

Week 6 - Regression, Gradient Descent, Clustering and K-means

Sequential Gradient Descent (SGD)

- Follow the **slope** (the **derivative** of a $f(x)$) to find the best w to minimize the loss

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

- Linear function $f(x) = wx + b$, slope =

D

- Quadratic function $f(x) = x^2$, slope = delta f over delta x

$$\frac{\Delta f}{\Delta x}$$

- Derivative

$$d(x^n) = nx^{n-1}$$

- How to adjust the model parameters?**

- Derivative is **zero** at **minima** and **maxima** of a function, $f'(x) = 0$, which is used as way of finding optimal values of objectives.
- In multiple dimensions, the **gradient** $f_w(w)$ (slope of the **tangent plane**) is a **vector of derivatives** in each dimension
 - $f(w_1, w_2) = w_1^2 + w_2^2$

- **Matrix notation:** $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = [w_1 \ w_2]^T$, $f_w(w) = \nabla f(w) = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \end{bmatrix}$ (

$\nabla f(w)$ pronounced as "nabla f" or "grad f")

- **The gradient gives a direction to which the function increases**

◦ Explanation

- General algorithm for finding a value of the model parameters w such that error function $F(w)$ is minimized, expressed using **multivariable calculus** as $F_w(w) = 0$ where F_w is the **(partial) derivative** of F with respect to vector w . (Note that this is actually a vector of D zeros, but we use the usual zero as a shorthand.)

$$F_w(w) = \begin{bmatrix} 0 \\ \dots \\ 0 \end{bmatrix}$$

- Idea: starting with a guess for w_n , take a "step" in direction of **steepest descent** of the loss function, $-F_w(w_n)$, use this as a better guess w_{n+1}
- Size of the step, $\alpha > 0$ ("learning rate"), determines how quickly the minimum is reached, but can **overshoot** and also **diverge (发散)** if α is too large; not guaranteed to find the minimum, unless the error function is **convex (凸函数)** with respect to w
- If a **loss** function has one global minimum, then it is a **convex** function.
- More advanced versions are widely used in modern deep learning

◦ **SGD: algorithm**

- **Step 1. Initialization:** Select an initial guess for w_0 , a convergence tolerance $\varepsilon > 0$, step size (learning rate) parameter $\alpha > 0$, set iteration number $n=0$

- **Step 2. Gradient descent step:** Compute new model parameters, $w_{n+1} = w_n - \alpha F_w(w_n)$
- **Step 3. Convergence test:** Compute new loss function value $F(w_{n+1})$, and loss function improvement, $\Delta F = |F(w_{n+1}) - F(w_n)|$ and if $\Delta F < \varepsilon$, exit with solution $w^* = w_{n+1}$ (when the difference between current Loss and last Lost \leq tolerance, it converges.)
- **Step 4. Iteration:** update $n=n+1$ and go to step 2.

```
def sgd():
    '''sequential gradient descent'''
    # step 1
    select an initial guess for w0
    select convergence tolerance  $\varepsilon > 0$ 
    select  $\alpha > 0$ 

    for n in range(R):
        # step 2
         $w_{n+1} = w_n - \alpha * F_w(w_n)$ 

        # step 3
         $\Delta F = \text{abs}(F(w_{n+1}) - F(w_n))$ 
        if  $\Delta F < \varepsilon$ :
            return  $w^* = w_{n+1}$ 

        # step 4: go back to step 2
```

- Euclidean distance

$$d = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

Supervised Learning

Regression

1. Regression model (D-dimensional):

$$\mathbf{D}(w^T \text{transpose } x)$$

- For linear regression, we transform $f(x) = wx + b$ to $f(x) = w_1x^1 + w_2x^2$ where w_2x^2 is b . Make it easier to compute the sum of square error **(we represent it as the latter, because transform can mean linear transformation) (w2 is b, x2 is set to 1)**

- sum of squares error function:

D

- Problem: find the optimal set of w

$$w^* = \arg \min_{w' \in W} F(w')$$

- Gradient with respect to w :

