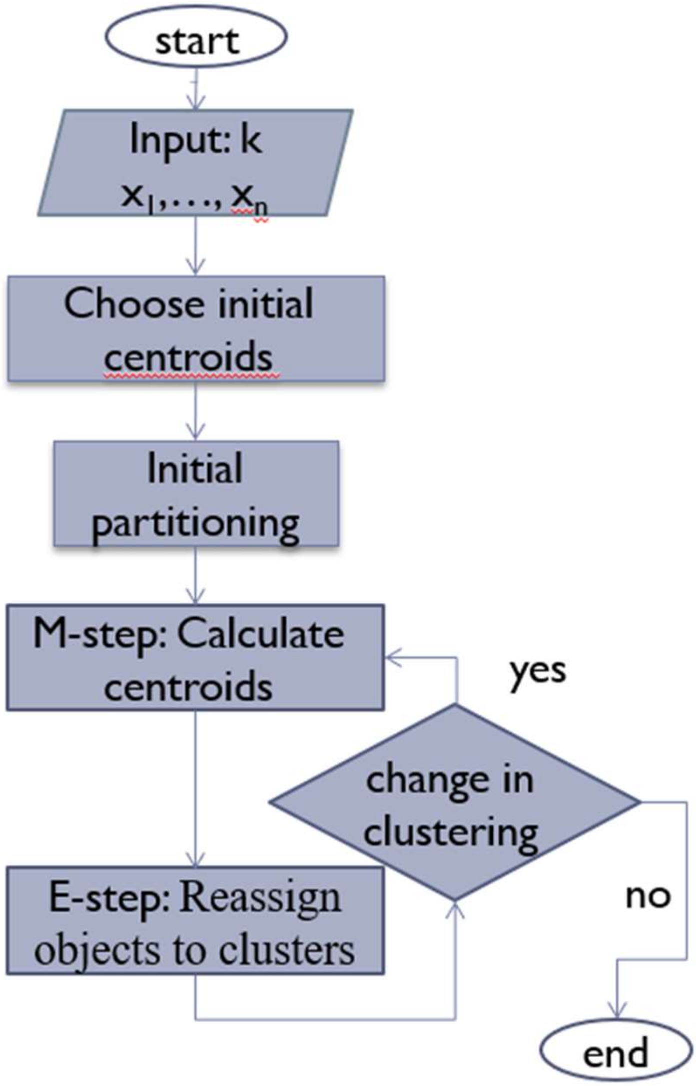
# Introduction to Machine Learning

## WSS ML Workshop

Hosein Hasani

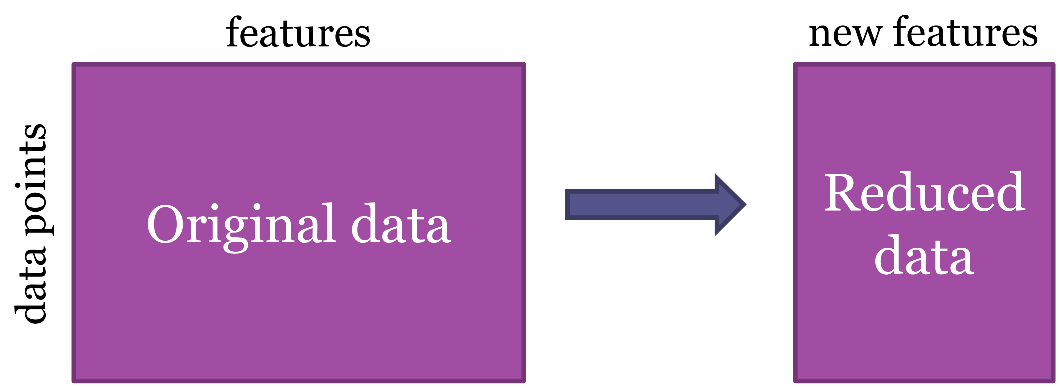
WSS 2024

# Clustering: K-means Algorithm



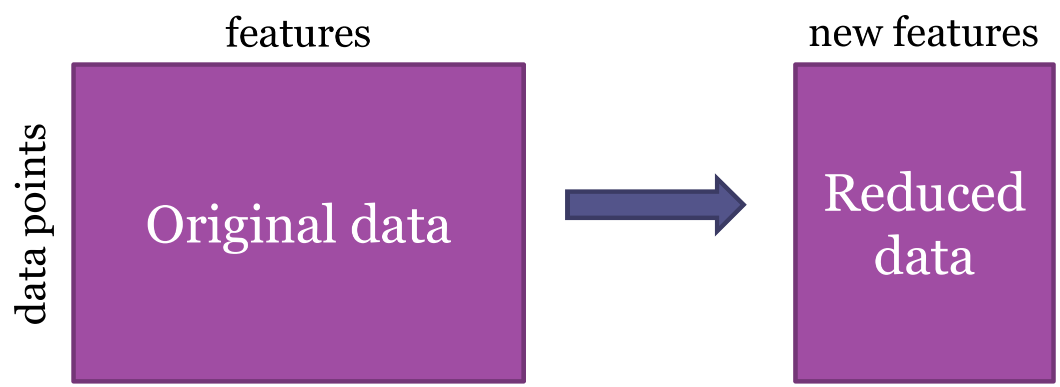
# Dimensionality Reduction

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



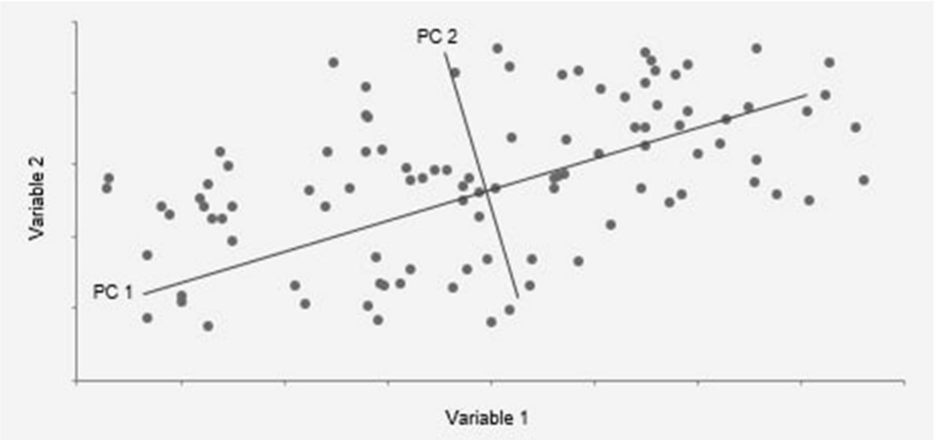
# Dimensionality Reduction

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



