The Mathematics of Machine Learning 2025

Lecture 10

Tim van Erven

University of Amsterdam

Outline

Unsupervised Learning: K-Means Clustering

1. Intro: Unsupervised Learning
2. The K-Means Algorithm
3. Convergence to Local Minimum
4. How to Choose K?
5. Example: Clustering Human Tumor Microarray Data

Stochastic Optimization

6. Gradient Descent
7. Stochastic Gradient Descent

Intro: Unsupervised Learning

Supervised: $T = \begin{pmatrix} Y_1 \\ X_1 \end{pmatrix}, \ldots, \begin{pmatrix} Y_N \\ X_N \end{pmatrix}$

Unsupervised: $T = X_1, \ldots, X_N$

What can we do with such data?

Intro: Unsupervised Learning

Supervised: $$T = \begin{pmatrix} Y_1 \\ X_1 \end{pmatrix}, \ldots, \begin{pmatrix} Y_N \\ X_N \end{pmatrix}$$

Unsupervised: $$T = X_1, \ldots, X_N$$

What can we do with such data?

Clustering: split data into groups of points (clusters) that are similar

1. What do you mean by "similar"?
2. What kind of clusters are you looking for?

Illustration: Splitting Data into Three Clusters

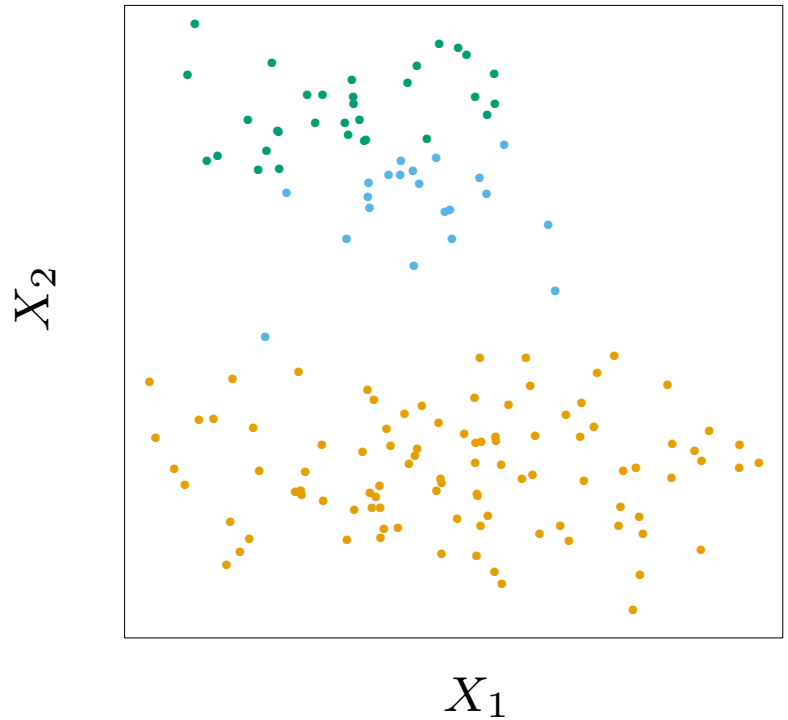


FIGURE 14.4. *Simulated data in the plane, clustered into three classes (represented by orange, blue and green) by the K-means clustering algorithm*

K-Means: Objective

1. Q. What do you mean by **"similar"**?
   A. **Squared Euclidean distance** between X_i, X_i':

$$d^2(X_i, X_i') = \|X_i - X_i'\|^2$$

2. Q. What **kind of clusters** are you looking for?
   A. Find K clusters such that **distance to mean within each cluster is small**.

K-Means: Objective

1. Q. What do you mean by **"similar"**?
   A. **Squared Euclidean distance** between X_i, X_i':

$$d^2(X_i, X_i') = \|X_i - X_i'\|^2$$

2. Q. What **kind of clusters** are you looking for?
   A. Find K clusters such that **distance to mean within each cluster is small**.

$C(i) \in \{1, \ldots, K\}$ is cluster assigned to data point X_i.

