Day-96, Falgun 22, 2081 (Mar 6, 2025)

1. Using per-item features
2. Collaborative filtering Algorithm
3. Supervised vs Unsupervised ML Algorithm
4. Recommenders $f(w, b, x)$ using even x' as parameters-

Source: Coursera Course - Unsupervised Learning, Recommenders, Reinforcement Learning. Week (2)

What if we have features of the movies?

Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

$n_u = 4$
$n_m = 5$
$n = 2$

$x^{(1)} \Rightarrow \begin{bmatrix} 0.9 \\ 0 \end{bmatrix}$

$x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$

4.95 ?

For user 1: Predict rating for movie i as: $w \cdot x^{(i)} + b$

$w^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$ $b^{(1)} = 0$ $x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$

$w^{(1)} \cdot x^{(3)} + b^{(1)} \Rightarrow 4.95$ → just linear Regression.

Now for each user we can predict

for user j. Predict j's rating for movie (i)

$i : r(i,j) = 1$

Cost function:

$$\frac{1}{2m^{(j)}} \sum_{i:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)} \right)^2$$

only user j has rated the movie i so the i is the 1.

use only

$J(w^{(j)}, b^{(j)})$

$\Rightarrow$

$$\min_{w^{(j)}, b^{(j)}} \frac{1}{2m^{(j)}} \sum_{i:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2m^{(j)}} \sum_{k=1}^{n} \left(w_k^{(j)} \right)^2$$

$\rightarrow$ Regularization terms

What if we have features of the movies?

$n_u = 4$
$n_m = 5$
$n = 2$

Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

$$x^{(1)} = \begin{bmatrix} 0.9 \\ 0 \end{bmatrix} \qquad x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$$

For user 1: Predict rating for movie i as: $w^{(1)} \cdot x^{(i)} + b^{(1)}$ ← just linear regression

$w^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$ $b^{(1)} = 0$ $x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$

$w^{(1)} \cdot x^{(3)} + b^{(1)} = 4.95$

For user j: Predict user j's rating for movie i as $w^{(j)} \cdot x^{(i)} + b^{(j)}$

DeepLearning.AI

Cost function

Notation:

- $r(i,j) = 1$ if user j has rated movie i (0 otherwise)
- $y^{(i,j)}$ = rating given by user j on movie i (if defined)
- $w^{(j)}, b^{(j)}$ = parameters for user j
- $x^{(i)}$ = feature vector for movie i

For user j and movie i, predict rating: $w^{(j)} \cdot x^{(i)} + b^{(j)}$

$m^{(j)}$ = no. of movies rated by user j

To learn $w^{(j)}, b^{(j)}$

Cost function

Notation:

- $r(i,j) = 1$ if user j has rated movie i (0 otherwise)
- $y^{(i,j)}$ = rating given by user j on movie i (if defined)
- $w^{(j)}, b^{(j)}$ = parameters for user j
- $x^{(i)}$ = feature vector for movie i

For user j and movie i, predict rating: $w^{(j)} \cdot x^{(i)} + b^{(j)}$

$m^{(j)}$ = no. of movies rated by user j

To learn $w^{(j)}$, $b^{(j)}$

number of features

$$\min_{w^{(j)}, b^{(j)}} J\left(w^{(j)}, b^{(j)}\right) = \frac{1}{2m^{(j)}} \sum_{i:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2m^{(j)}} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2$$

DeepLearning.AI Stanford ONLINE Andrew Ng

Cost function:

To learn parameters $w^{(j)}, b^{(j)}$ for user j:

$$J(w^{(j)}, b^{(j)}) \Rightarrow \frac{1}{2} \sum_{i:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{k=1} \left(w_k^{(j)}\right)^2$$

To learn parameters $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, \ldots w^{(n_u)}, b^{(n_u)}$ for all users:

$$J\begin{pmatrix} w^{(1)} & \cdots & w^{(n_u)} \\ b^{(1)} & \cdots & b^{(n_u)} \end{pmatrix} = \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2$$

$f(x)$

Under the hood we can use Linear Regression.

Cost function

To learn parameters $w^{(j)}, b^{(j)}$ for user j :

$$J\left(w^{(j)}, b^{(j)}\right) = \frac{1}{2} \sum_{i:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2$$

To learn parameters $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, \cdots w^{(n_u)}, b^{(n_u)}$ for all users :

$$J\begin{pmatrix} w^{(1)}, \ldots, w^{(n_u)} \\ b^{(1)}, \ldots, b^{(n_u)} \end{pmatrix} = \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \underbrace{\left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2}_{f(x)} + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2$$

DeepLearning.AI Stanford | ONLINE Andrew Ng

Collaborative filtering algorithm.