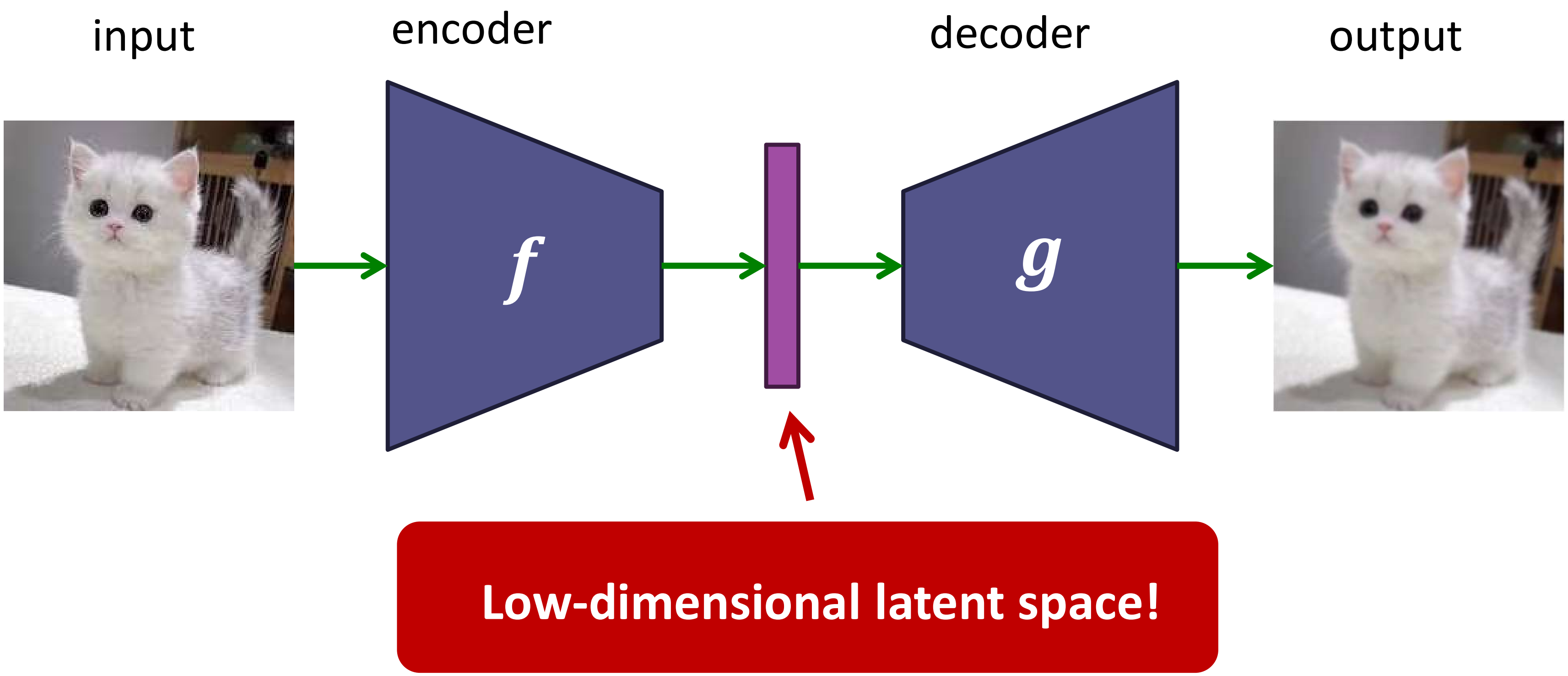
# Dimensionality Reduction

- 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

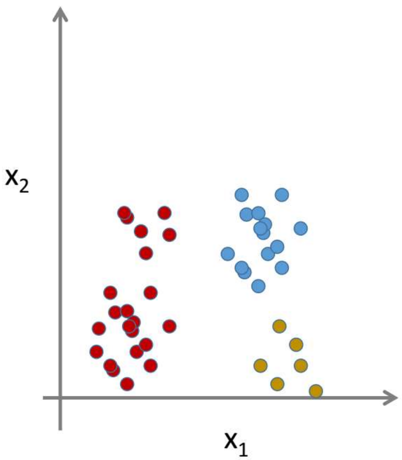
Given: Training set

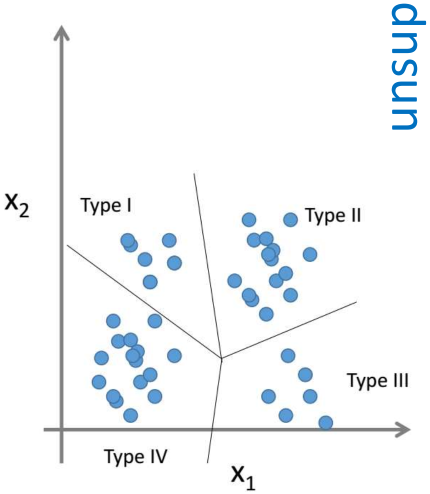
$$D = \{(x^{(i)})\}_{i=1}^{N}$$

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

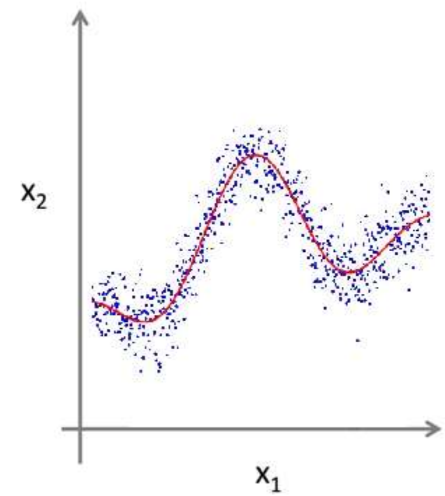
- Main Approaches:
  - Density Estimation
  - Generative Modelling
  - Clustering
  - Dimensionality Reduction

