

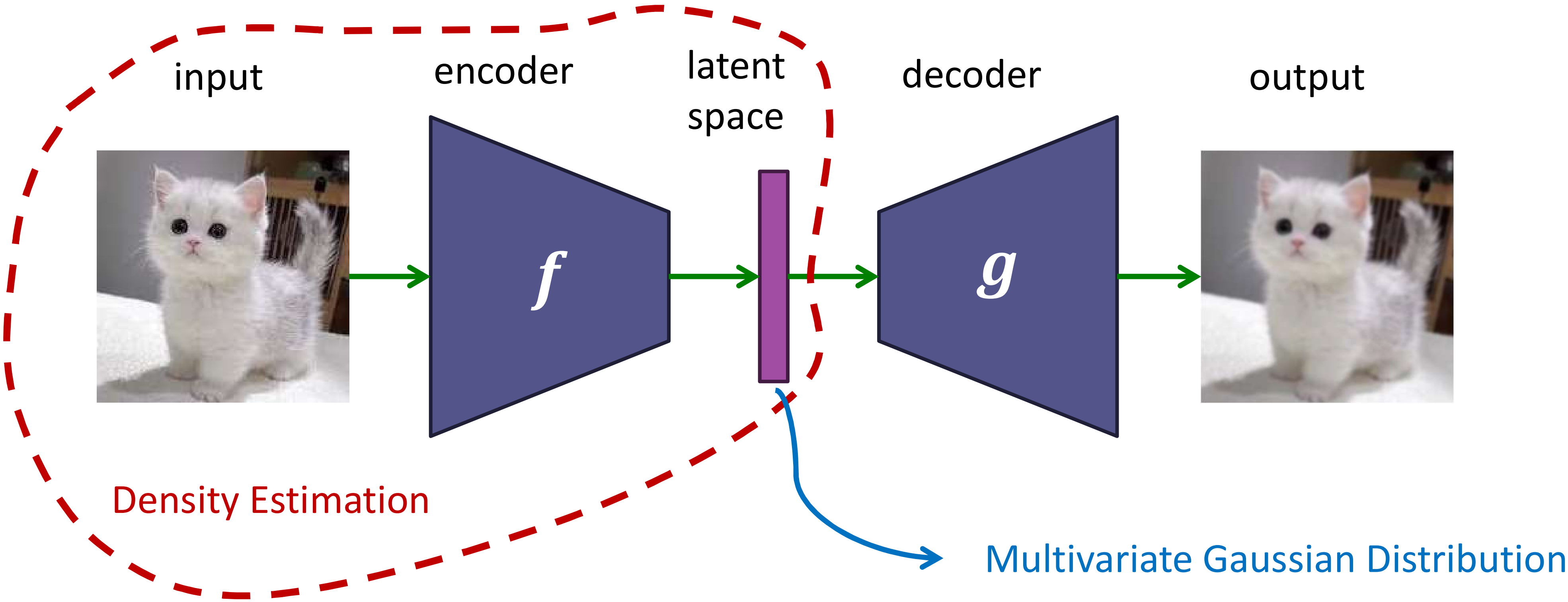
More sophisticated applications:



Variational Autoencoder (VAE)

Density Estimation

More sophisticated applications:

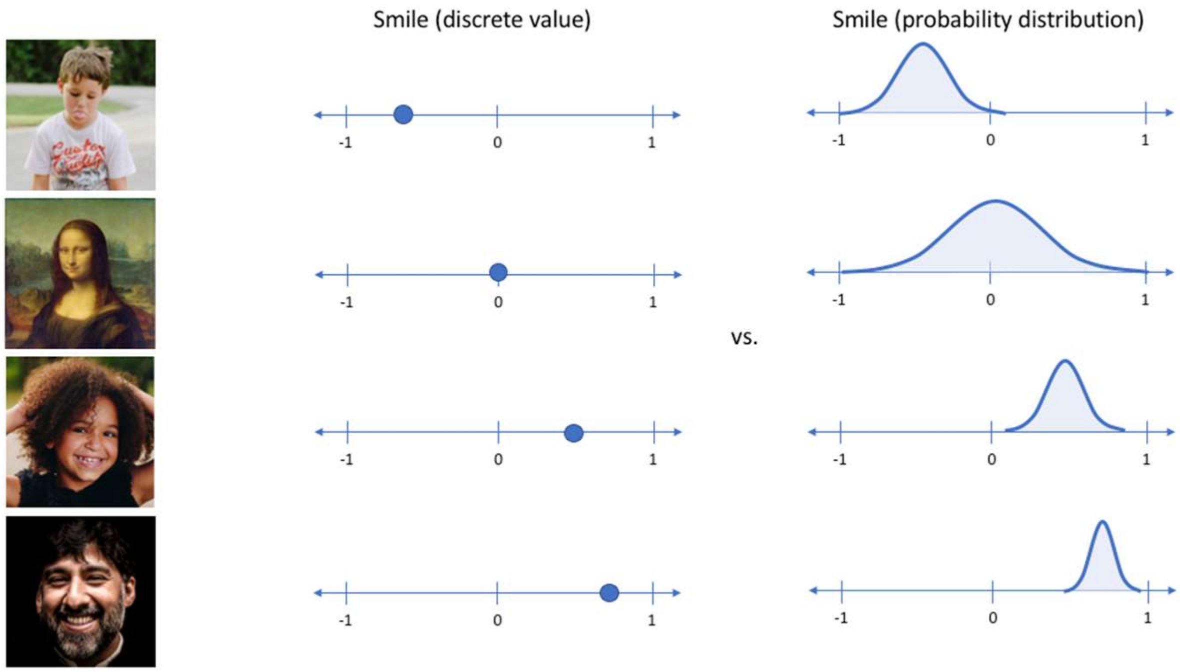


Variational Autoencoder (VAE)

Density Estimation

More sophisticated applications:

latent variable value



Variational Autoencoder (VAE)

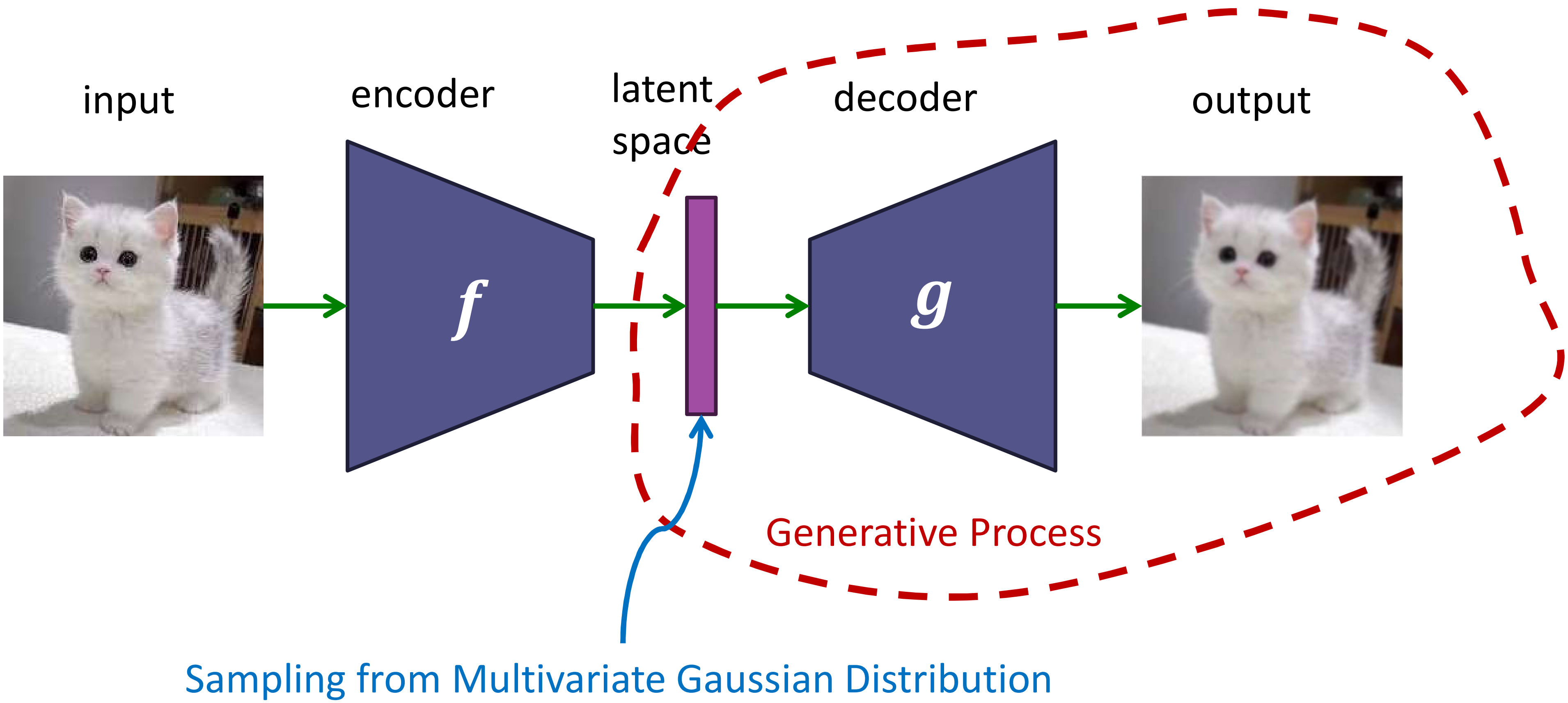
Generative Modeling

- Sampling new data points based on observed data distribution.