D

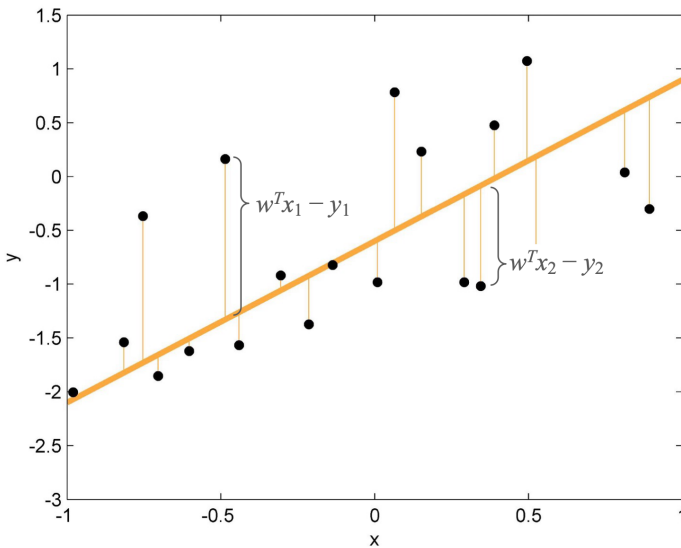
(multiply the result by 2 to account for the derivative of the squared term itself which is 2)

- SGD parameter update step

D

- Sum of Squared Error (SSE)

$$SSE = \sum_{i=1}^k \sum_{j=1}^{n_i} ||x_{ij} - \mu_i||^2$$



- Large error (bad fit):

$$F(w) = \sum_{i=1}^N (w^T x_i - y_i)^2$$

$$F(w) = 11.65$$

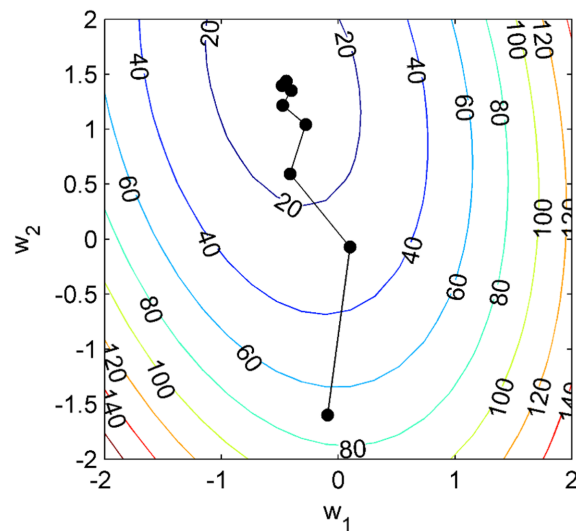
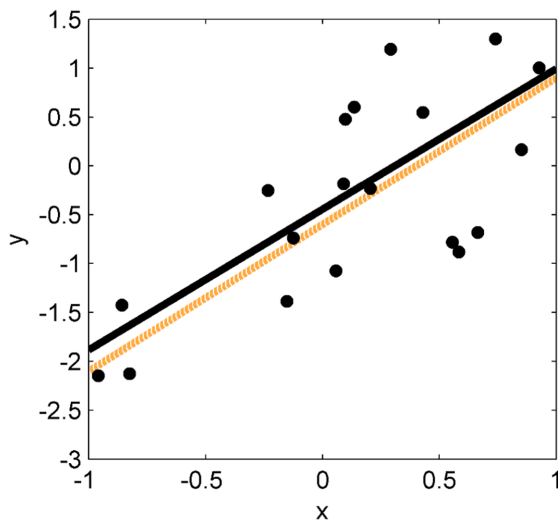
- Error is sum of squares of perpendicular distances to best fit line

perpendicular distances

Linear regression: SGD in action

39014916

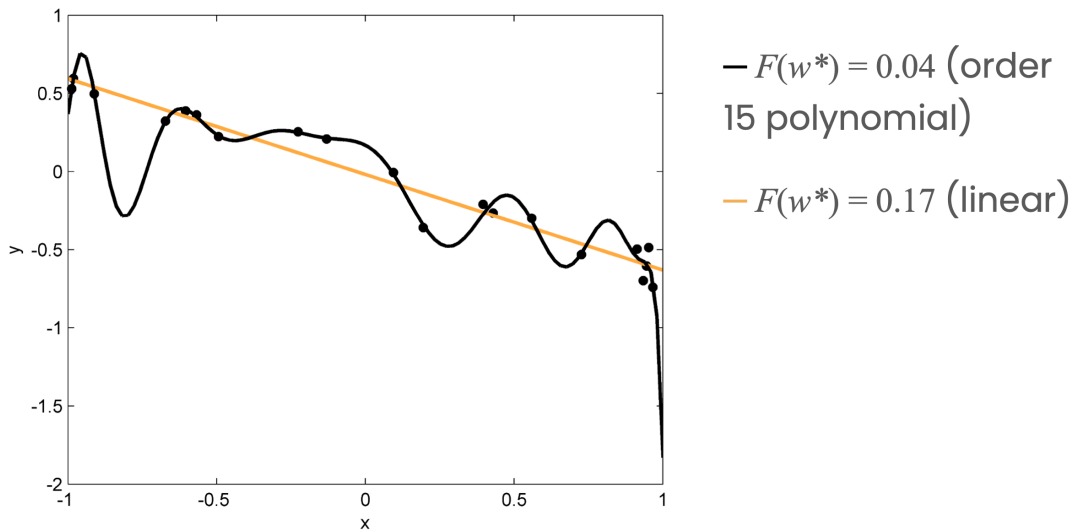
$n=7$, $F(w)=11.69$, $w_7=[-0.44, 1.44]$, $\Delta F=0.03 < \varepsilon = 0.05$ (exit), $w^*=[-0.44, 1.44]$