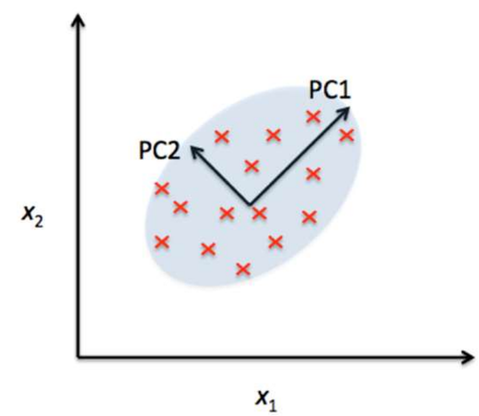
# Supervised Learning vs. Unsupervised Learning




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



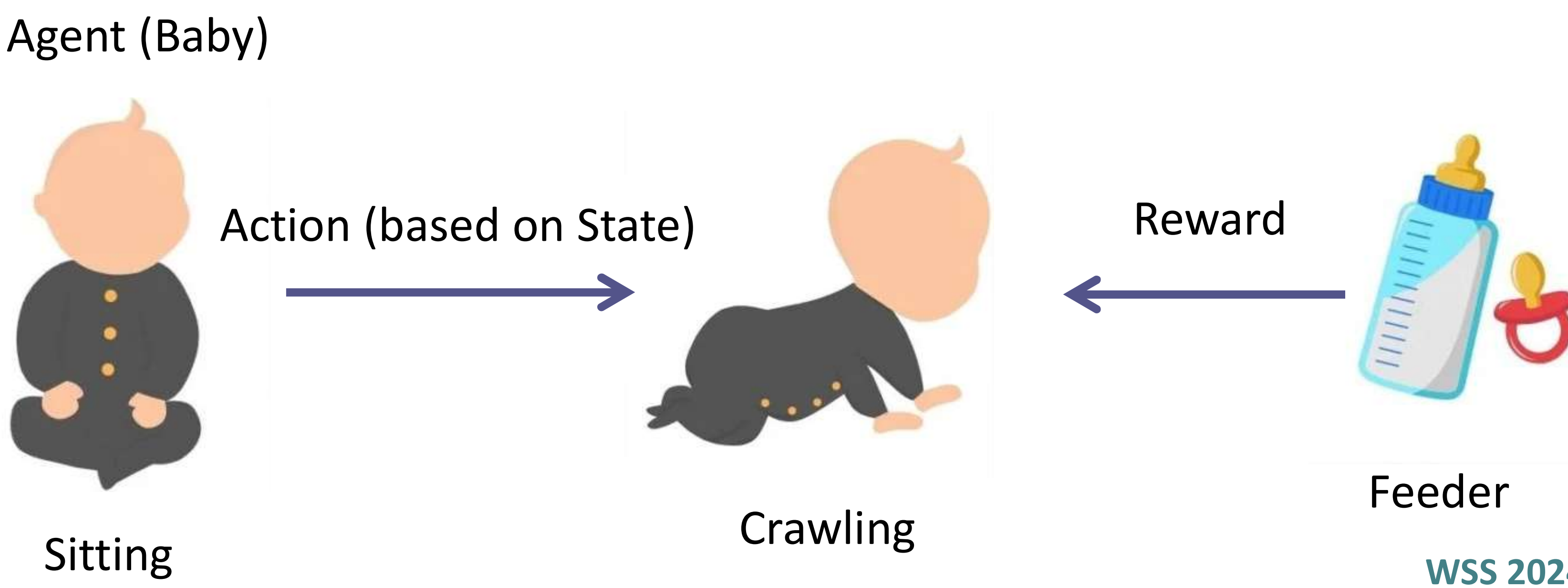



# Outline

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

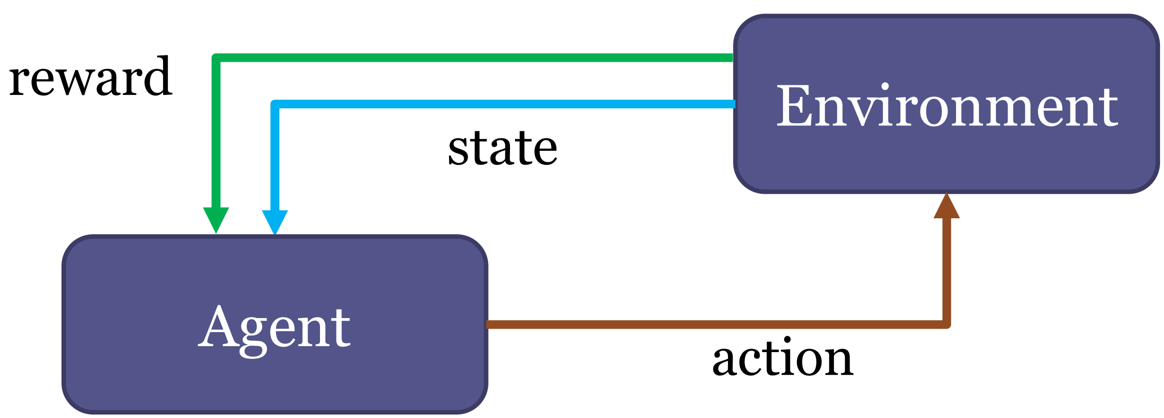
# Reinforcement Learning

- 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.