- This can be achieved with / without explicitly learning density function $p(\boldsymbol{x})$.

Generative Modeling:

More sophisticated applications (explicit density estimation)



Variational Autoencoder (VAE)

Generative Modeling:

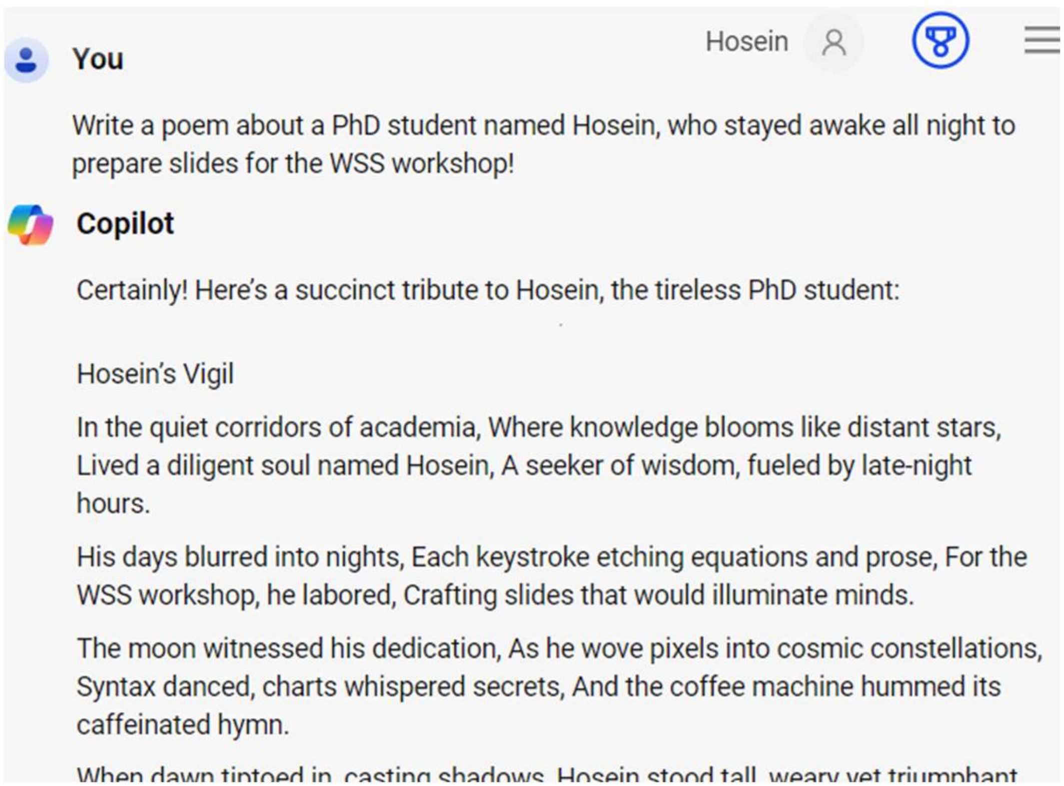
More sophisticated applications

- Generative Adversarial Networks (implicit density estimation)
- Diffusion Models

Generative Modeling:

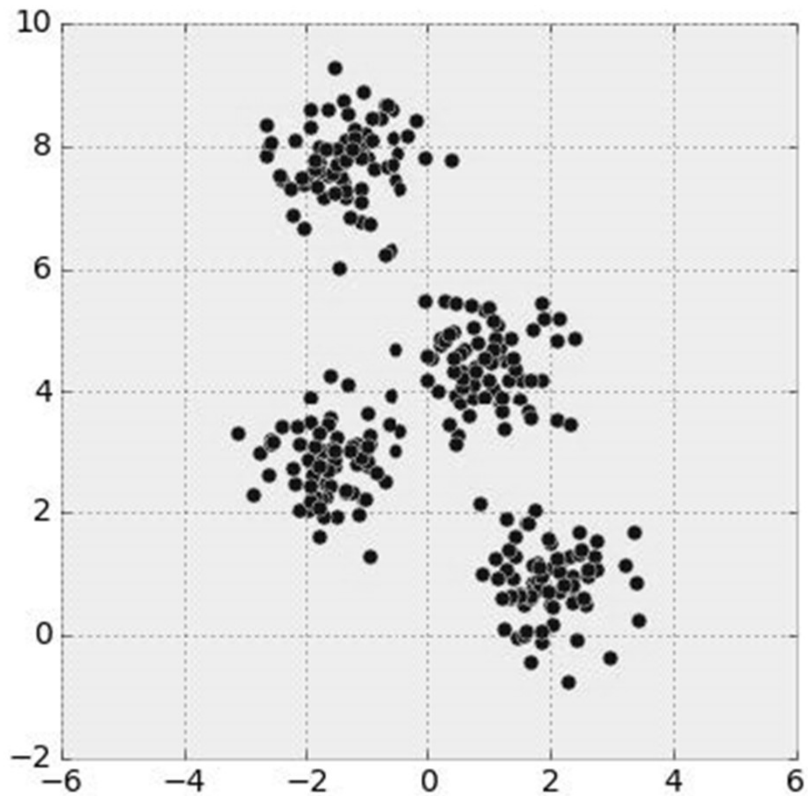
More sophisticated applications

- Large Language Models (LLMs)
- Generative pre-trained transformers (GPT)



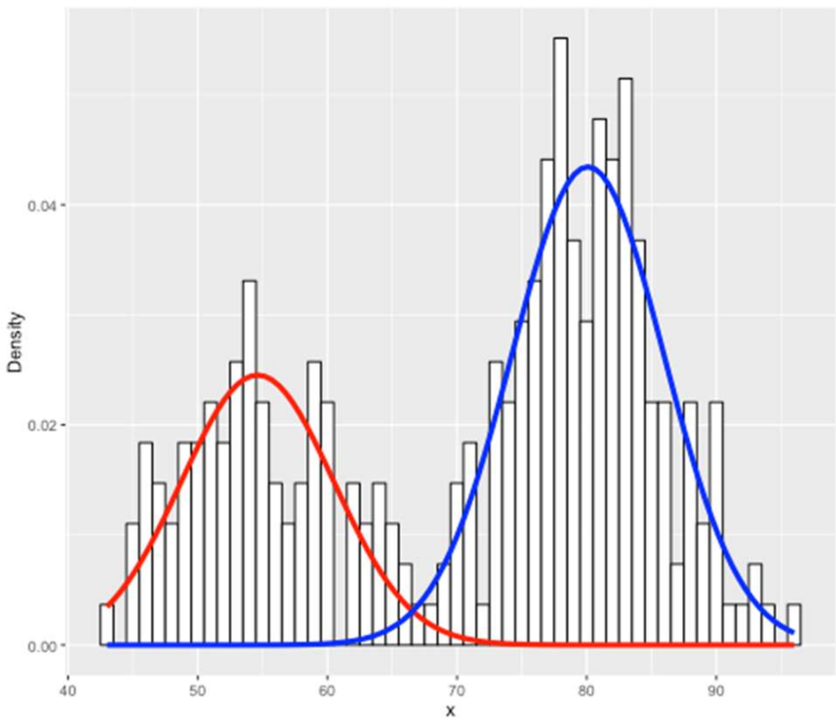
Clustering

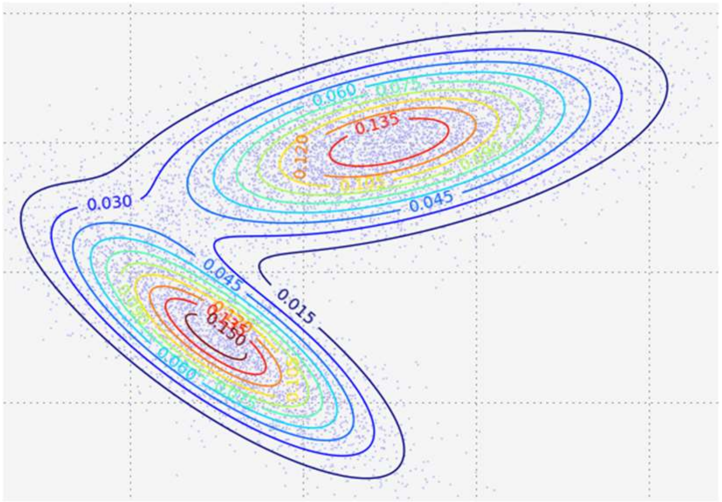
- A technique to assign each point into a specific group.



Clustering

- A technique to assign each point into a specific group.