K-Means: Objective

1. Q. What do you mean by **"similar"**?
   A. **Squared Euclidean distance** between X_i, X_i':

$$d^2(X_i, X_i') = \|X_i - X_i'\|^2$$

2. Q. What **kind of clusters** are you looking for?
   A. Find K clusters such that **distance to mean within each cluster is small**.

$C(i) \in \{1, \ldots, K\}$ is cluster assigned to data point X_i.

$$\text{Choose } C \text{ to minimize } \quad \sum_{k=1}^{K} \sum_{i:C(i)=k} d^2(X_i, \mu_k),$$

where $\mu_k = \frac{1}{N_k} \sum_{i:C(i)=k} X_i$ is mean of points in cluster k with $N_k = |\{i : C(i) = k\}|$ the number of points in cluster k.

K-Means: Objective

1. Q. What do you mean by **"similar"**?
   A. **Squared Euclidean distance** between X_i, X_i':

$$d^2(X_i, X_i') = \|X_i - X_i'\|^2$$

2. Q. What **kind of clusters** are you looking for?
   A. Find K clusters such that **distance to mean within each cluster is small**.

$C(i) \in \{1, \ldots, K\}$ is cluster assigned to data point X_i.

$$\text{Choose } C \text{ to minimize } \quad \sum_{k=1}^{K} \sum_{i:C(i)=k} d^2(X_i, \mu_k),$$

where $\mu_k = \frac{1}{N_k} \sum_{i:C(i)=k} X_i$ is mean of points in cluster k with $N_k = |\{i : C(i) = k\}|$ the number of points in cluster k.

NB. Mistake in formula in book! (multiplies by N_k in each cluster)

K-Means: Algorithm

Choose C to minimize $$\sum_{k=1}^{K} \sum_{i:C(i)=k} d^2(X_i, \mu_k) \qquad (\star)$$

- Finding an exact minimizing C is computationally intractable
- Instead, K-means approximates the minimum with a greedy algorithm:

K-Means: Algorithm

Choose C to minimize $$\sum_{k=1}^{K} \sum_{i:C(i)=k} d^2(X_i, \mu_k) \qquad (\star)$$

- Finding an exact minimizing C is computationally intractable
- Instead, K-means approximates the minimum with a greedy algorithm:

Algorithm:

1. Initialize cluster assignment C
2. For each k: set μ_k to current mean in cluster k
3. Given means $\mu_1, \ldots, \mu_K$, assign each X_i to nearest mean to get new C
4. Repeat from 2 until no changes in C

K-Means: Algorithm

$$\text{Choose } C \text{ to minimize } \sum_{k=1}^{K} \sum_{i:C(i)=k} d^2(X_i, \mu_k) \qquad (\star)$$

- Finding an exact minimizing C is computationally intractable
- Instead, K-means approximates the minimum with a greedy algorithm:

Algorithm:

1. Initialize cluster assignment C
2. For each k: set μ_k to current mean in cluster k
3. Given means $\mu_1, \ldots, \mu_K$, assign each X_i to nearest mean to get new C
4. Repeat from 2 until no changes in C

Initialization: E.g. set $\mu_1, \ldots, \mu_K$ equal to K randomly chosen points + run step 3.