1. **A trade off between model complexity and test error:** common feature of most machine learning models: want a model which is as simple as possible but no simpler (**Occam's razor**)

Model complexity and Occam's razor

20017510



Unsupervised Learning

Clustering: K-means (minimizes the **distance** between all data points and their assigned **clusters**)

1. Given N data points in D -dimensional data space R^D , **clustering** finds the optimal way to **partition** the set of data into K groups
2. Conditions
 - a. High intra-cluster similarity 类内相似度高
 - b. Low inter-cluster similarity 类间相似度低
3. For a fixed value of clusters, K ,
 - a. Assign a cluster to each data point, e.g.,

$$x_1 = \begin{bmatrix} k_1 & k_2 \\ 0 & 1 \end{bmatrix}$$

- b. Compute the average value of each feature for the cluster (**indicator function** notation X_{ik})

$$X_{ik} = \begin{cases} 1 & \text{data item } x_i \text{ is assigned to cluster } k \\ 0 & \text{otherwise} \end{cases} \quad (10.1)$$

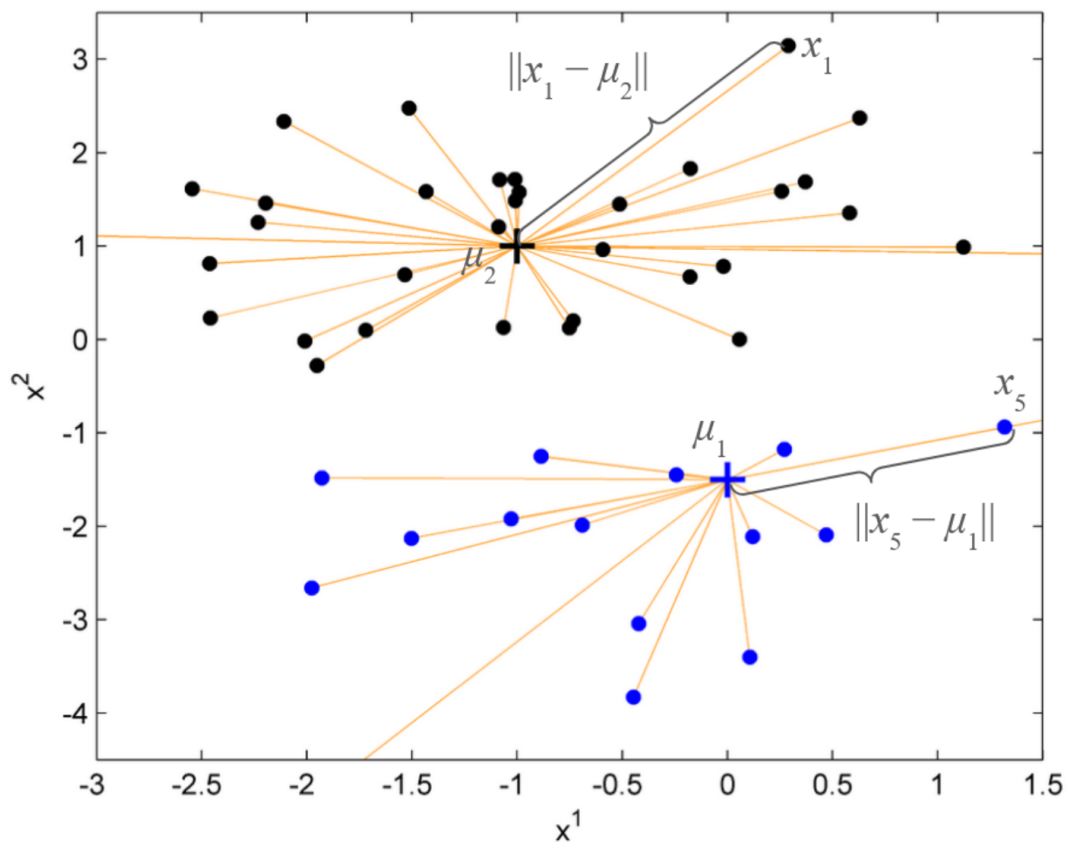
As an example, if we have data x_1, \dots, x_5 and $K = 3$ classes, then the configuration,