Gaussian Mixture Model

Clustering: K-means Algorithm

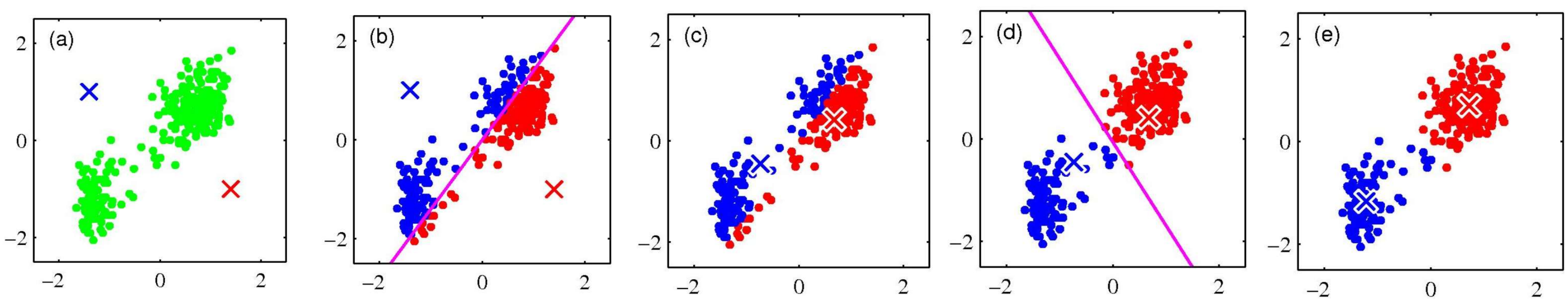
K-means Algorithm:

1. Choose number of clusters K.
2. Pick K random points as cluster centers (centroid)
3. Compute the distance between data points and all centroids
4. Assign each data point to the closest centroid
5. Compute the centroids for the clusters (by averaging)
6. Iterate steps 3-5 until convergence (no centroid change)

Clustering: K-means Algorithm

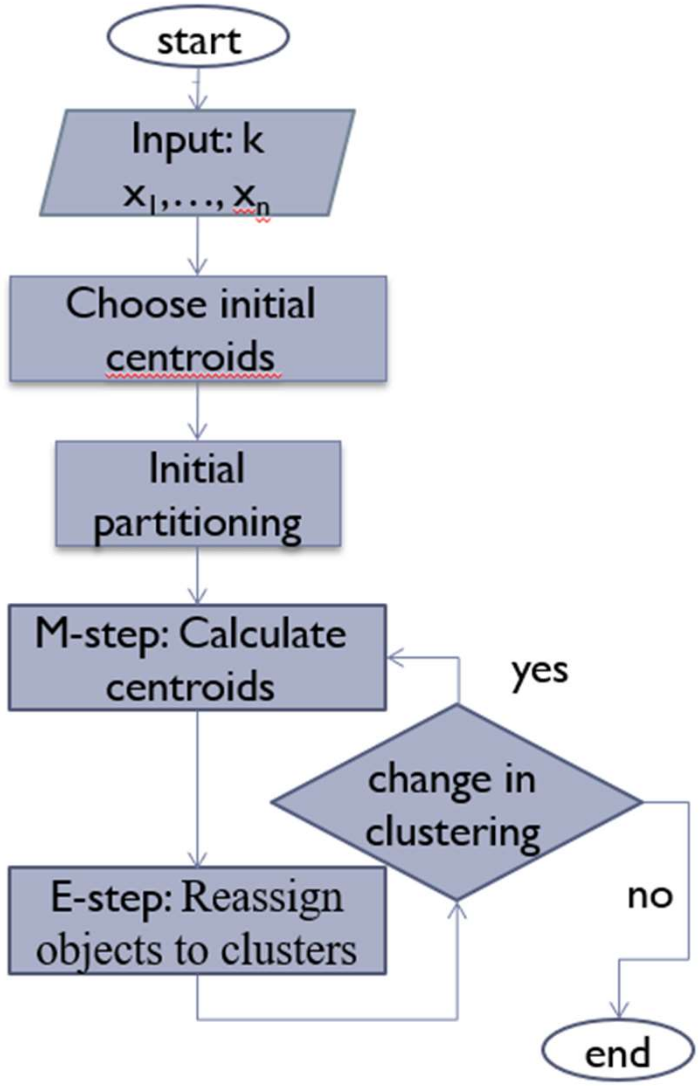
K-means Algorithm:

1. Choose number of clusters K.
2. Pick K random points as cluster centers (centroid)
3. Compute the distance between data points and all centroids
4. Assign each data point to the closest centroid
5. Compute the centroids for the clusters (by averaging)
6. Iterate steps 3-5 until convergence (no centroid change)



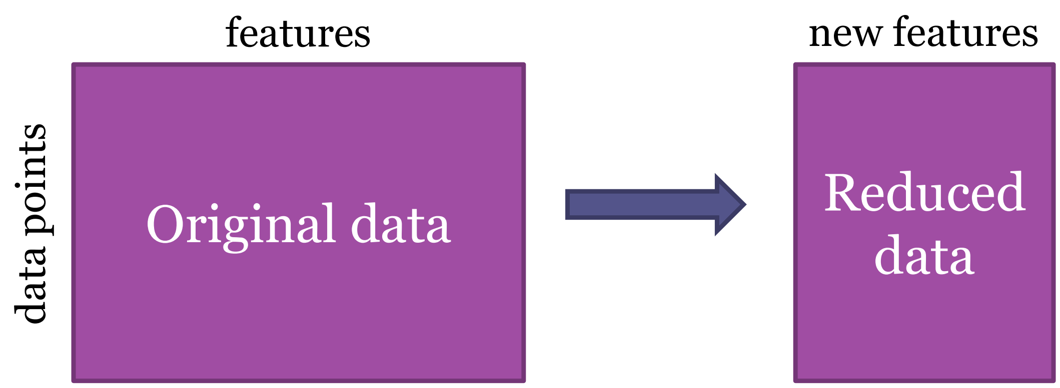
(2) (3-4) (5) (3-4) (6)

Clustering: K-means Algorithm



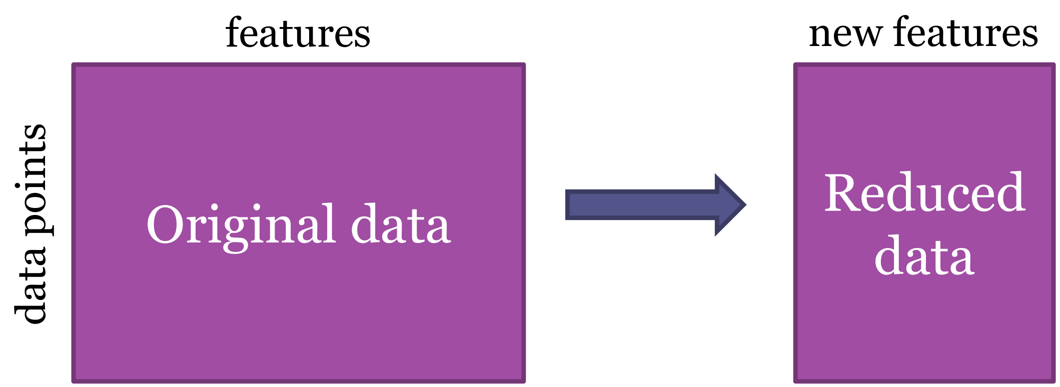
Dimensionality Reduction

- A technique to find a lower-dimensional representation of data features that preserves some of its properties.



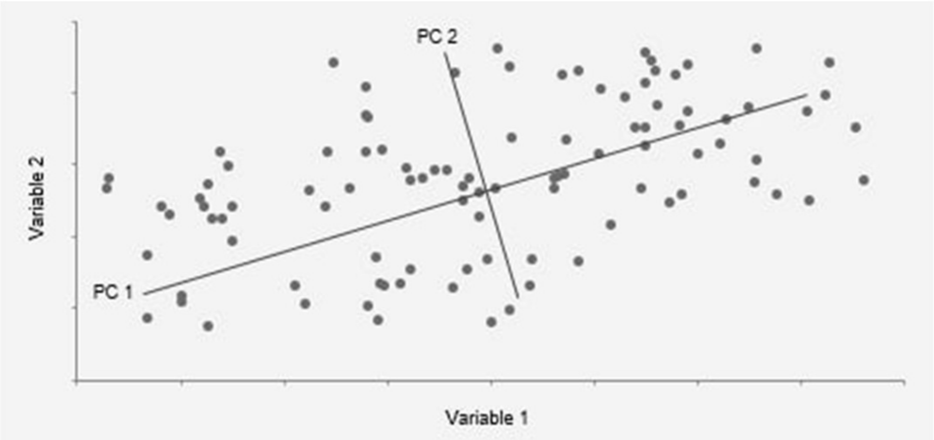
Dimensionality Reduction

- A technique to find a lower-dimensional representation of data features that preserves some of its properties.
- Motivations:
  - Computation
  - Visualization
  - Feature extraction