Illustration of K-Means Clustering

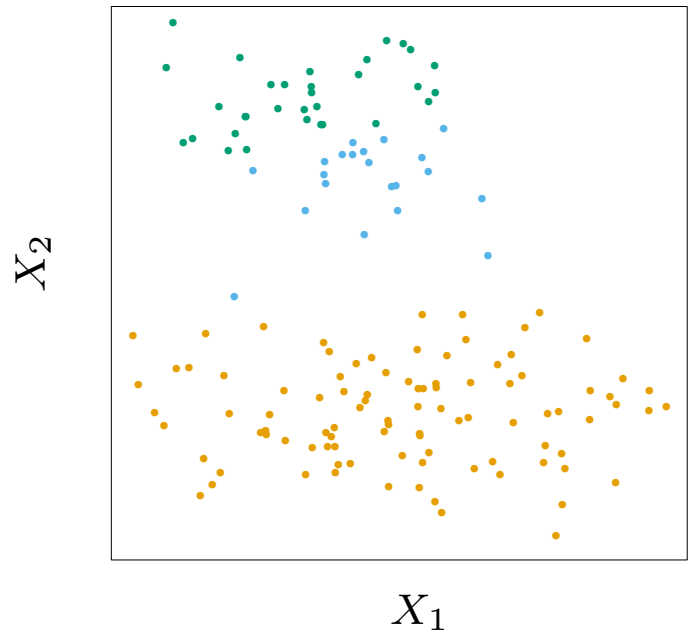


FIGURE 14.4. *Simulated data in the plane, clustered into three classes (represented by orange, blue and green) by the K-means clustering algorithm*

Animations of K-means:

- `https://shabal.in/visuals/kmeans/1.html`
- `https://www.youtube.com/watch?v=OrvltpuxVQQ`

Convergence to Local Minimum I

$$\sum_{k=1}^{K} \sum_{i:C(i)=k} d^2(X_i, \mu_k) \qquad (\star)$$

Algorithm:

1. Initialize cluster assignment C
2. For each k: set μ_k to current mean in cluster k
3. Given means $\mu_1, \ldots, \mu_K$, assign each X_i to nearest mean to get new C
4. Repeat from 2 until no changes in C

Observations:

- Step 2 minimizes $(\star)$ in $\mu_1, \ldots, \mu_K$ given C, because

$$\mu_k = \arg\min_{\mu} \sum_{i:C(i)=k} d^2(X_i, \mu)$$

- Step 3 minimizes $(\star)$ in C given $\mu_1, \ldots, \mu_K$.

Convergence to Local Minimum I

$$\sum_{k=1}^{K} \sum_{i:C(i)=k} d^2(X_i, \mu_k) \qquad (\star)$$

Observations:

- Step 2 minimizes $(\star)$ in $\mu_1, \ldots, \mu_K$ given C, because

$$\mu_k = \arg\min_{\mu} \sum_{i:C(i)=k} d^2(X_i, \mu)$$

- Step 3 minimizes $(\star)$ in C given $\mu_1, \ldots, \mu_K$.

Conclusion: K-means is guaranteed to stop eventually

- Thus: every round of K-means strictly decreases $(\star)$ if it changes C (assuming any ties in step 3 are always broken the same way)
- So the algorithm never revisits a past cluster assignment C
- But there are a finite number of possible C
- At some point C will stop changing: K-means has converged!

Convergence to Local Minimum II

Does K-means always converge to the **same clustering?**

Convergence to Local Minimum II

Does K-means always converge to the **same clustering?**

No, final result depends on initialization!
`https://shabal.in/visuals/kmeans/1.html`

Convergence to Local Minimum II

Does K-means always converge to the **same clustering?**

No, final result depends on initialization!
`https://shabal.in/visuals/kmeans/1.html`

Practise: Try several times with different initializations and select answer that minimizes $(\star)$
(helps if bad solutions, for which $(\star)$ is large, are not too common)

Convergence to Local Minimum II

Does K-means always converge to the **same clustering?**

No, final result depends on initialization!
`https://shabal.in/visuals/kmeans/1.html`

Practise: Try several times with different initializations and select answer that minimizes $(\star)$
(helps if bad solutions, for which $(\star)$ is large, are not too common)

Theory: Sufficient conditions that guarantee K-means converges (close to) global minimum of $(\star)$, but these require strong (unrealistic) assumptions on separability of the clusters

How to Choose K?

- K is a hyperparameter, so use cross-validation?

How to Choose K?

- K is a hyperparameter, so use cross-validation? Undefined because no labels $Y_1, \ldots, Y_N$

How to Choose K?

- K is a hyperparameter, so use cross-validation? Undefined because no labels $Y_1, \ldots, Y_N$
- **No objectively best choice for K!**

How to Choose K?

- K is a hyperparameter, so use cross-validation? Undefined because no labels $Y_1, \ldots, Y_N$
- **No objectively best choice for K!**

Observation: larger K generally lead to smaller $(\star)$

How to Choose K?