→ Linear Regression needs x_1 and x_2. But what if you don't have these x_1 and x_2 → then we need to extract or find from the dataset itself.

ignore $b^{(i)}$.

$w^{(1)} \cdot x^{(1)} \approx 5$

$w^{(2)} \cdot x^{(1)} \approx 5$

$w^{(3)} \cdot x^{(1)} \approx 0$

$w^{(4)} \cdot x^{(1)} \approx 0$

What value must $x^{(1)}$ hold?

$x^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	?	?
Romance forever	5	?	?	0	?	?
Cute puppies of love	?	4	0	?	?	?
Nonstop car chases	0	0	5	4	?	?
Swords vs. karate	0	0	5	?	?	?

Assume $w^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$, $w^{(2)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$, $w^{(3)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$, $w^{(4)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$

$b^{(1)} = 0$, $b^{(2)} = 0$, $b^{(3)} = 0$

Here's the data that we had before.

$b^{(4)} = 0$ ← using $w^{(j)} \cdot x^{(i)} + b^{(i)}$

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	?	?
Romance forever	5	?	?	0	?	?
Cute puppies of love	?	4	0	?	?	?
Nonstop car chases	0	0	5	4	?	?
Swords vs. karate	0	0	5	?	?	?

(2)

(3)

$$w^{(1)}=\begin{bmatrix}5\\0\end{bmatrix} \quad ,w^{(2)}=\begin{bmatrix}5\\0\end{bmatrix} \quad ,w^{(3)}=\begin{bmatrix}0\\5\end{bmatrix} \quad ,w^{(4)}=\begin{bmatrix}0\\5\end{bmatrix}$$

$$b^{(1)}=0 \quad , \; b^{(2)}=0 \quad , \; b^{(3)}=0 \quad , \; b^{(4)}=0$$

using $w^{(j)} \cdot x^{(i)} + b^{(j)}$

$$w^{(1)} \cdot x^{(1)} \approx 5$$
$$w^{(2)} \cdot x^{(1)} \approx 5$$
$$w^{(3)} \cdot x^{(1)} \approx 0$$
$$w^{(4)} \cdot x^{(1)} \approx 0$$

$$\rightarrow \quad x^{(1)}=\begin{bmatrix}1\\0\end{bmatrix}$$

Goal: To learn meaningful features for movies based on user ratings, can accurately predict missing ratings —

① Find the best $x^{(i)}$ (movie features) and $w^{(j)}$ (user preferences) that minimize prediction.

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	?	?
Romance forever	5	?	?	0	?	?
Cute puppies of love	?	4	0	?	?	?
Nonstop car chases	0	0	5	4	?	?
Swords vs. karate	0	0	5	?	?	?

$w^{(1)} = \begin{bmatrix}5\\0\end{bmatrix}$, $w^{(2)} = \begin{bmatrix}5\\0\end{bmatrix}$, $w^{(3)} = \begin{bmatrix}0\\5\end{bmatrix}$, $w^{(4)} = \begin{bmatrix}0\\5\end{bmatrix}$

$b^{(1)} = 0$, $b^{(2)} = 0$, $b^{(3)} = 0$, $b^{(4)} = 0$ ←

using $w^{(j)} \cdot x^{(i)} + b^{(j)}$

$w^{(1)} \cdot x^{(1)} \approx 5$
$w^{(2)} \cdot x^{(1)} \approx 5$
$w^{(3)} \cdot x^{(1)} \approx 0$
$w^{(4)} \cdot x^{(1)} \approx 0$

→ $x^{(1)} = \begin{bmatrix}1\\0\end{bmatrix}$

② Adjust feature values ($x^{(i)}$) to best fit the given ratings.

$x^{(1)}$ [1 0]

→ (High Romance, Not Action)

Cost function

Given $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, \dots, w^{(n_u)}, b^{(n_u)}$

to learn $x^{(i)}$.

$$J(x^{(i)}) \Rightarrow \frac{1}{2} \sum_{j:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} \left(x_k^{(i)} \right)^2$$

→ To learn $x^{(1)}, x^{(2)} \dots x^{(n_m)}$:

$$J(x^{(1)}, x^{(2)}, \dots x^{(n_m)}) \Rightarrow \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i:r} \sum_{k=1}^{m} \left(x_k^{(i)} \right)^2$$

In this Algorithm we can actually learn features from the movies and ratings.

