Supervised Machine Learning
Regression & Classification

WEEK-1

Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.
— Arthur Samuel (1959)

Machine Learning Algorithms:
- Supervised learning (learns from being given right answers)
- Unsupervised learning
- Recommender systems
- Reinforcement learning

1) Supervised learning

$x \longrightarrow y$

Input Output

Learns from being given right answers

Examples: Spam filter, Speech recognition, Machine translation, Online advertising, Self-driving cars, Visual inspection

Input (x)	Output (y)	Application
email	spam? (0/1)	spam filtering
audio	text transcript	speech recognition
English	Spanish	machine translation
ad, user info	click? (0/1)	online advertising
image, radar info	position of other cars	self-driving car
image of phone	defect? (0/1)	visual inspection

(1) Regression: Predict a number from infinitely many possible outputs.

Regression - Housing price prediction

Regression predicts infinitely many possible outputs

(2) Classification: Predict categories. Learns from being given right answers.

Classification: Breast cancer detection

Note: There are between just two categories.

Unlike regression classification predicts small no. of possible outputs

Two or more inputs

Note: Algorithm can use both age and tumor size to make its prediction.

(3) Unsupervised Learning: Unlike SL, unsupervised learning gives data without predefined labels for training. Data comprises only input (x) without output (y).

Algorithm must find patterns in the data.

Unsupervised learning
Find something interesting
from unlabeled data.

(1) Clustering: Organize data into similar groups according to shared characteristics.

Clustering: Grouping customers

Terminology:

(1) Training Data, and its Notation

Notation:
- x = input variable
- y = output variable
- (x, y) = single training example
- $(x^{(i)}, y^{(i)})$ = i^{th} training example
- m = # training examples
- $f(x)$ = function output, formerly $f(x^{(i)}, w, b)$

Terminology:

Training Set: Data used to train the model

Size in feet² (x)	Price in $1000's ($y$)
2104	460
1416	232
1534	315
852	178
…	…

$x^{(1)} = 2104$, $y^{(1)} = 460$

$x^{(2)} = 1416$, $y^{(2)} = 232$

$x^{(3)} = 1534$, $y^{(3)} = 315$

$x^{(4)} = 852$, $y^{(4)} = 178$

Notation:
- x = "input" variable
- y = "output" variable
- (x, y) = single training example
- $(x^{(i)}, y^{(i)})$ = i^{th} training example

$(x^{(1)}, y^{(1)}) = (2104, 460)$

$(x^{(2)}, y^{(2)}) = (1416, 232)$

$(x^{(3)}, y^{(3)}) = (1534, 315)$

$(x^{(4)}, y^{(4)}) = (852, 178)$

$\hat{y} = f(x)$ (estimated y)

$f(x) = wx + b$

$f_{w,b}(x) = wx + b$

Linear regression

Univariate linear regression (one feature variable)

---

(2) Cost Function

Linear regression:

$f_{w,b}(x) = wx + b$

Parameters: w, b

Cost function: $J(w, b)$

Goal: $\min_{w,b} J(w, b)$

Squared error cost function

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$$

Model: $f_{w,b}(x) = wx + b$

parameters: w, b

cost function: $J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$

goal: $\min_{w,b} J(w,b)$

Simplified (fixed $b = 0$)

Model: $f_w(x) = wx$

Cost: $J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2$

Goal: $\min_w J(w)$

Gradient descent (GD) is a general procedure for minimizing any function, including cost functions. It adjusts parameters (w, b) to reduce cost J(w,b) iteratively, moving in the direction of steepest descent (negative gradient). Crucial for training models.

Algorithm:

Repeat until convergence: take small steps in opposite direction of steepest ascent.

1) Initialize w and b (commonly by random initialization or all zeroes)

2) Keep changing w and b to reduce $J(w,b)$ until we settle at or near a minimum.

$$\frac{\partial}{\partial w}J(w,b) \qquad \frac{\partial}{\partial b}J(w,b)$$

Slope: positive, take a small step to move in negative direction of positive gradient.

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

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

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

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

Notation:

Repeat until convergence {

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

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

}

Assignment operator

Learning rate (usually small positive number between 0 and 1)

Correct: Simultaneous update

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

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

$w = tmp_w$

$b = tmp_b$

Incorrect

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

$w = tmp_w$

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

$b = tmp_b$

Learning rate (α): α big → $w = w - \frac{\partial}{\partial w}J(w,b)$

Big learning rate:
- "Overshoot" minimum
- Fail to converge, diverge

Small learning rate:
- Take small steps
- Takes a long time

Can reach local minimum with fixed learning rate

Near a local minimum,
- Derivative becomes smaller
- Update steps become smaller

Can reach minimum without decreasing learning rate α

Linear regression model

$f_{w,b}(x) = wx + b$

Cost function:

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$$

Gradient Descent Algorithm:

Repeat until convergence {

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

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

}

$$\frac{\partial}{\partial w}J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial}{\partial w}J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})^2$$

Derivatives

$$\frac{\partial}{\partial w}J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})\cdot 2x^{(i)}$$

$$\frac{\partial}{\partial b}J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})\cdot 2$$

Update:

Gradient descent algorithm

Repeat until convergence {

$$w = w - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})\cdot x^{(i)}$$

$$b = b - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})$$

}

Convex function: A function with bowl shape.

Proposed squared error function with linear regression is not convex/bowl-shaped, so local minima matter. Switch to logistic regression cost function for better convexity.

Batch gradient descent (BGD) is an optimization algorithm for training models that uses all training examples in each step to update parameters.

"Batch": Each step of gradient descent uses all the training examples.