- K is a hyperparameter, so use cross-validation? Undefined because no labels $Y_1, \ldots, Y_N$
- **No objectively best choice for K!**

Observation: larger K generally lead to smaller $(\star)$

Possible ways to choose K:

Option 1: K is determined by context: e.g. assign N customers to K sales people.

How to Choose K?

- K is a hyperparameter, so use cross-validation? Undefined because no labels $Y_1, \ldots, Y_N$
- **No objectively best choice for K!**

Observation: larger K generally lead to smaller $(\star)$

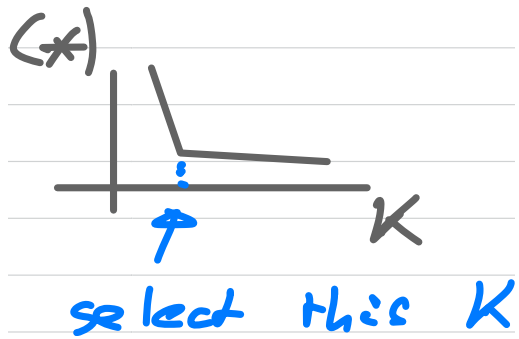
Possible ways to choose K:

Option 1: K is determined by context: e.g. assign N customers to K sales people.

Option 2: Imagine K^* "natural" clusters: distances within each cluster much smaller than distances between clusters.

- $K < K^*$: increasing K decreases $(\star)$ a lot
- $K > K^*$: increasing K decreases $(\star)$ a little.

Expect to (roughly) see a hinge like this:



Example from Book: Human Tumor Microarray Data

Data:

- Measurements $X_i \in \mathbb{R}^p$ of gene expressions for $p = 6830$ genes for $N = 64$ people.
- Each sample has a label Y_i that indicates the type of cancer, but this is **not** used in the clustering

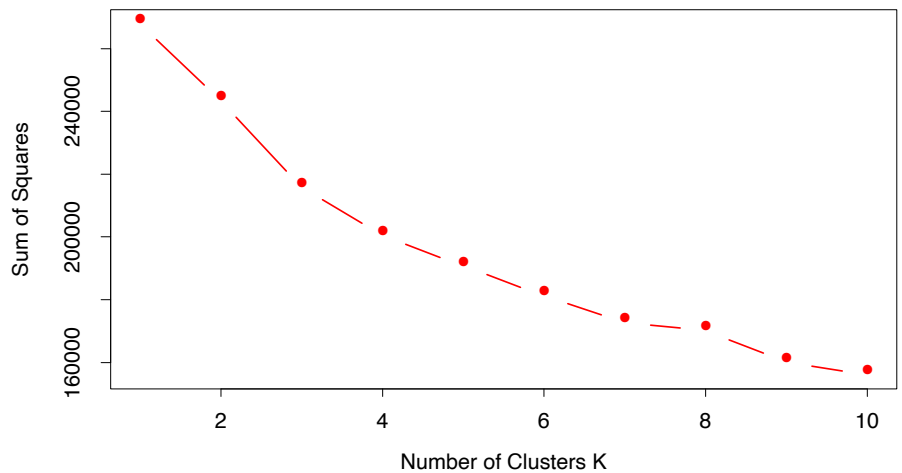


FIGURE 14.8. *Total within-cluster sum of squares for K-means clustering applied to the human tumor microarray data.*

Example from Book: Human Tumor Microarray Data

Data:

- Measurements $X_i \in \mathbb{R}^p$ of gene expressions for $p = 6830$ genes for $N = 64$ people.
- Each sample has a label Y_i that indicates the type of cancer, but this is **not** used in the clustering