Given w^1, b^1, w^2, b^2,

DeepLearning.AI Stanford ONLINE Andrew Ng

Cost function

Given $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, \cdots, w^{(n_u)}, b^{(n_u)}$

to learn $x^{(i)}$:

$$J\left(x^{(i)}\right) = \frac{1}{2} \sum_{j:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2$$

→ To learn $x^{(1)}, x^{(2)}, \cdots, x^{(n_m)}$:

$$J\left(x^{(1)}, x^{(2)}, \ldots, x^{(n_m)}\right) = \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2$$

that will give us our collaborative filtering algorithm.

DeepLearning.AI Stanford ONLINE Andrew Ng

Collaborative filtering

Cost function to learn $w^{(1)}, b^{(1)}, \cdots w^{(n_u)}, b^{(n_u)}$:

$$\min_{w^{(1)},b^{(1)},\cdots,w^{(n_u)},b^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2$$

Cost function to learn $x^{(1)},\cdots,x^{(n_m)}$:

$$\min_{x^{(1)},\cdots,x^{(n_m)}} \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

this term here is exactly
the same as this term here.

DeepLearning.AI Stanford ONLINE Andrew Ng

Collaborative filtering

		j=1 Alice	j=2 Bob	j=3 Carol
→ i=1	Movie 1	5	5	?
→ i=2	Movie 2	?	2	3

Cost function to learn $w^{(1)}, b^{(1)}, \cdots w^{(n_u)}, b^{(n_u)}$:

$$\min_{w^{(1)},b^{(1)},\cdots,w^{(n_u)},b^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2$$

Cost function to learn $x^{(1)},\cdots,x^{(n_m)}$:

$$\min_{x^{(1)},\cdots,x^{(n_m)}} \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

Put them together:

$$\min_{w^{(1)},\cdots,w^{(n_u)},\ b^{(1)},\cdots,b^{(n_u)},\ x^{(1)},\cdots,x^{(n_m)}} J(w,b,x)$$

$$J(w,b,x) = \frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

the regularization term from

Collaborative filtering

	$j=1$ Alice	$j=2$ Bob	$j=3$ Carol
$i=1$ Movie1	5	5	?
$i=2$ Movie2	?	2	3

Cost function to learn $w^{(1)}, b^{(1)}, \cdots w^{(n_u)}, b^{(n_u)}$:

$$\min_{w^{(1)},b^{(1)},\cdots,w^{(n_u)},b^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2$$

Cost function to learn $x^{(1)},\cdots,x^{(n_m)}$:

$$\min_{x^{(1)},\cdots,x^{(n_m)}} \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

Put them together:

$$\min_{\substack{w^{(1)},\ldots,w^{(n_u)}\\ b^{(1)},\ldots,b^{(n_u)}\\ x^{(1)},\ldots,x^{(n_m)}}} J(w,b,x) = \frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

as a function of w, b, and x?

DeepLearning.AI Stanford ONLINE Andrew Ng

Gradient Descent

collaborative filtering

Linear regression (course 1)

repeat {

$$w_i = w_i - \alpha \frac{\partial}{\partial w_i} J(w,b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

}

$$w_i^{(j)} = w_i^{(j)} - \alpha \frac{\partial}{\partial w_i^{(j)}} J(w,b,x)$$

$$b^{(j)} = b^{(j)} - \alpha \cdot \frac{\partial}{\partial b^{(j)}} J(w,b,x)$$

$$x_k^{(i)} = x_k^{(i)} - \alpha \frac{\partial}{\partial x_k^{(i)}} J(w,b,x)$$

parameters w, b (X) parameter

w and b here to denote the parameters for all

DeepLearning.AI Stanford | ONLINE Andrew Ng

What Are We Trying to Do?

We are trying to **predict movie ratings** for users and, at the same time, **learn features for movies** (e.g., how much romance or action they contain).

The Problem:

- Normally, in machine learning, we need predefined features (like romance/action scores) to train a model.
- But here, we **don't know** these features in advance!
- So, we need to **learn them from user ratings** instead.

How We Solve It:

1. **User Preferences (w^j)**
   - Each user has a preference vector (w^j) showing how much they like different movie features.
2. **Movie Features (x^i)**
   - We adjust movie features (x^i) so that when multiplied by user preferences, they **match the given ratings**.
3. **Cost Function (Error Minimization)**
   - We use **squared error** to measure the difference between predicted and actual ratings.
   - We optimize **both user preferences and movie features** using **gradient descent**.
4. **Collaborative Filtering**
   - Since multiple users rate the same movies, we can **infer good features** by looking at shared ratings.
   - This helps predict ratings for users who haven't watched a movie yet.