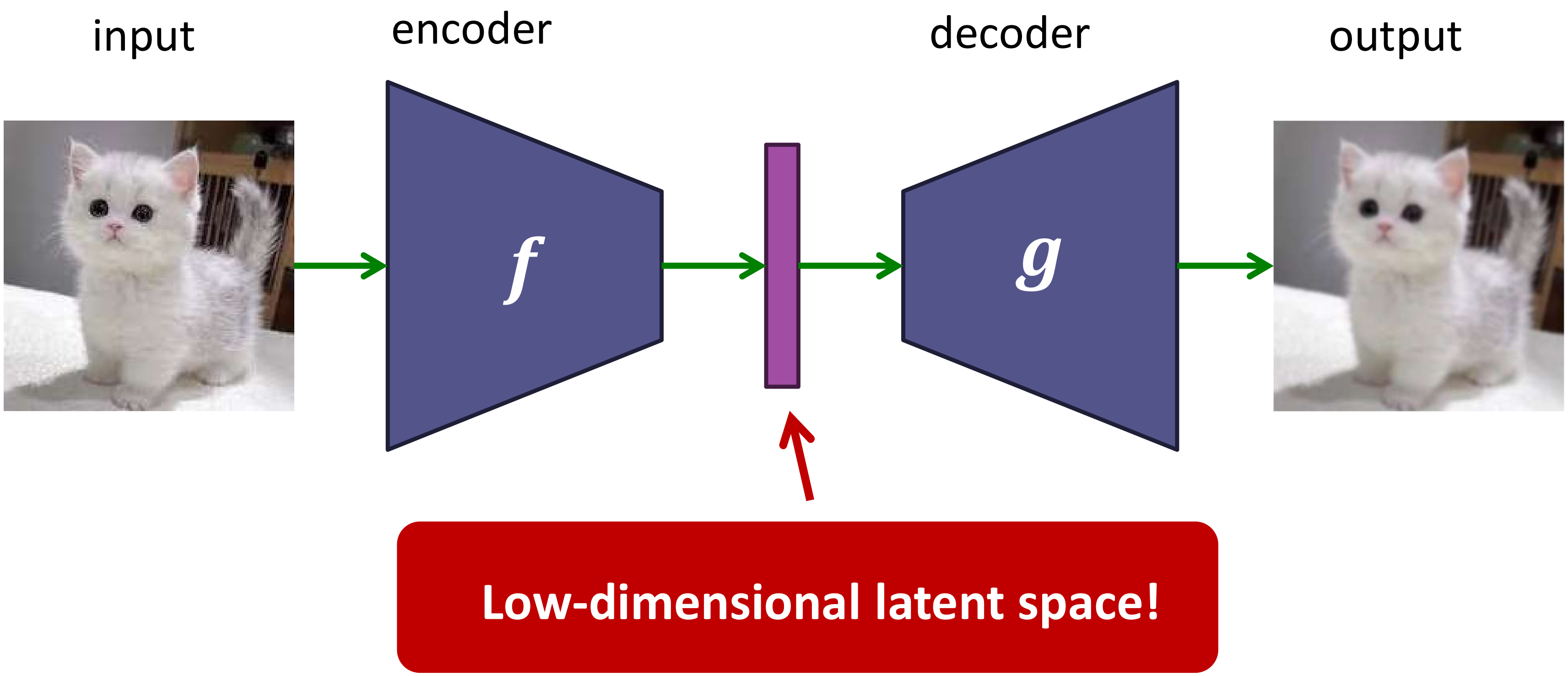
Dimensionality Reduction

- A technique to find a lower-dimensional representation of data features that preserves some of its properties.
- Case Study:
  Principal Component Analysis (PCA)



Dimensionality Reduction

More sophisticated methods:



Variational Autoencoder (VAE)

Unsupervised Learning (Recap)

Unsupervised learning

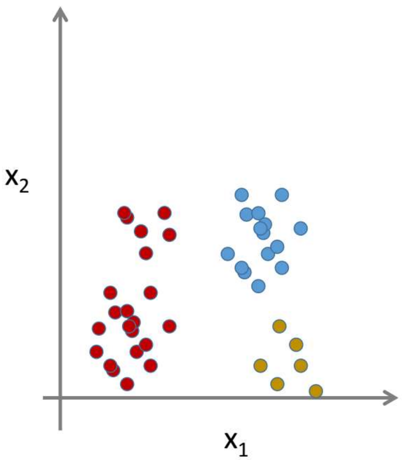
Given: Training set

$$D = \{(x^{(i)})\}_{i=1}^{N}$$

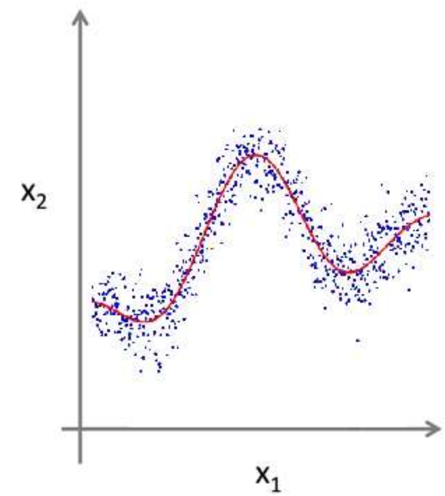
Goal: Revealing structure in the observed data and finding groups or intrinsic structures in the data

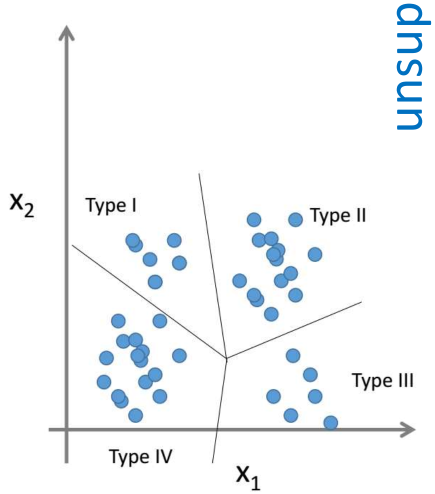
- Main Approaches:
  - Density Estimation
  - Generative Modelling
  - Clustering
  - Dimensionality Reduction

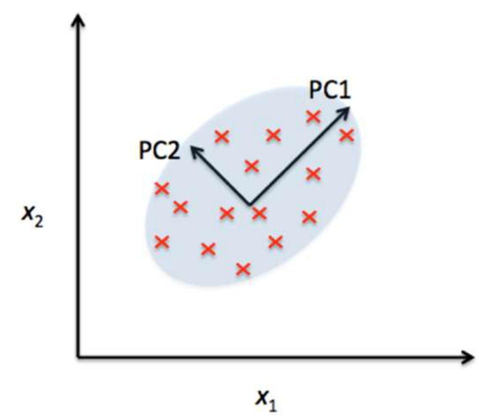
Supervised Learning vs. Unsupervised Learning



	discrete	Continues
supervised	**classification**	**regression**
unsupervised	**clustering**	**dimensionality reduction**





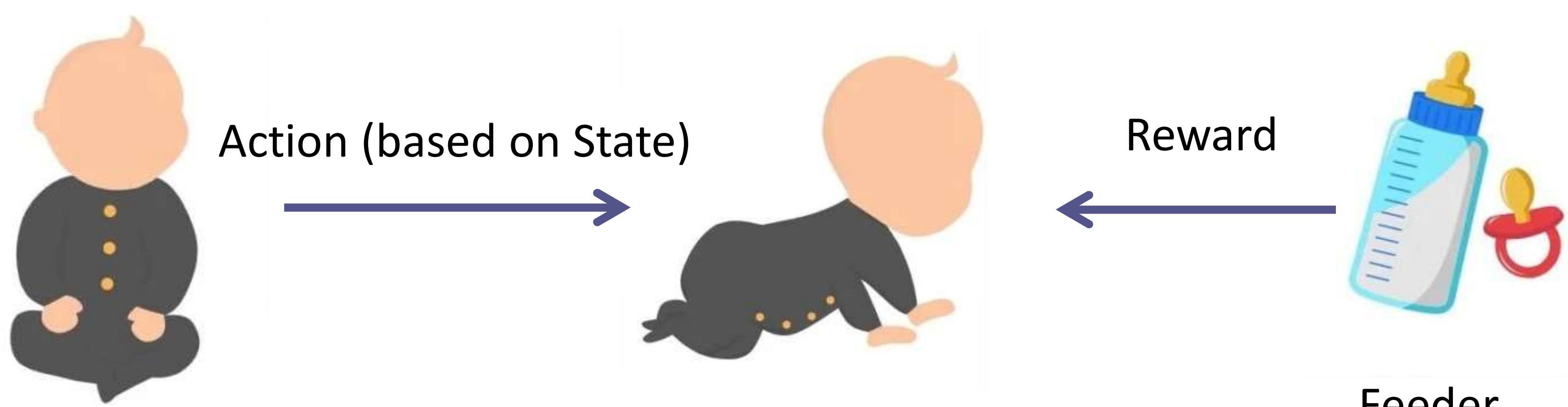


Outline

- Introduction and Motivation
- ML Problems
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning
- Loss Function and Optimization
- Generalization and Overfitting

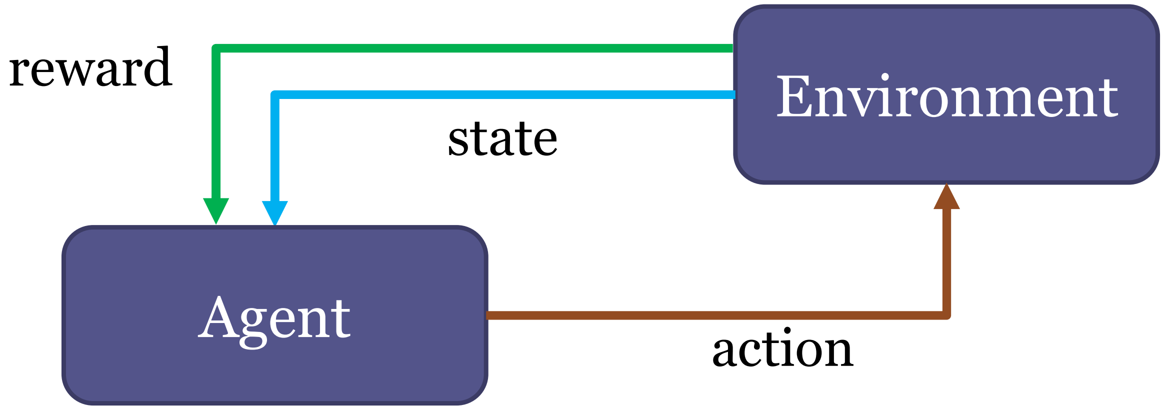
Reinforcement Learning

- Most natural way of learning
- Examples:
  - Baby movement
  - Investment



Reinforcement Learning

- Sequential decision making with (possibly delayed) rewards
- An agent learns appropriate actions (policy) based on environment feedbacks to maximize reward.