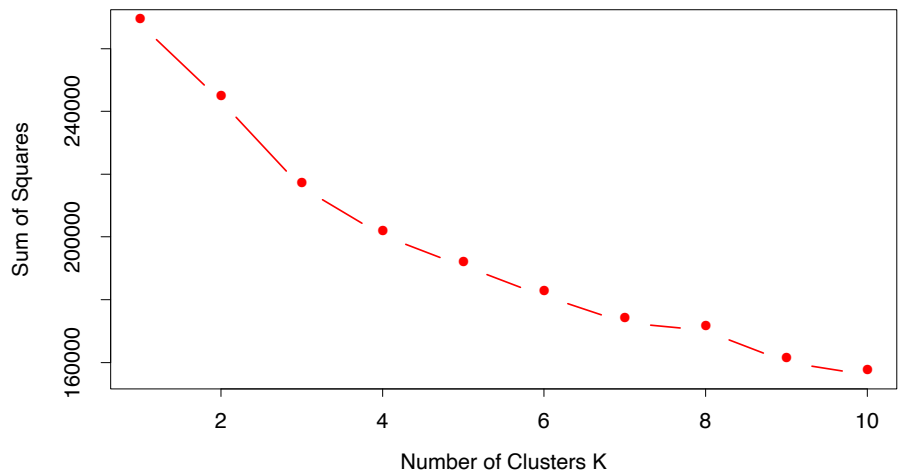


FIGURE 14.8. *Total within-cluster sum of squares for K-means clustering applied to the human tumor microarray data.*

NB Cannot use Option 2 to select K, because we do not see any hinge.

Example from Book: Human Tumor Microarray Data

Data:

- Measurements $X_i \in \mathbb{R}^p$ of gene expressions for $p = 6830$ genes for $N = 64$ people.
- Each sample has a label Y_i that indicates the type of cancer, but this is **not** used in the clustering

TABLE 14.2. *Human tumor data: number of cancer cases of each type, in each of the three clusters from K-means clustering.*

Cluster	Breast	CNS	Colon	K562	Leukemia	MCF7
1	3	5	0	0	0	0
2	2	0	0	2	6	2
3	2	0	7	0	0	0
Cluster	Melanoma	NSCLC	Ovarian	Prostate	Renal	Unknown
1	1	7	6	2	9	1
2	7	2	0	0	0	0
3	0	0	0	0	0	0

Using $K = 3$ (for illustration)

Outline

Unsupervised Learning: K-Means Clustering

1. Intro: Unsupervised Learning
2. The K-Means Algorithm
3. Convergence to Local Minimum
4. How to Choose K?
5. Example: Clustering Human Tumor Microarray Data

Stochastic Optimization

6. Gradient Descent
7. Stochastic Gradient Descent

Non-stochastic Optimization

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} F(\boldsymbol{\beta}) \qquad \boldsymbol{\beta} = (\beta_1, \dots, \beta_p)$$

Using bold to distinguish vector $\boldsymbol{\beta}$ from scalar β_j

Non-stochastic Optimization

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} F(\boldsymbol{\beta}) \qquad \boldsymbol{\beta} = (\beta_1, \dots, \beta_p)$$

Using bold to distinguish vector $\boldsymbol{\beta}$ from scalar β_j

Examples:

$$F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - X_i^\top \boldsymbol{\beta})^2 \qquad \text{(least squares)}$$

$$F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - X_i^\top \boldsymbol{\beta})^2 + \lambda\sum_{j=2}^{p}\beta_j^2 \qquad \text{(ridge regression)}$$

$$F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - X_i^\top \boldsymbol{\beta})^2 + \lambda\sum_{j=2}^{p}|\beta_j| \qquad \text{(lasso)}$$

$$F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N}\log(1 + e^{-Y_i X_i^\top \boldsymbol{\beta}}) \qquad \text{(logistic regression)}$$

- Dividing by N here, which makes no difference for optimal parameters $\boldsymbol{\beta}^*$ (if we divide λ by N as well)
- F is **convex** in all these examples