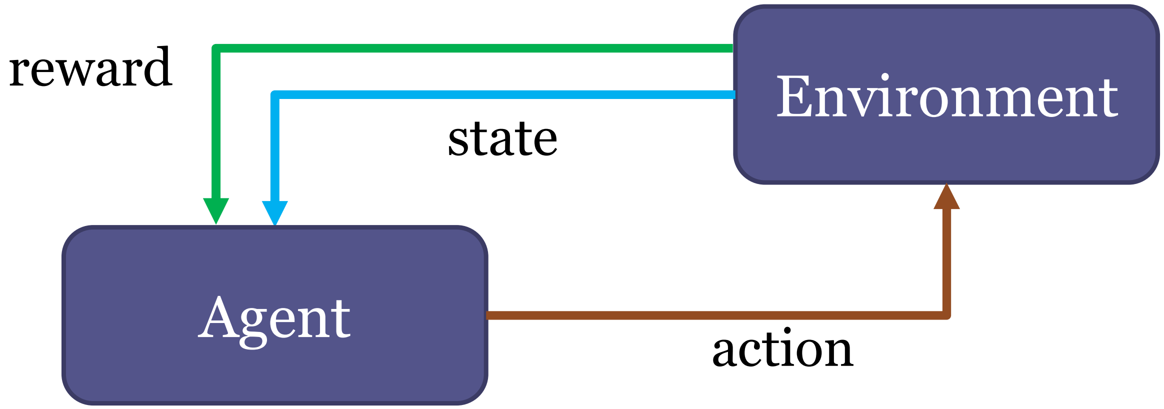
# Reinforcement Learning

- 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

$$\pi_\theta: S \rightarrow A$$

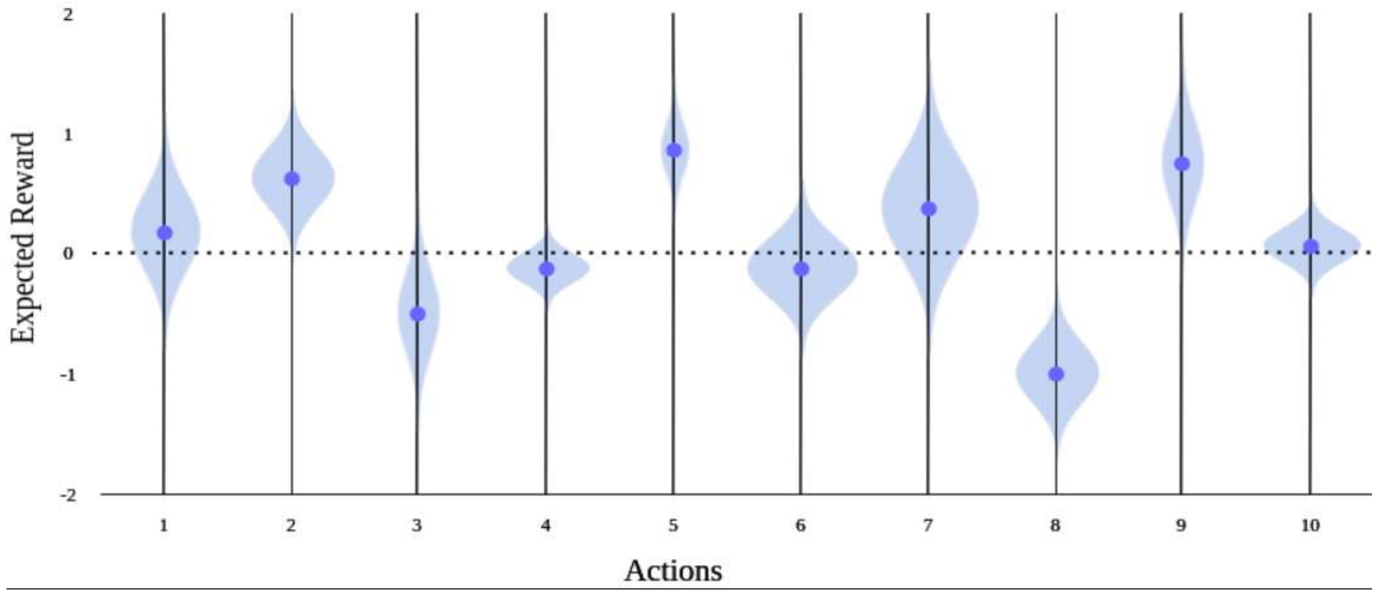
- Goal: Learning an optimal policy in order to maximize its long-term reward