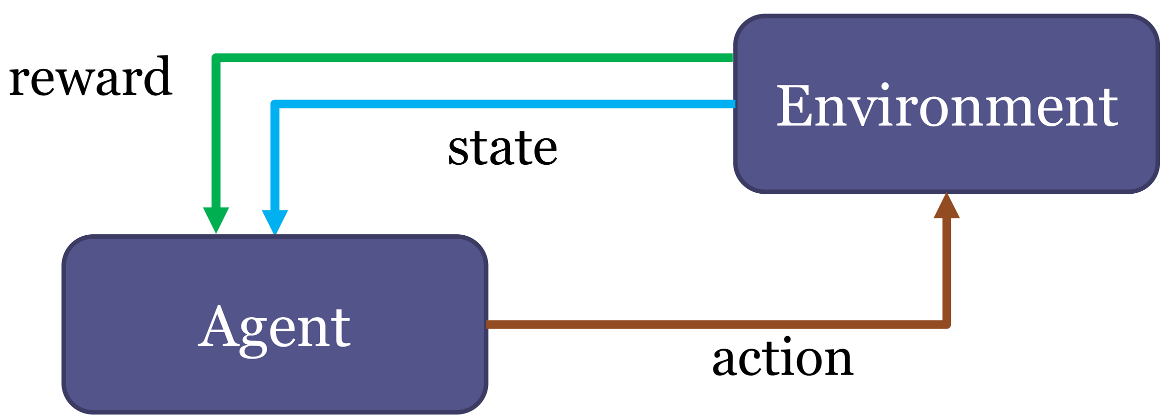
Reinforcement Learning

- Sequential decision making with (possibly delayed) rewards
- Data in supervised learning:

  (input, label)

- Data in reinforcement learning:

  (input, some output, a grade of reward for this output)



Reinforcement Learning

- State: Agent's observation from the world
- Environment model:
  - Transition probability $p(s_{t+1}|s_t, a_t)$
  - Reward function $R(s_t, a_t, s_{t+1})$
- Policy: Mapping from states to actions

$$\pi_\theta : S \to A$$

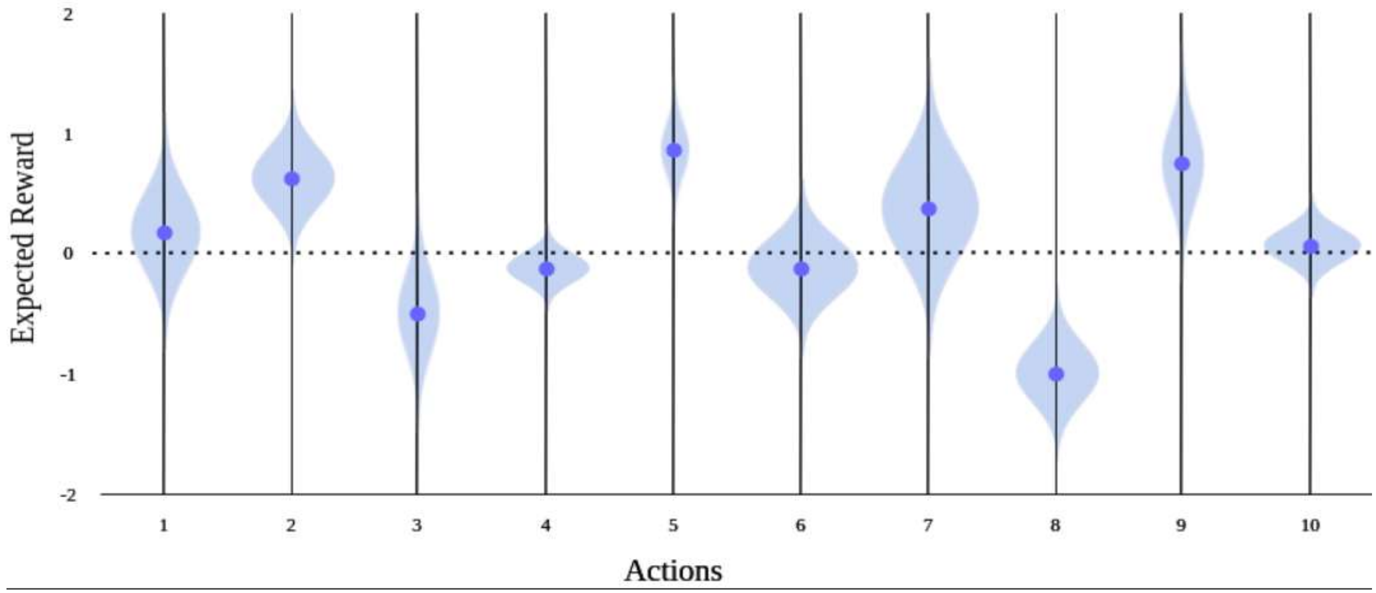
- Goal: Learning an optimal policy in order to maximize its long-term reward

Multi-Armed Bandit

Multiple bandits with unknown average rewards

Multi-Armed Bandit

- Finding the **best arm** (in the sense of expected reward) with minimum trial and error.
- Minimizing cumulative **regret**.
- **Exploration-exploitation** trade-off!

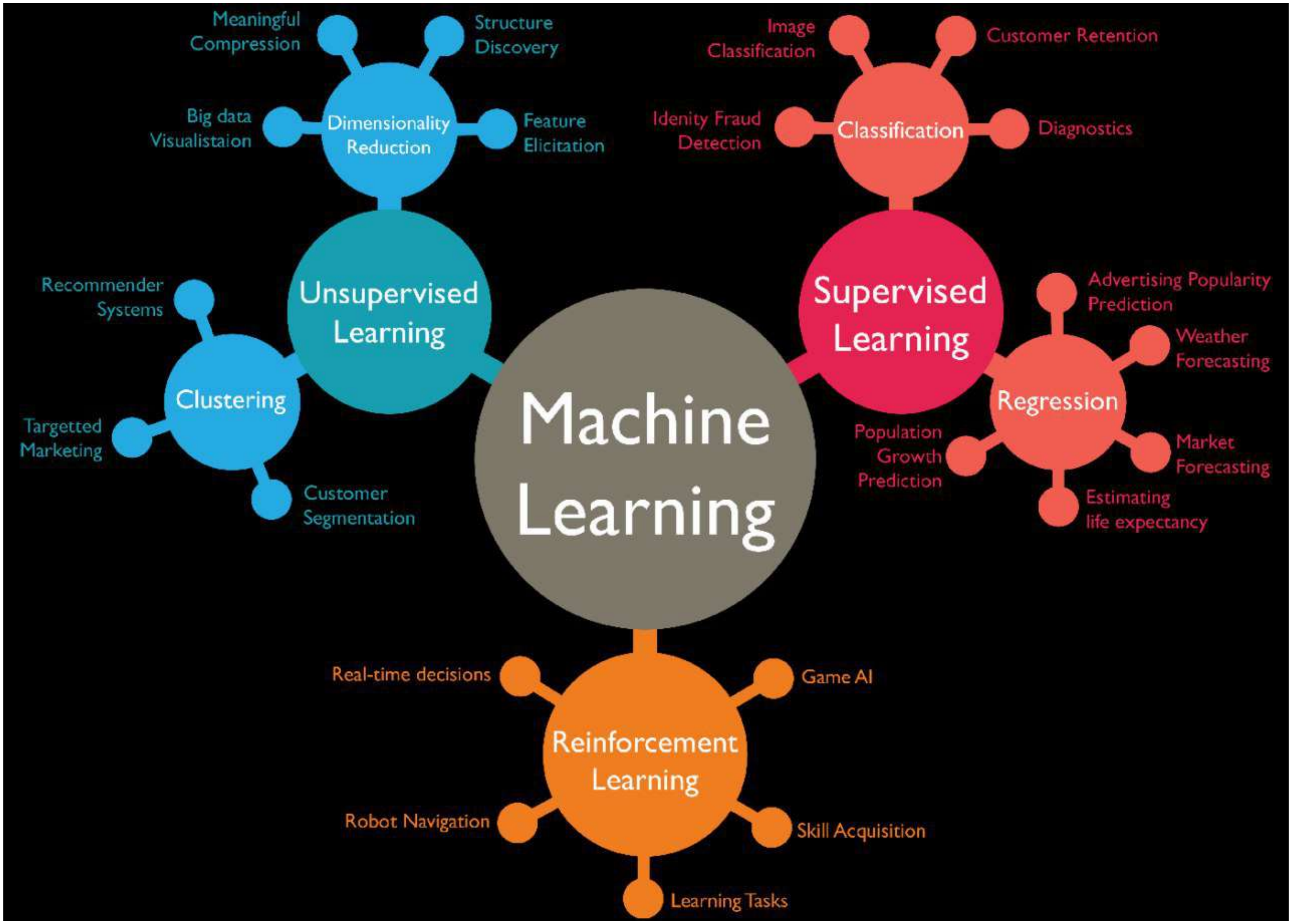


Multi-Armed Bandit

Applications:

- Online advertisement
- Recommender systems
- Clinical trials
- Mining
- Network (packet routing)

Primary ML Problems (Review)

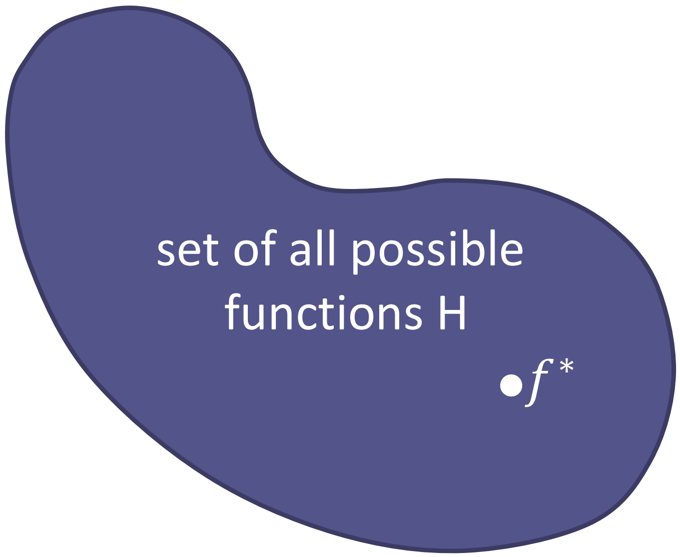


Hypothesis Class (Recap)

- The aim of supervised learning is to find f^* (best solution) from **hypothesis space** (e. g. the set of all possible functions)
- Example:

In linear regression the hypothesis space include all possible functions with the form of

$$f(x; w_0, w_1) = w_0 + w_1 x$$



Loss Function and Optimization

- **Loss Function**:

Quantifies how much undesirable is each parameter vector across the training data.

- **Optimization**:

Apply an **optimization** algorithm that finds the parameters that minimize the loss function.

Steps of Learning Procedure

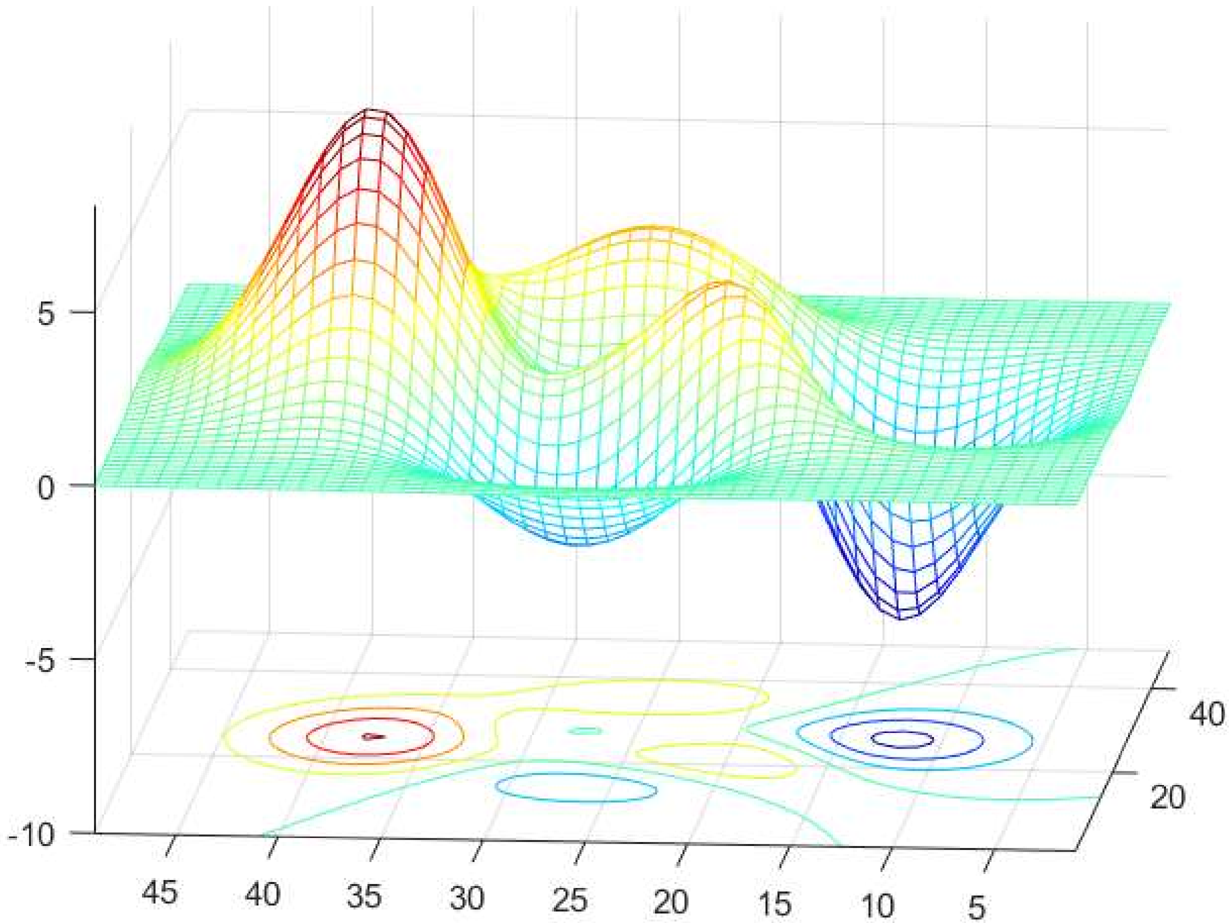
Typical steps of solving (supervised) learning problems:

- Select the **hypothesis space**:
- Define a **loss function** that quantifies how much undesirable is each parameter vector across the training data.
- Apply an **optimization** algorithm that efficiently finds the parameters that minimize the loss function.
- **Evaluate** the obtained model.