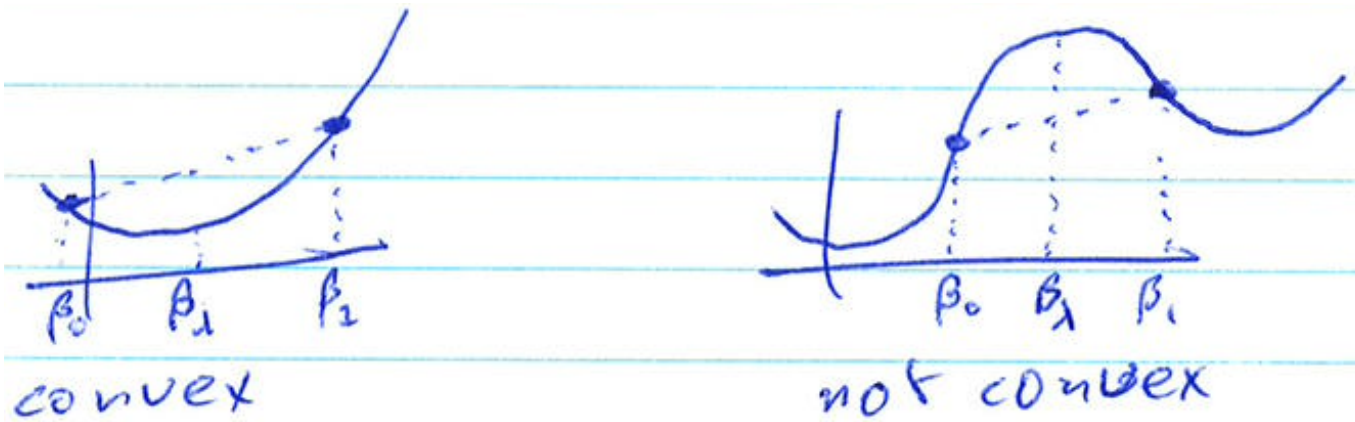
Convexity

For any two parameter vectors $\boldsymbol{\beta}_0$ and $\boldsymbol{\beta}_1$, the line between $F(\boldsymbol{\beta}_0)$ and $F(\boldsymbol{\beta}_1)$ lies above the function F:

$$F(\boldsymbol{\beta}_\lambda) \leq (1-\lambda)F(\boldsymbol{\beta}_0) + \lambda F(\boldsymbol{\beta}_1) \qquad \text{for all } \lambda \in [0,1],$$

where

$$\boldsymbol{\beta}_\lambda = (1-\lambda)\boldsymbol{\beta}_0 + \lambda\boldsymbol{\beta}_1$$



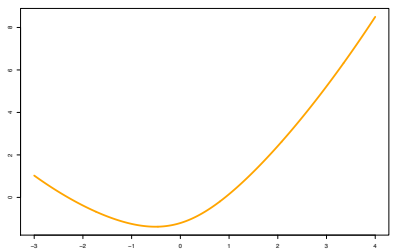
Convexity

For any two parameter vectors $\boldsymbol{\beta}_0$ and $\boldsymbol{\beta}_1$, the line between $F(\boldsymbol{\beta}_0)$ and $F(\boldsymbol{\beta}_1)$ lies above the function F:

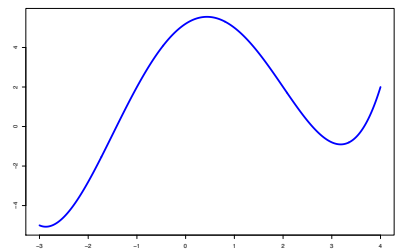
$$F(\boldsymbol{\beta}_\lambda) \leq (1-\lambda)F(\boldsymbol{\beta}_0) + \lambda F(\boldsymbol{\beta}_1) \qquad \text{for all } \lambda \in [0,1],$$

where

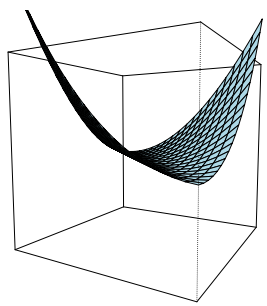
$$\boldsymbol{\beta}_\lambda = (1-\lambda)\boldsymbol{\beta}_0 + \lambda\boldsymbol{\beta}_1$$



