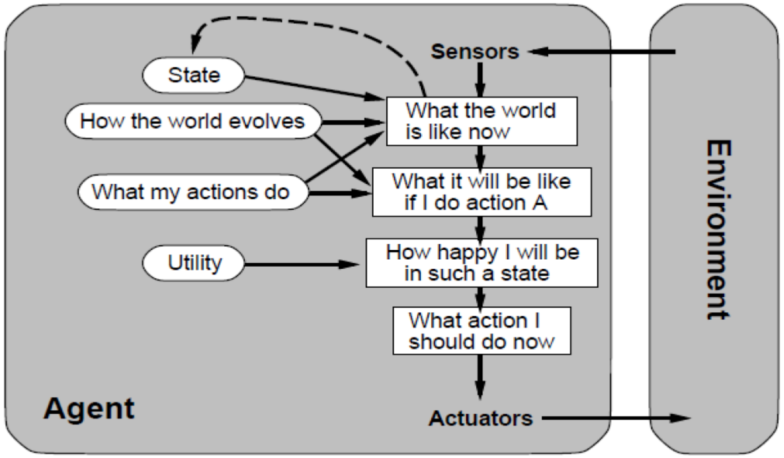
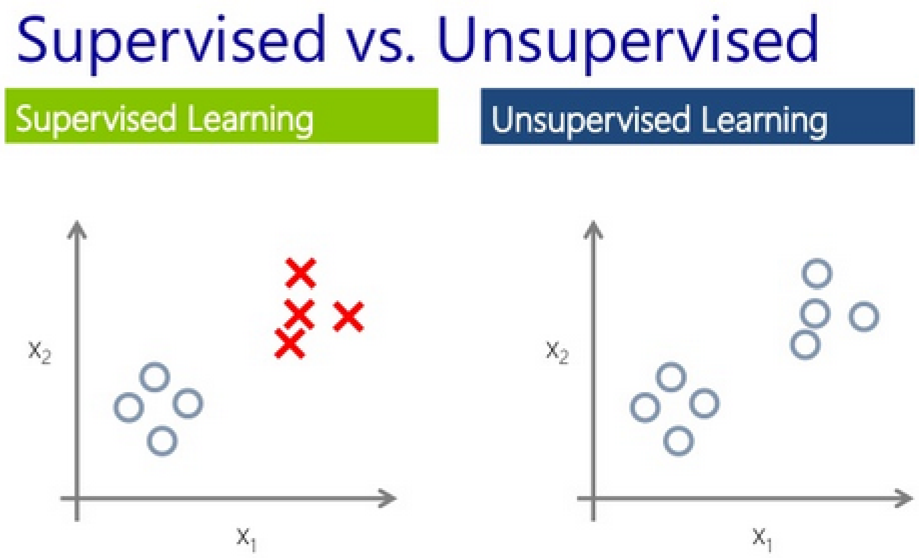
Learning from Examples

- Which component is to be improved?
  - A direct mapping from conditions on the current state to actions.
  - A means to infer relevant properties of the world from the percepts.
  - Information about the way the world evolves.
  - Utility information indicating the desirability of world states.
  - Action-value information indicating the desirability of actions.
  - Goals that describe classes of states whose achievement maximizes the agent's utility.



Learning from feedback

- What feedback is available to learn from
  - In unsupervised learning the agent learns patterns in the input even though no explicit feedback is supplied (e.g. clustering)
  - In supervised learning the agent observes some example input–output pairs and learns a function that maps from input to output



In reinforcement learning the agent learns from a series of reinforcements rewards or punishments

Supervised Learning

The task of supervised learning is this:

> Given a **training set** of N example input–output pairs
>
> $$(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N) \; ,$$
>
> where each y_j was generated by an unknown function $y = f(x)$, discover a function h that approximates the true function f.