# 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!

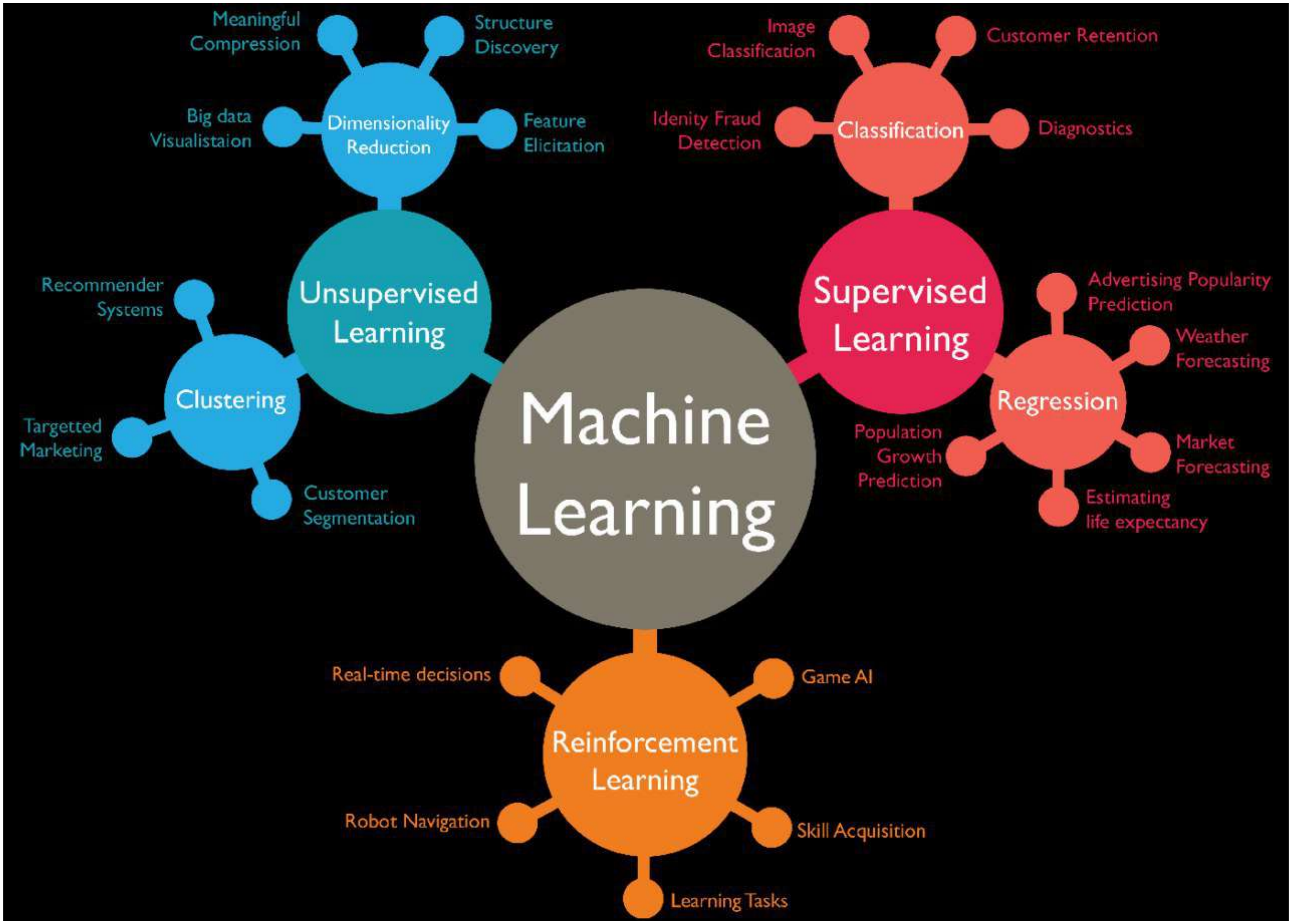


# Multi-Armed Bandit

Applications:

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

# Primary ML Problems (Review)

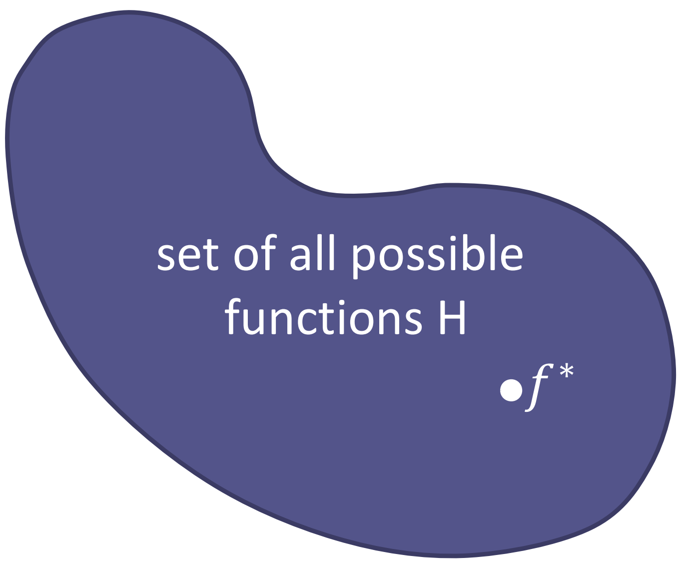


# Hypothesis Class (Recap)

- 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

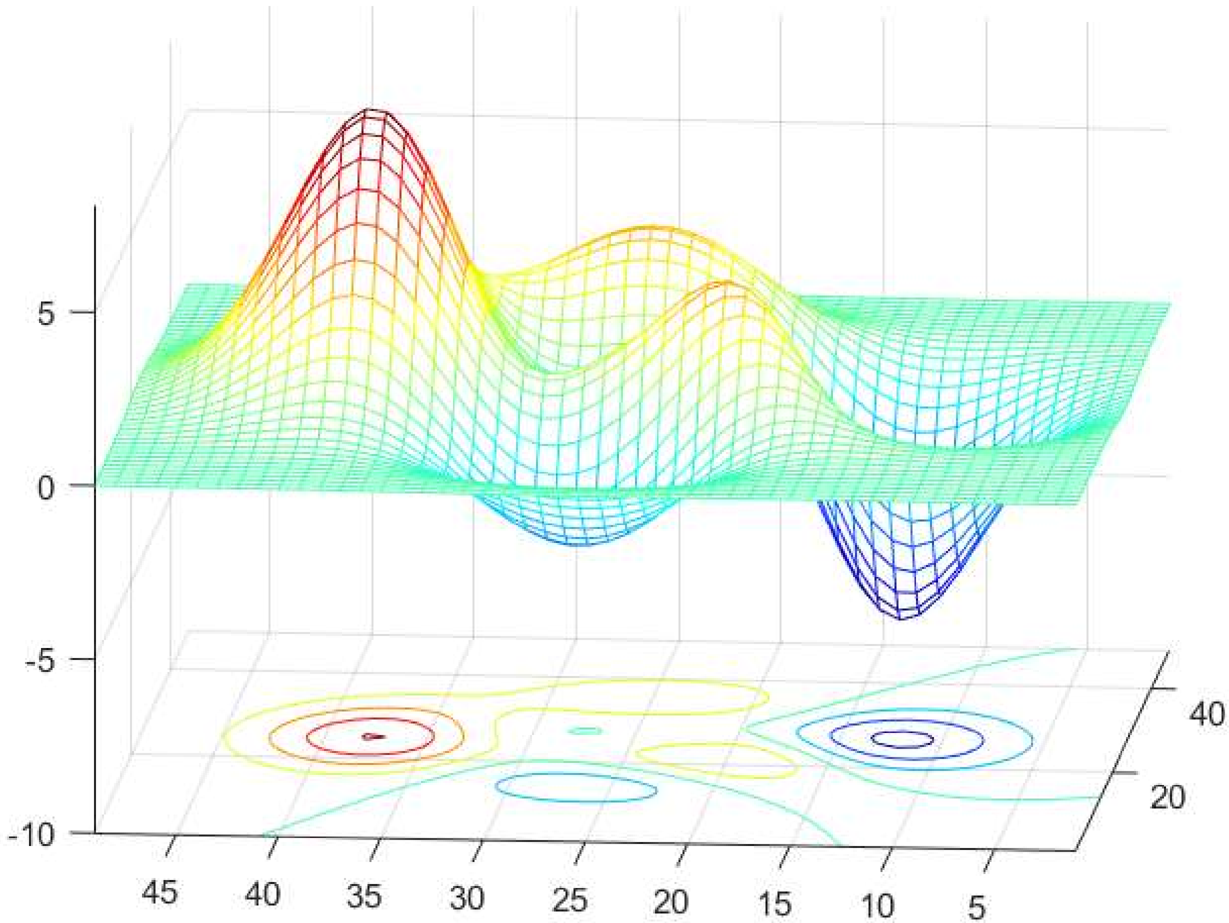
Typical steps of solving (supervised) learning problems:

- 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.

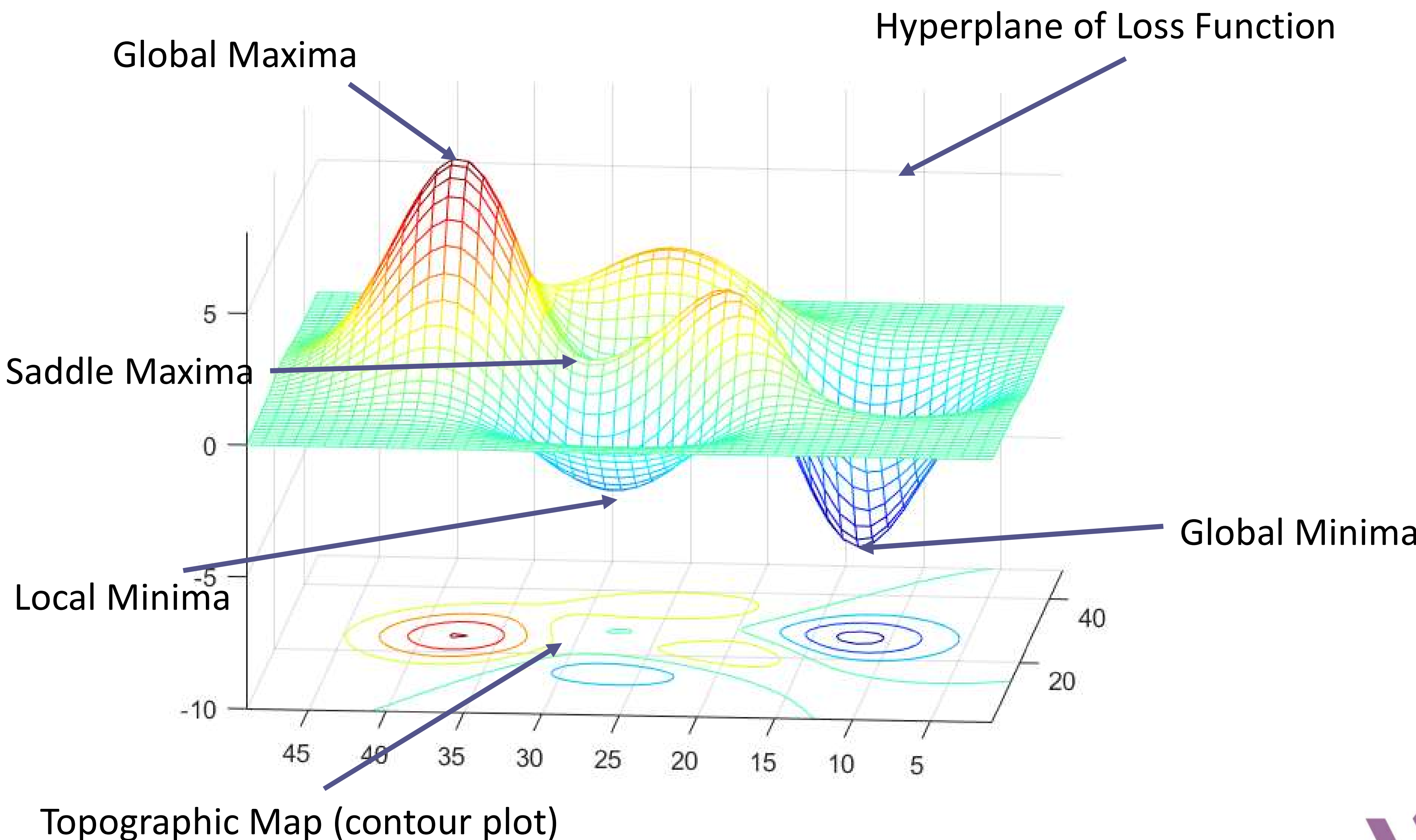
# Loss Function

- 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

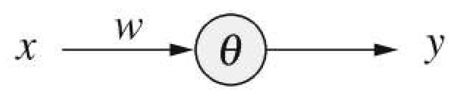


# Loss Function: Loss Landscape



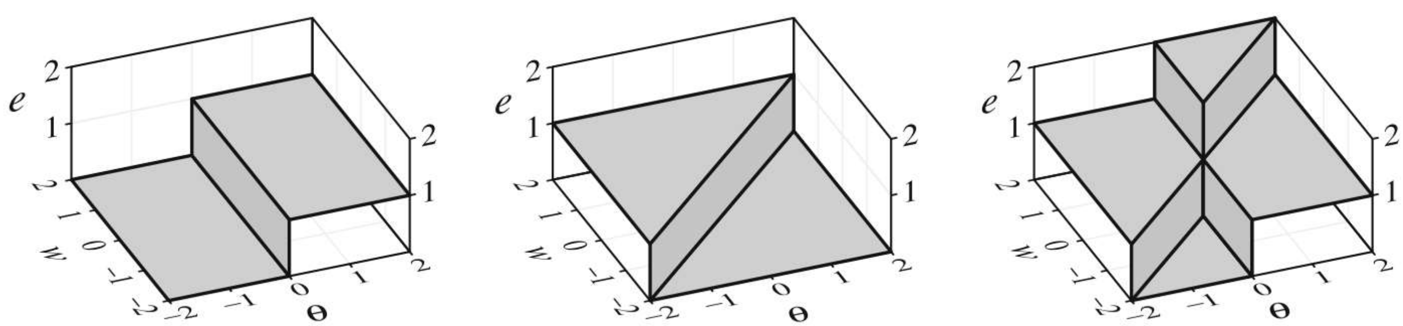
# Loss Functions: Negation Problem

- 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 $\theta$ and $w$:




error for $x = 0$    error for $x = 1$    sum of errors

$$L = \sum_i L_i$$

# 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

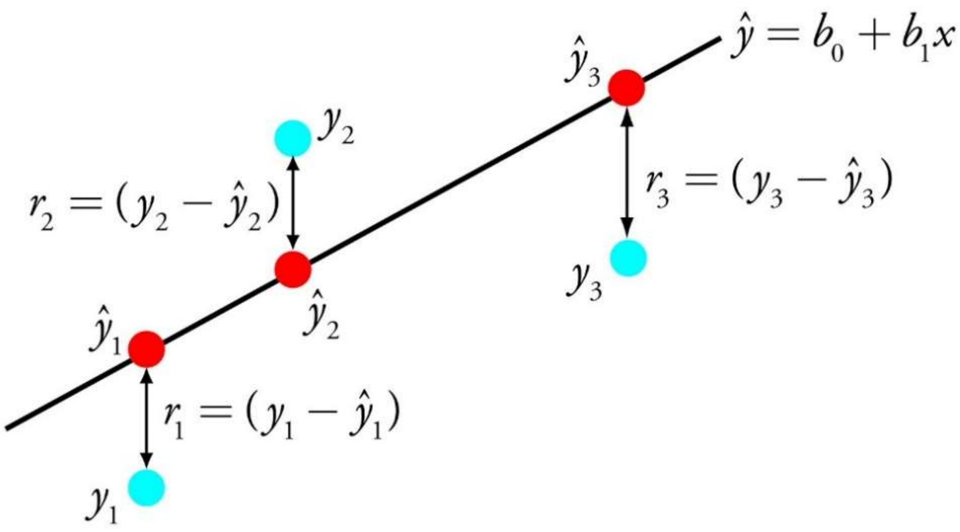
**Mean Squared Error** (MSE) loss function is widely used in regression problems.

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

# 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

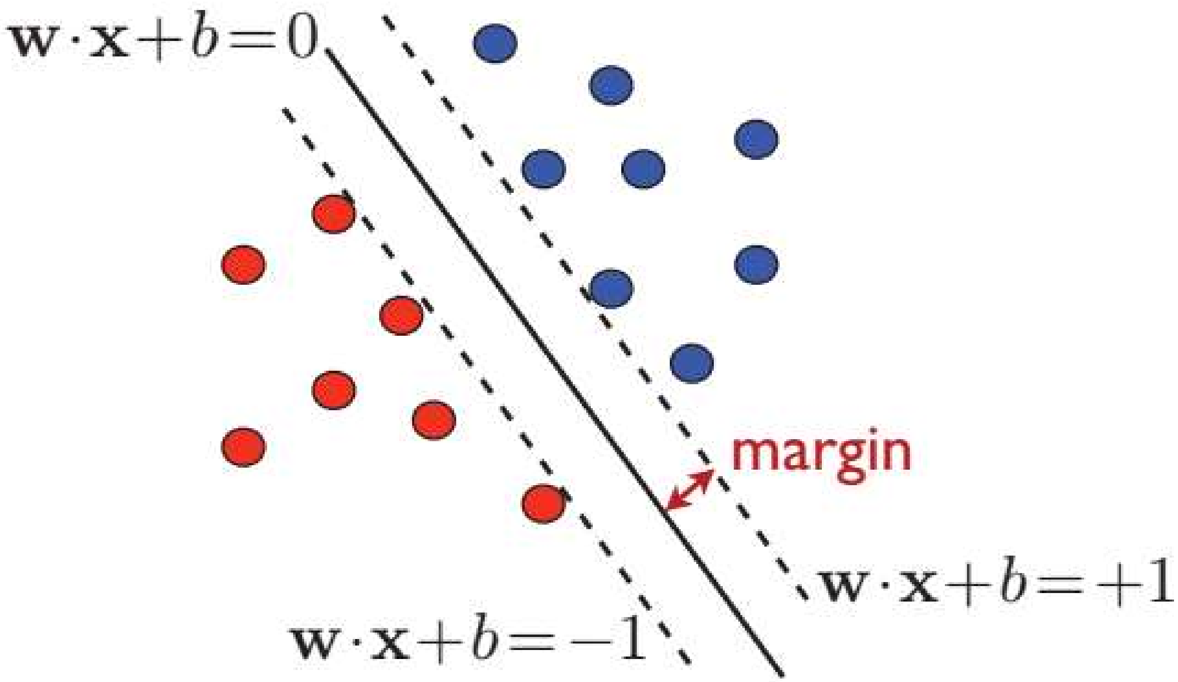
# Loss Functions in Supervised Learning

- Regression Losses
  - Mean Square Error (L2 Loss)