What Are We Trying to Do?

We want to **predict movie ratings** for users while **learning meaningful features** for movies from data.

Step 1: Predicting Movie Ratings

Given a user j and a movie i, the predicted rating is computed as:

$$\hat{y}_{i,j} = w^j \cdot x^i + b^j$$

where:

- w^j is the preference vector for user j (how much they like different genres).
- x^i is the feature vector for movie i (how much action, romance, etc.).
- b^j is the user's bias (how lenient or strict they are in rating).
- $\cdot$ represents the **dot product**.

To simplify, we **ignore** b^j (set it to 0) for now.

Step 2: Learning Movie Features (x^i)

We assume we already have user parameters w^j, and we want to determine good values for x^i by minimizing the error.

For a given movie i, the cost function is:

$$J(x^i) = \frac{1}{2} \sum (w^j \cdot x^i - y_{i,j})^2$$

$$w^j := w^j - \alpha \frac{\partial J(w^j)}{\partial w^j}$$

Step 4: Combining Everything (Collaborative Filtering Algorithm)

The **final cost function** includes **both user preferences** and **movie features**:

$$J(w,x) = \frac{1}{2} \sum_{(i,j):r_{i,j}=1} (w^j \cdot x^i - y_{i,j})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (w_k^j)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^i)^2$$

where:

- n_u is the number of users, and n_m is the number of movies.
- The first term minimizes prediction error.
- The last two terms **regularize** w and x.

We **optimize this using gradient descent** by updating **both** w **and** x simultaneously.

Key Takeaways

- We **don't need predefined features**; we **learn them** from user ratings.
- **Collaborative filtering** allows multiple users' ratings to guide feature learning.
- We optimize **both user preferences (w) and movie features (x)** using **gradient descent**.

Would you like a worked-out example to see it in action?

Allows Automatically Learning Meaningful features from data.
Better Recommendation for don't have predefined Movie Categories.

Supervised

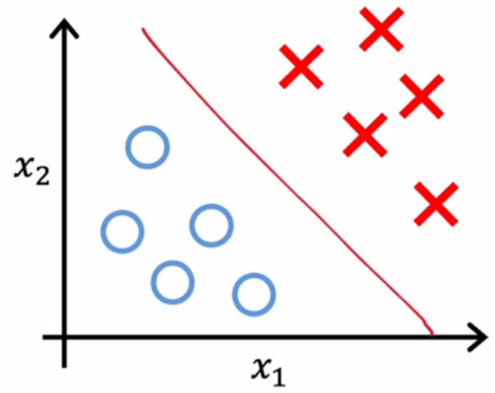
↳ Classification

↳ Regression.

Teach the Algorithm giving x and y

Supervised learning

y label given

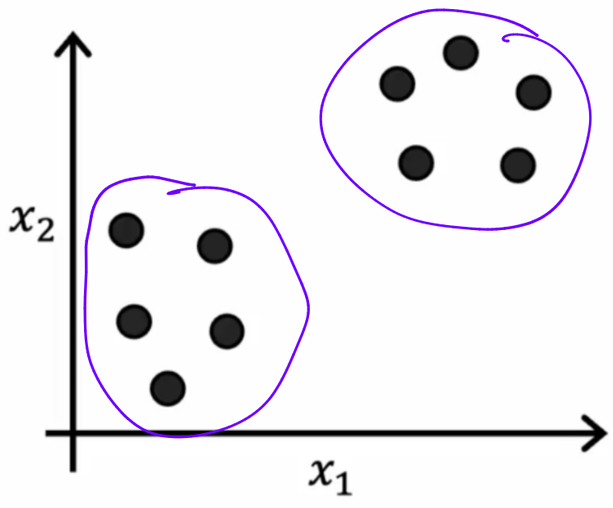


Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})\}$?

DeepLearning.AI Stanford ONLINE Andrew Ng

Unsupervised learning

Make Algorithm learn by itself giving x only



Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

Doesn't have label y

Allow Algorithm to learn y itself

DeepLearning.AI Stanford ONLINE Andrew Ng

→ Clustering

→ Applications of clustering

1. News Grouping
2. Market Segmentation (Teens)
3. DNA Analysis.
4. Astronomical Data Analysis.