$$X = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad (10.2)$$

$$\mu_1 = \frac{1}{N_k} \sum_{i=1}^N X_{ik} \cdot x_i$$

- c. Compute the Euclidean distance between each point in the cluster and its average

$$\text{Euclidean Distance} = \|x_i - \mu_k\|$$



- d. **K-means objective**, a widely-used measure of **clustering quality**.
The sum of squared distances between all data points and their assigned clusters in the set partition X

$$F(X, \mu) = \sum_{i=1}^N \sum_{k=1}^K X_{ik} \cdot \|x_i - \mu_i\|^2$$

$$\|v\|^2 = \sum_{d=1}^D (v^d)^2$$

- e. K-means clustering attempts to solve the following optimization problem

$$(X^*, \mu^*) = \arg \min_{(X', \mu') \in W} F(X', \mu')$$

where W is the set of all possible partitions (i.e. just those indicators where each data item is assigned to a unique class), along with their corresponding centroid averages.

4. Algorithm

- Step 1. Initialization: Select an initial guess for all μ_k^0 for $k=1, 2, \dots, K$, set iteration number $n=0$
- Step 2. Update configuration: Set $X^{n+1}=0$, except where

$$k = \arg \min_{k'=1,2,\dots,K} \|x_i - \mu_{k'}\|^2$$

for which

$$X_{ik}^{n+1} = 1$$

- Step 3. Update **centroids μ** : Compute cluster averages,

$$\mu_k^{n+1} = \frac{1}{N_k} \sum_{i=1}^N X_{ik}^{n+1} x_i$$

where

$$N_k \sum_{i=1}^N X_{ik}^{n+1}$$

- d. Step 4. Convergence check: If $n > 0$ and $X^{n+1} = X^n$ then **exit with solution $X^* = X^{n+1}$ and $\mu^* = \mu^{n+1}$**
- e. Step 5. Iteration: update $n = n + 1$ and go back to step 2.

```
# step 1
initial guess for all  $\mu$  with a cluster  $k$ 

for n in range(R):
    # step 2 Update configuration

    # step 3 Update centroids

    # step 4 Convergence check
    if  $X^{n+1} = X^n$ :
        return  $X^* = X^{n+1}$ ,  $\mu^* = \mu^{n+1}$ 
```

f. K-means clustering: analysis

- i. $F(X_{n+1}, \mu_{n+1}) \leq F(X_n, \mu_n)$ i.e. the **K-means objective function is never increasing**

5. Applications

- a. Compress gray scale images (image dictionary-based methods)

6. Compute the value of the centroid as the average of the x value and the average of the y value.

$$centroid = (x_{avg}, y_{avg})$$

- 7. **Euclidean distance** (distance from the centroid = sum of squared difference for each dimension)

$$(x_{currentPoint} - x_{centroid})^2 + (y_{currentPoint} - y_{centroid})^2$$

- Support session

Tutorial Consolidation Week

1. Consider a small town with five coffee shop spots. Each café was evaluated by analysts in terms of its popularity (based on customer ratings and footfall) and accessibility (based on its proximity to transportation links or central areas in the town). The numerical score for each category varied from 1 to 10.

Coffee Shop	Popularity	Accessibility
A	7	8
B	4	6
C	9	6
D	5	7
E	8	5

Suppose we wish to categorise these coffee shops into two groups to assist new residents or visitors in town in quickly identifying the type of cafés they might be interested in. Utilise the K-means algorithm to group the coffee shops. Start with a "qualified guess" by supposing the coffee shops will be grouped between those well-evaluated ($\mu_1^0 = [8 \ 8]$) and those poorly evaluated for both features ($\mu_2^0 = [3 \ 3]$).

2. Consider the task of sorting a list of unique letters, such as $[C, A, R, D]$.

- How can this combinatorial optimization problem be mathematically formalised?
- List the configuration set and problem size of this task.
- Construct the computational graph representing the fully exhaustive SDP for solving this task. Identify the optimal solution in your graph.
- Use the greedy SDP to solve this task, and construct the computational graph representing this solution. For this specific problem, would the greedy solution be great? Explain.

Handwritten notes on the screen include:

- Centroids: $[7 \ 8]$ and $[3 \ 3]$
- Distance calculations: $(7-8)^2 + (8-8)^2 = 1$ and $(9-3)^2 + (6-3)^2 = 41$
- Assignment: $k=2$ with a vertical line separating the two groups.