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

# Hinge Loss: SVM Classifier

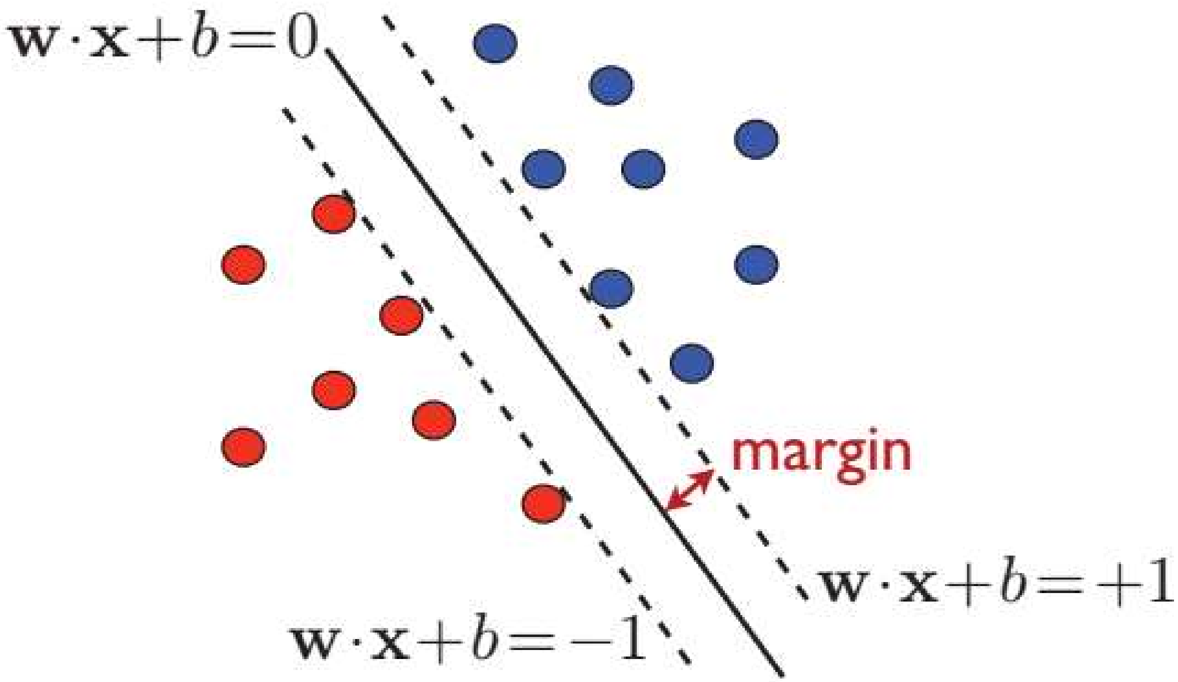
Support Vector Machine (SVM)




[Mohri]

# Hinge Loss: SVM Classifier

Support Vector Machine (SVM)

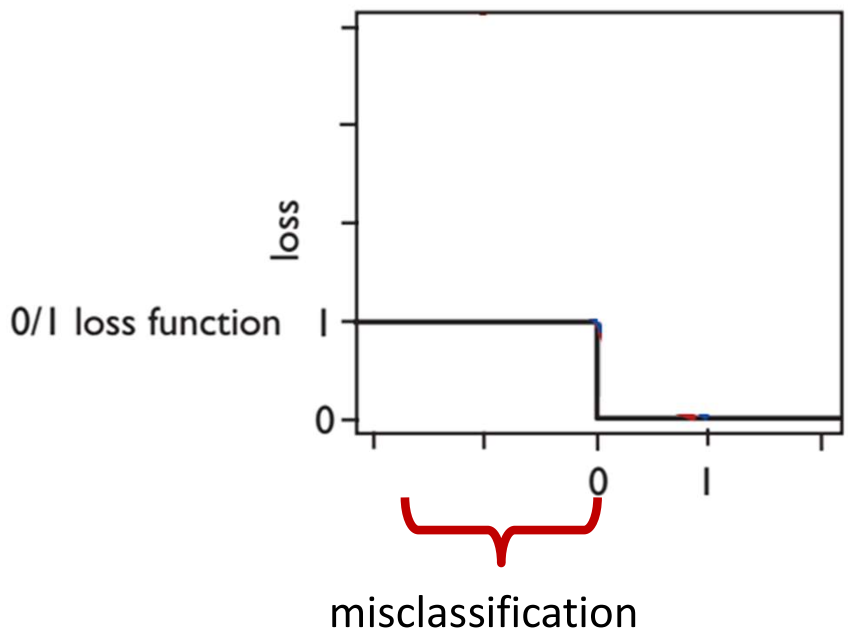



Misclassification Error: $-\,sign(y^{(i)}[w.x^{(i)}+b])$

[Mohri]

# Hinge Loss: SVM Classifier

Support Vector Machine (SVM)

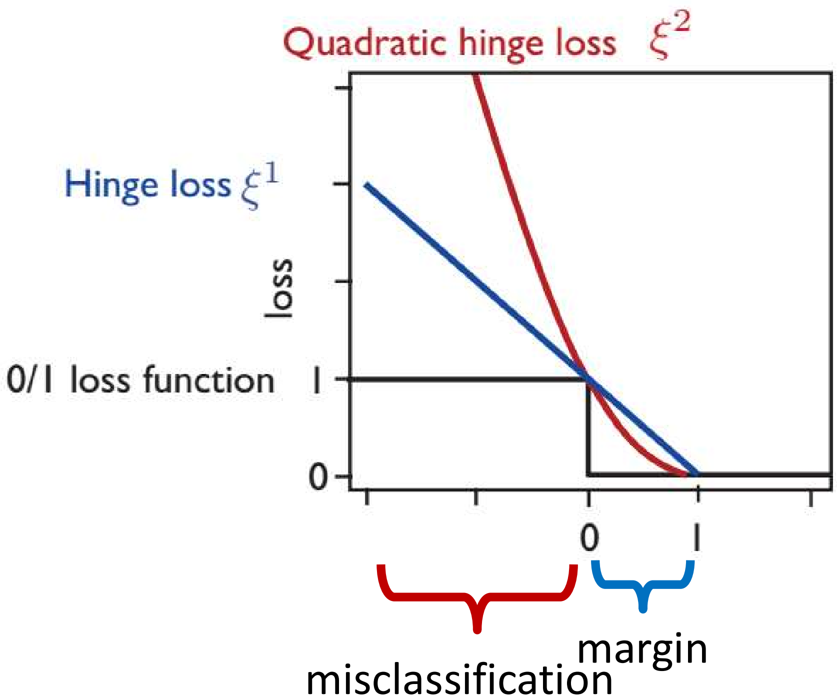



Misclassification Error: $-\,sign(y^{(i)}[w.x^{(i)}+b])$

[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:

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

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