Here x and y can be any value; they need not be numbers. The function h is a **hypothesis.**[1]

- Output is discrete: Classification
- Output is continuous: Regression

Recall Notation

$(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$ training set

Where each y_j was generated by
an unknown function $y = f(\mathbf{x})$

Discover a function h that best
approximates the true function f

hypothesis

Loss Functions

Suppose the true prediction for input $\mathbf{x}$ is $f(\mathbf{x}) = y$

but the hypothesis gives $h(\mathbf{x}) = \hat{y}$

$$L(\mathbf{x}, y, \hat{y}) = Utility(\text{result of using } y \text{ given input } \mathbf{x}) - Utility(\text{result of using } \hat{y} \text{ given input } \mathbf{x})$$

Simplified version: $L(y, \hat{y})$

Absolute value loss: $L_1(y, \hat{y}) = |y - \hat{y}|$

Squared error loss: $L_2(y, \hat{y}) = (y - \hat{y})^2$

0/1 loss: $L_{0/1}(y, \hat{y}) = 0 \text{ if } y = \hat{y}, \text{ else } 1$

Generalization loss: expected loss over all possible examples

Empirical loss: average loss over available examples

Univariate Linear Regression contd.

$$\mathbf{w} = [w_0, w_1]$$ weight vector

$$h_{\mathbf{w}}(x) = w_1 x + w_0$$

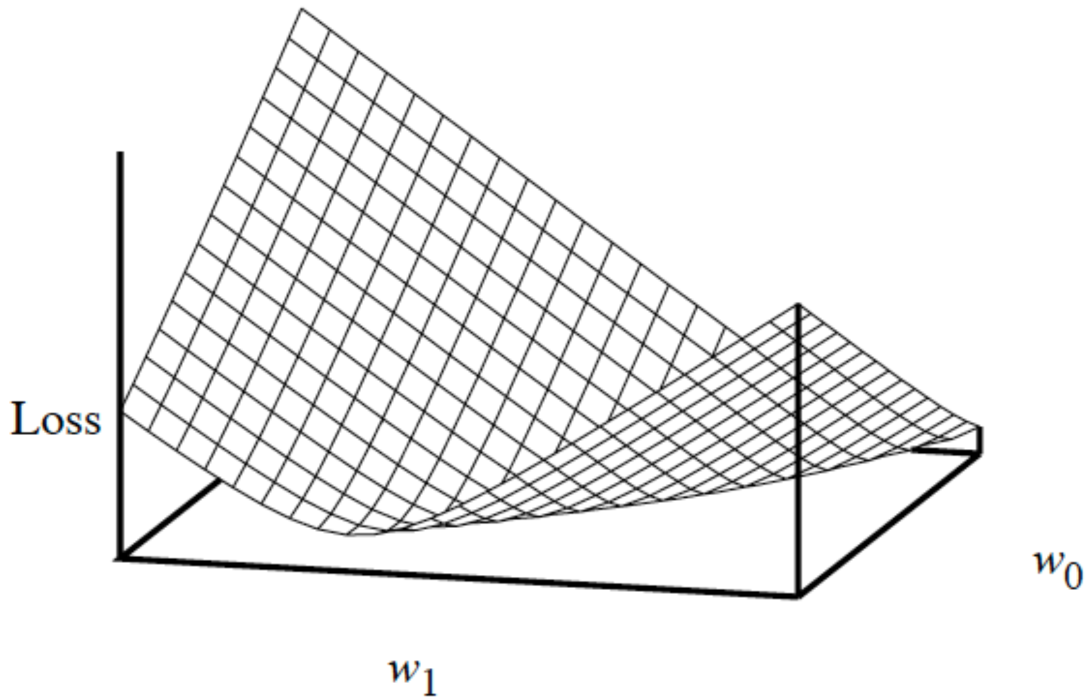
Find weight vector that minimizes empirical loss, e.g., L_2:

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j)) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$$

i.e., find $\mathbf{w}^*$ such that

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} Loss(h_{\mathbf{w}})$$

Weight Space



Finding w*

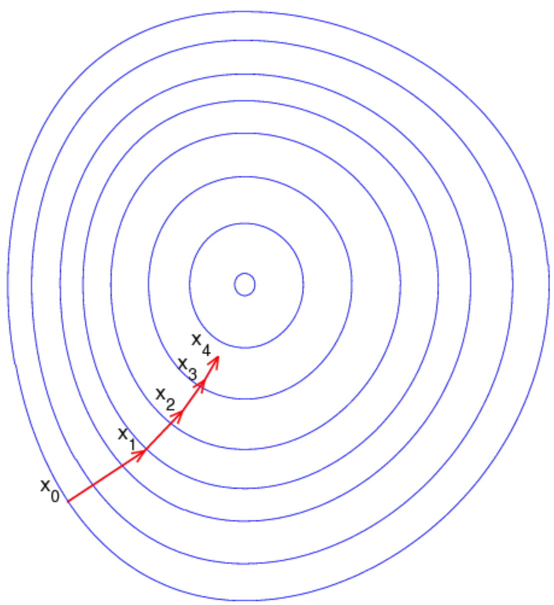
Find weights such that:

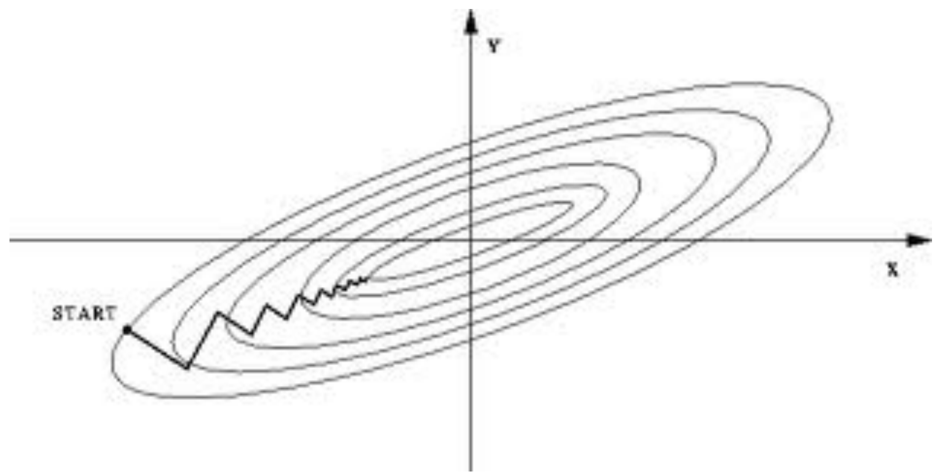
$$\frac{\partial}{\partial w_0}\sum_{j=1}^{N}(y_j - (w_1 x_j + w_0))^2 = 0 \text{ and } \frac{\partial}{\partial w_1}\sum_{j=1}^{N}(y_j - (w_1 x_j + w_0))^2 = 0$$

Gradient Descent

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

step size or
learning rate





Gradient Descent contd.

For one training example (x, y):

$$w_0 \leftarrow w_0 + \alpha(y - h_{\mathbf{w}}(x)) \quad \text{and} \quad w_1 \leftarrow w_1 + \alpha(y - h_{\mathbf{w}}(x))x$$

For *N* training examples:

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \quad \text{and} \quad w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j))x_j$$

batch gradient descent

stochastic gradient descent: take a step for one training example at a time

Perceptron Learning Rule

For a single sample $(\mathbf{x}, y)$:

$$w_i \leftarrow w_i + \alpha\big(y - h_{\mathbf{w}}(\mathbf{x})\big)x_i$$

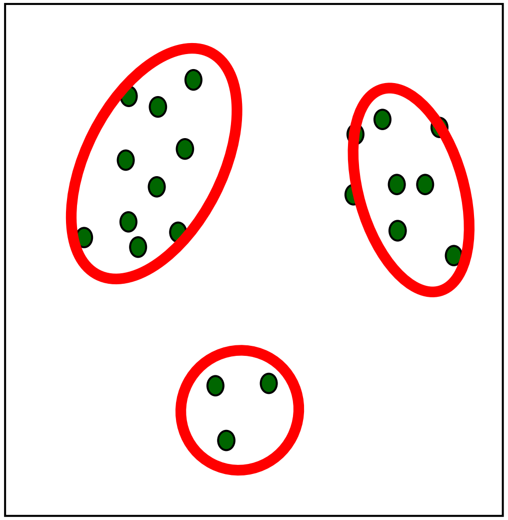
- If the output is correct, i.e., $y = h_{\mathbf{w}}(\mathbf{x})$, then the weights don't change
- If $y = 1$ but $h_{\mathbf{w}}(\mathbf{x}) = 0$, then w_i is *increased* when x_i is positive and *decreased* when x_i is negative.
- If $y = 0$ but $h_{\mathbf{w}}(\mathbf{x}) = 1$, then w_i is *decreased* when x_i is positive and *increased* when x_i is negative.

<u>Perceptron Convergence Theorem</u>: For any data set that's linearly separable and any training procedure that continues to present each training example, the learning rule is guaranteed to find a solution in a finite number of steps.