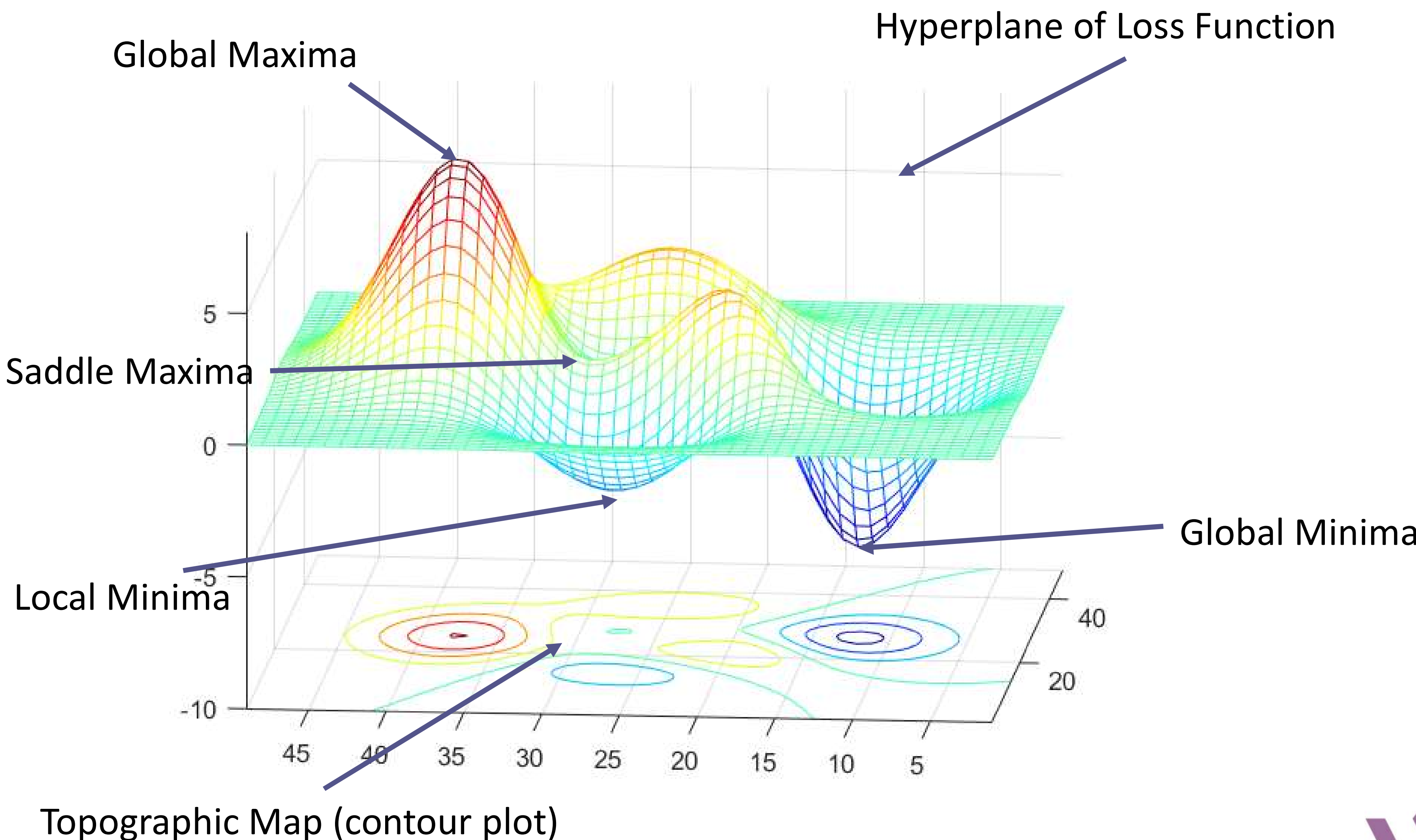
Loss Function

- Error: The **difference** between the actual outputs (e.g. **ground truth**) and the predicted outputs.
- The function that is used to compute this error is known as Loss Function $\boldsymbol{L}(.)$ or $\boldsymbol{J}(.)$.
- Generally, consists of **empirical risk term** and **regularization term.**

Loss Function: Loss Landscape

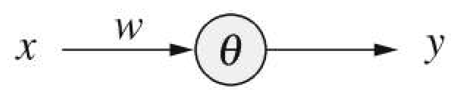


Loss Function: Loss Landscape



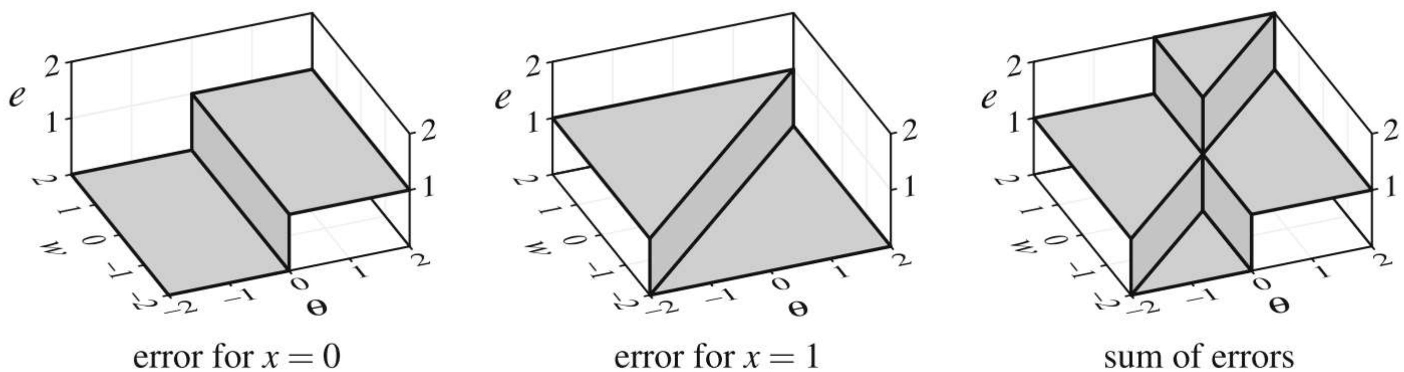
Loss Functions: Negation Problem

- Consider a threshold logic unit with a single input and training examples for the negation:



x	y
0	1
1	0

- Error of computing the negation w.r.t. the θ and w:



$$L = \sum_i L_i$$

Loss Functions in Supervised Learning

- Regression Losses
  - Mean Square Error (L2 Loss)

- Classification Losses
  - Hinge Loss (Multi class SVM)
  - Cross Entropy Loss

Mean Squared Error

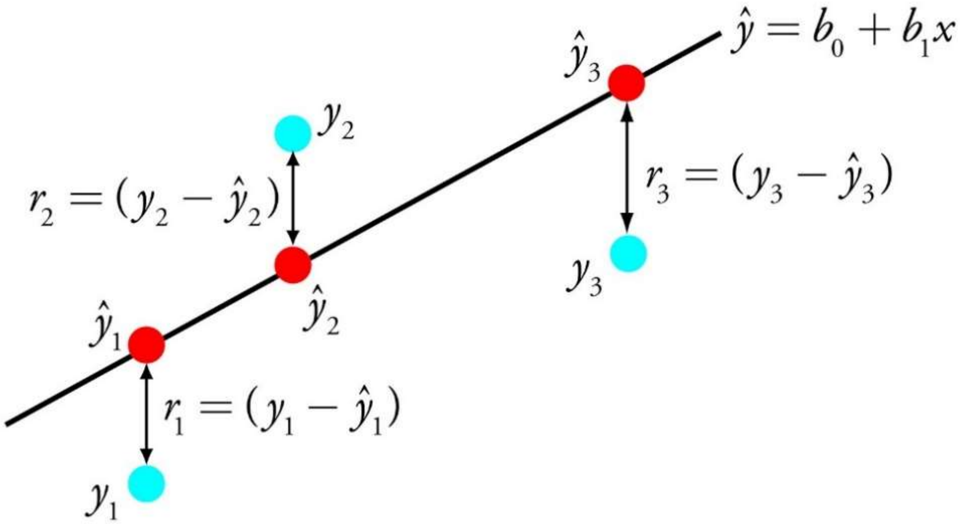
Mean Squared Error (MSE) loss function is widely used in regression problems.

$$J(w) = \sum_{i=1}^{N} (y^{(i)} - \underbrace{w^T x^{(i)}}_{\widehat{y^{(i)}}})^2$$

Mean Squared Error

Mean Squared Error (MSE) loss function is widely used in regression problems.

$$J(w) = \sum_{i=1}^{N}(y^{(i)} - \underbrace{w^T x^{(i)}}_{\widehat{y^{(i)}}})^2$$



Mean Squared Error

Mean Squared Error (MSE) loss function is widely used in regression problems.

$$J(w) = \sum_{i=1}^{N}(y^{(i)} - \underbrace{w^T x^{(i)}}_{\widehat{y^{(i)}}})^2$$

Goal: Find w^* which minimizes J(w):

$$w^* = argmin_w\, J(w)$$

Mean Squared Error

Mean Squared Error (MSE) loss function is widely used in regression problems.

$$J(w) = \sum_{i=1}^{N}(y^{(i)} - \underbrace{w^T x^{(i)}}_{\widehat{y^{(i)}}})^2$$

Goal: Find w^* which minimizes J(w):

$$\boxed{w^* = argmin_w\, J(w)}$$

Optimization

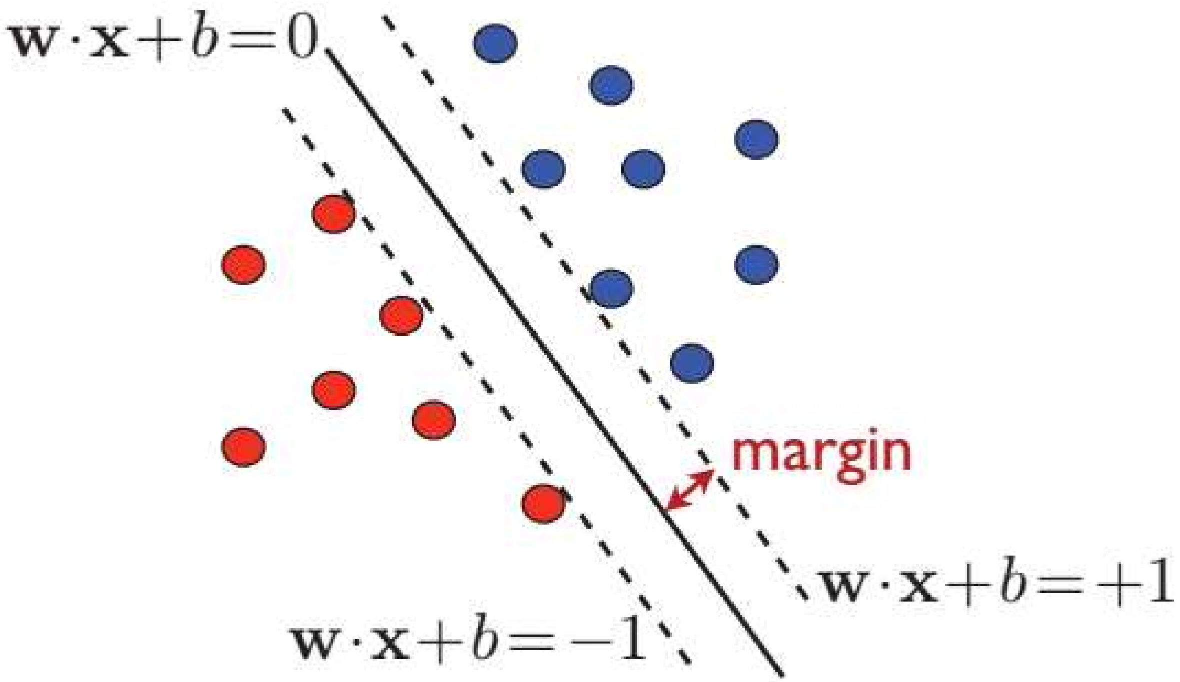
Loss Functions in Supervised Learning

- Regression Losses
  - Mean Square Error (L2 Loss)