Unsupervised learning

- Supervised learning
- Predict target value ("y") given features ("x")

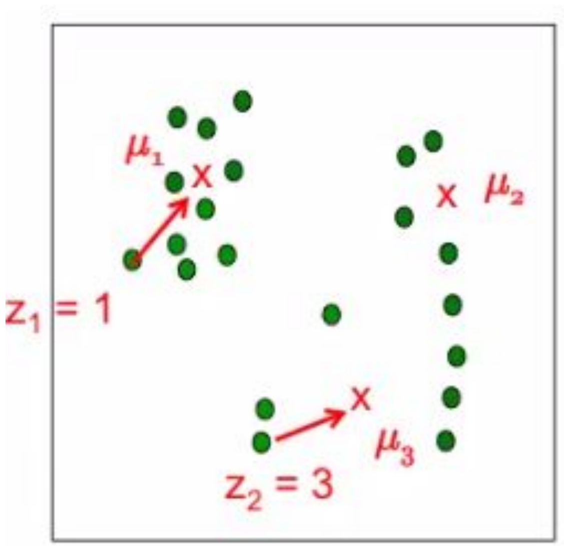
- Unsupervised learning
- Understand patterns of data (just "x")
- Useful for many reasons
- Data mining ("explain")
- Representation (feature generation or selection)



- One example: *clustering*
- *Describe data by discrete "groups" with some characteristics*

K-Means Clustering

- A simple clustering algorithm
- Iterate between
  - Updating the assignment of data to clusters
  - Updating the cluster's summarization
- Suppose we have K clusters, c=1..K
- Represent clusters by locations $^{1}_{c}$
- Example i has features x_i
- Represent assignment of i^{th} example z_i 2 1..K



K-Means Clustering

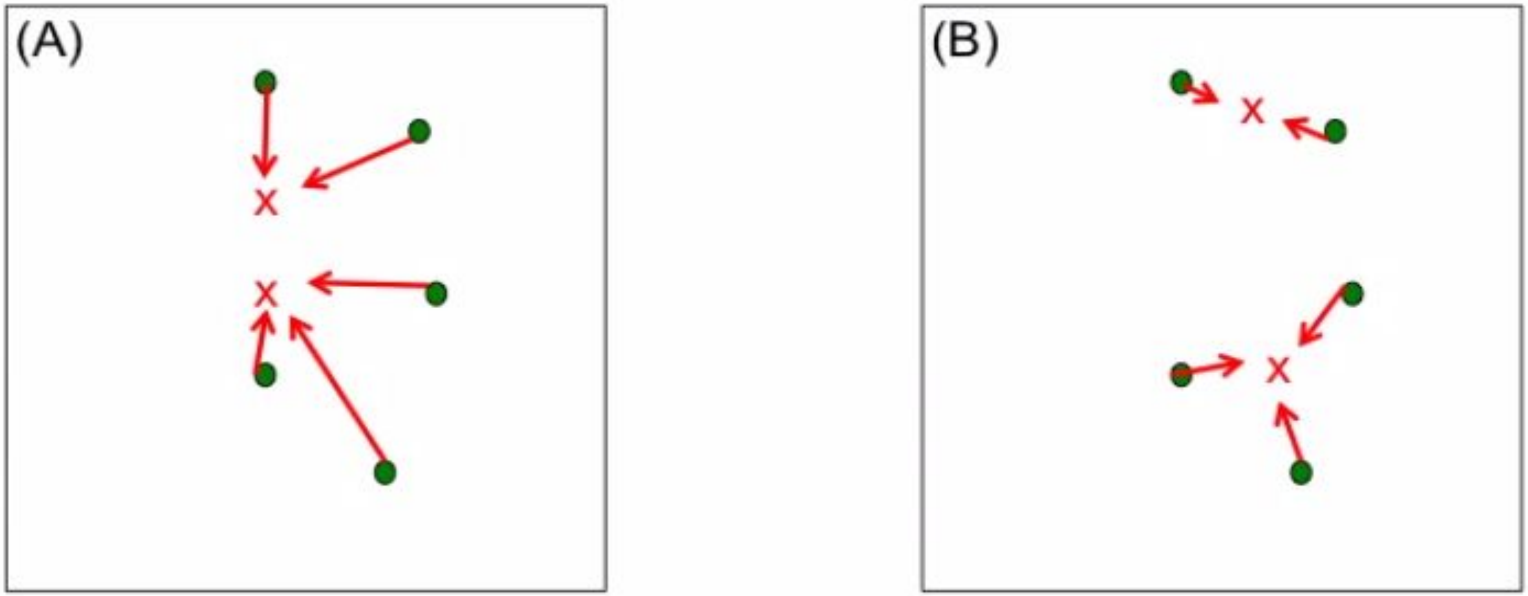
Iterate until convergence:

- (A) For each datum, find the closest cluster

$$z_i = \arg\min_c \|x_i - \mu_c\|^2 \qquad \forall i$$

- (B) Set each cluster to the mean of all assigned data:

$$\forall c, \qquad \mu_c = \frac{1}{m_c} \sum_{i \in S_c} x_i \qquad\qquad S_c = \{i : z_i = c\},\ m_c = |S_c|$$



K-Means Clustering

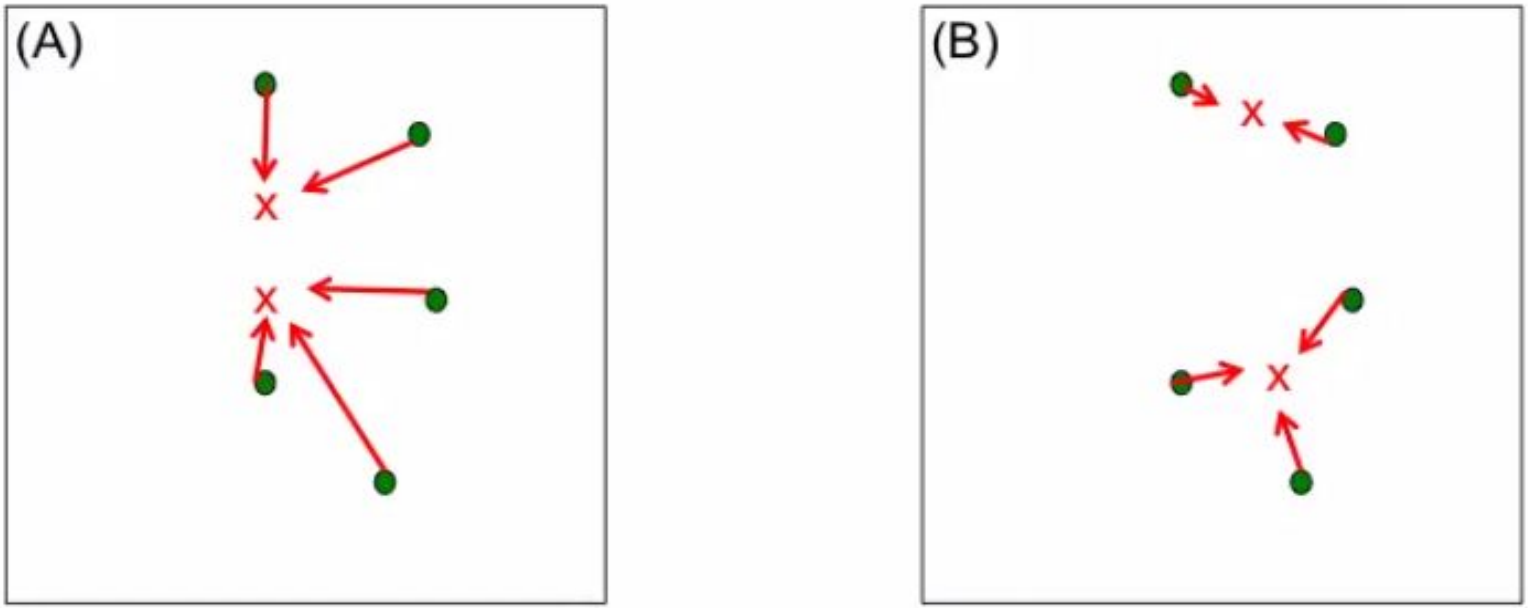
- Optimizing the cost function:

$$C(\underline{z}, \underline{\mu}) = \sum_i \|x_i - \mu_{z_i}\|^2$$

- Coordinate descent:

Over the cluster assignments:
Only one term in sum depends on z_i
Minimized by selecting closest μ_c

Over the cluster centers:
Cluster c only depends on x_i with z_i=c
Minimized by selecting the mean



K-Means clustering

- As with any descent method, beware of local minima
- Algorithm behavior depends significantly on initalization

$$C(\underline{z}, \underline{\mu}) = \sum_i \|x_i - \mu_{z_i}\|^2$$