- Classification Losses
  - Hinge Loss (Multi class SVM)
  - Cross Entropy Loss

Hinge Loss: SVM Classifier

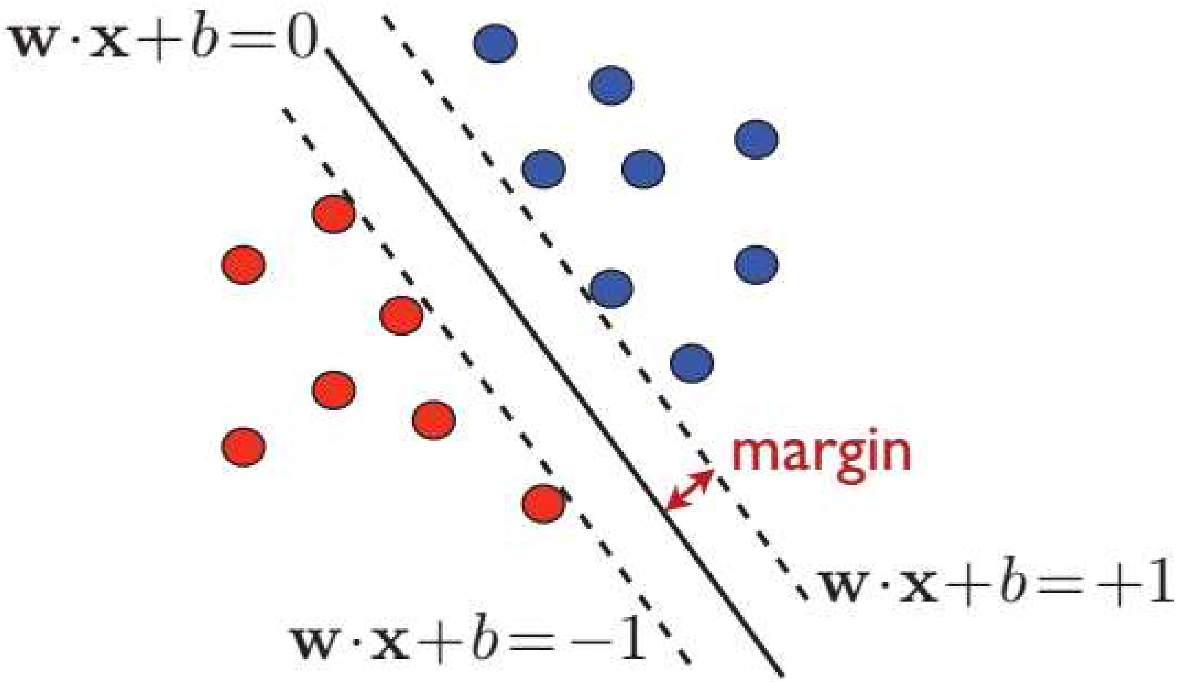
Support Vector Machine (SVM)



[Mohri]

Hinge Loss: SVM Classifier

Support Vector Machine (SVM)

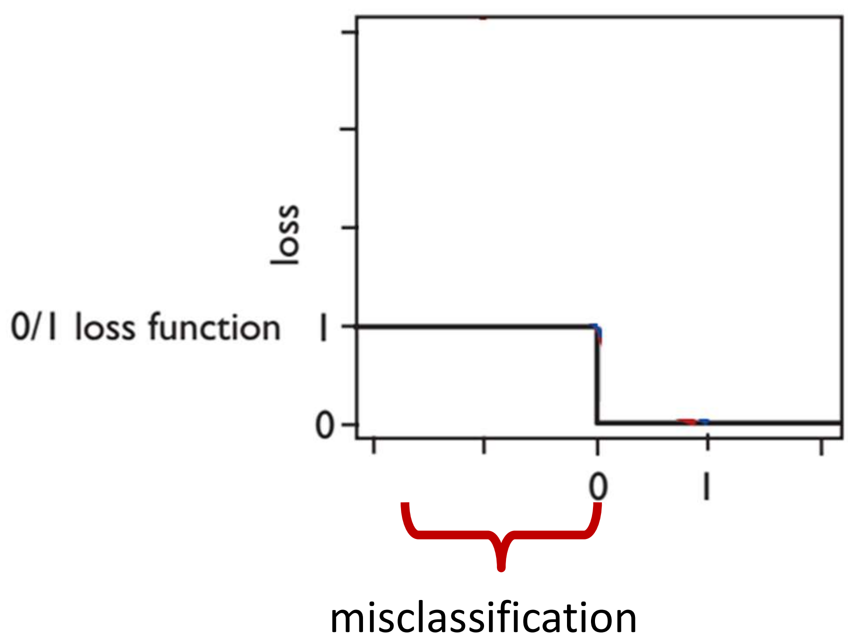


Misclassification Error: $-\,sign(y^{(i)}[w.x^{(i)}+b])$

[Mohri]

Hinge Loss: SVM Classifier

Support Vector Machine (SVM)

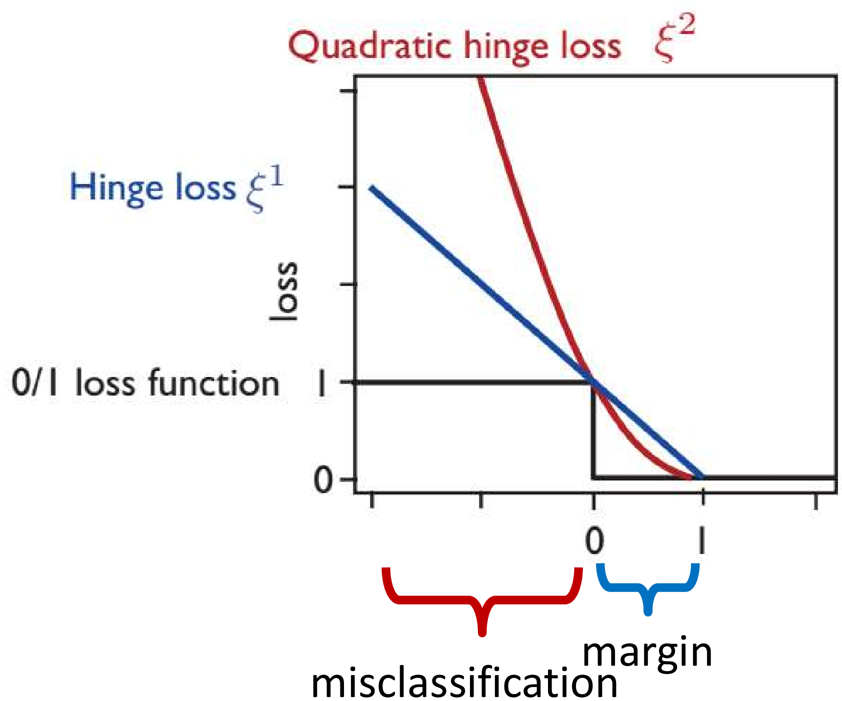


Misclassification Error: $-\,sign(y^{(i)}[w.x^{(i)}+b])$

[Mohri]

Hinge Loss: SVM Classifier

Support Vector Machine (SVM)



Misclassification Error: $-\,sign(y^{(i)}[w.x^{(i)}+b])$

[Mohri]

Cross Entropy Loss

- Cross-entropy:

$$H(q,p) = -\sum_{x} q(x) \log p(x)$$

Cross Entropy Loss

- Cross-entropy:

$$H(q,p) = -\sum_{x} q(x)\log p(x)$$

- Multi-class cross-entropy Loss:

$$L(\hat{y},y) = -\sum_{j} y_j \log \widehat{y_j}$$

Cross Entropy Loss

- Multi-class cross-entropy Loss:

$$L(\hat{y}, y) = -\sum_j y_j \log \boxed{\hat{y}_j}$$

predicted probability of each class!

Cross Entropy Loss: Softmax Classifier

- Multi-class cross-entropy Loss:

$$L(\hat{y}, y) = -\sum_{j} y_j \log \boxed{\widehat{y_j}}$$

predicted probability of each class!

- Sotmax classifier:

$$s = f(x)$$

$$\widehat{y_j} = \frac{e^{s_j}}{\sum_{k=1}^{C} e^{s_c}}$$

Cross Entropy Loss: Softmax Classifier

- Multi-class cross-entropy Loss:

$$L(\hat{y}, y) = -\sum_{j} \boxed{y_j} \log \boxed{\hat{y}_j}$$

predicted probability of each class!

The true distribution (one-hot vector: q = [0,0,0, …,1, …,0])

- Sotmax classifier:

$$s = f(x)$$

$$\hat{y}_j = \frac{e^{s_j}}{\sum_{k=1}^{C} e^{s_c}}$$

Loss Functions in Supervised Learning

- Regression Losses
  - Mean Square Error (L2 Loss)

- Classification Losses
  - Hinge Loss (Multi class SVM)
  - Cross Entropy Loss