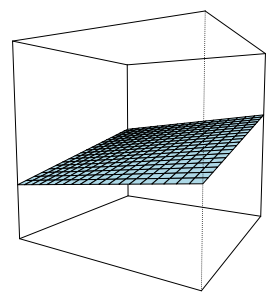
(a) Convex function



(b) Non-convex function



(c) Convex function of two parameters



(d) Another convex function of two parameters (this function is actually linear)

Gradient Descent

Convexity: $\Rightarrow$ Can find minimum by making small improvements to the parameters $\boldsymbol{\beta}$:

new parameters

previous parameters

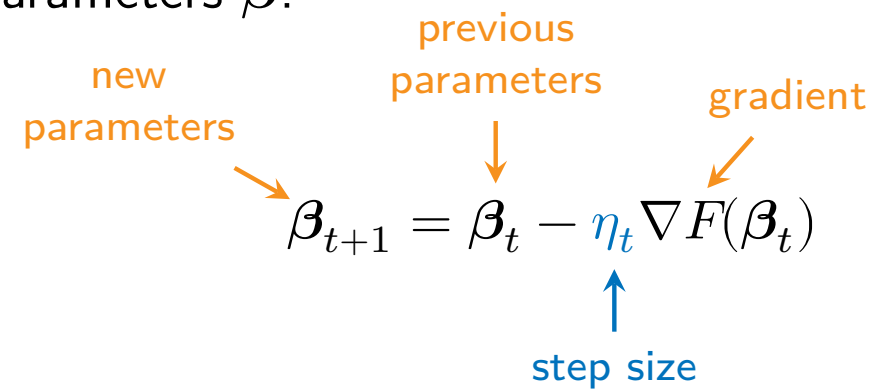
gradient

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \nabla F(\boldsymbol{\beta}_t) \qquad \text{(gradient descent)}$$

step size

Gradient Descent

Convexity: $\Rightarrow$ Can find minimum by making small improvements to the parameters $\boldsymbol{\beta}$:



Gradient: vector of partial derivatives

$$\nabla F(\boldsymbol{\beta}) = \begin{pmatrix} \frac{\partial}{\partial \beta_1} F(\boldsymbol{\beta}) \\ \vdots \\ \frac{\partial}{\partial \beta_p} F(\boldsymbol{\beta}) \end{pmatrix}$$

- Vector pointing in direction that is the most up from $\boldsymbol{\beta}$
- So move in direction of negative gradient to go the most down

Gradient Descent

Convexity: $\Rightarrow$ Can find minimum by making small improvements to the parameters $\boldsymbol{\beta}$:

new parameters

previous parameters

gradient

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \nabla F(\boldsymbol{\beta}_t) \qquad \text{(gradient descent)}$$

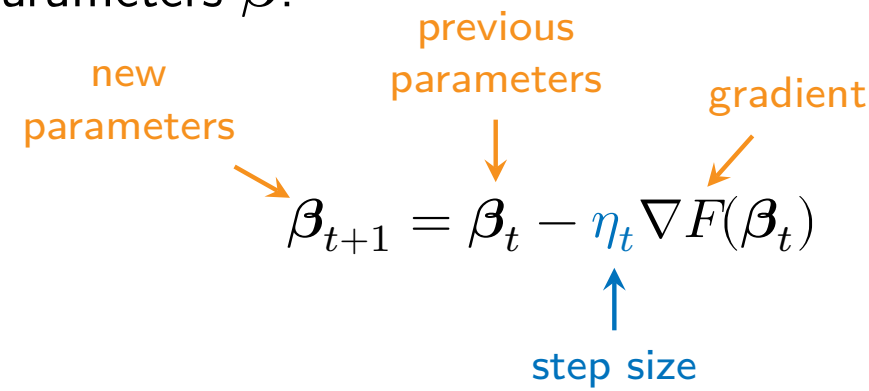
step size

Best choice of $\eta_t > 0$:

- Depends on type of function.
- Common choices: $\eta_t = C$, $\eta_t = C/\sqrt{t}$ and $\eta_t = C/t$, where $C > 0$ is some constant.

Gradient Descent

Convexity: $\Rightarrow$ Can find minimum by making small improvements to the parameters $\boldsymbol{\beta}$:



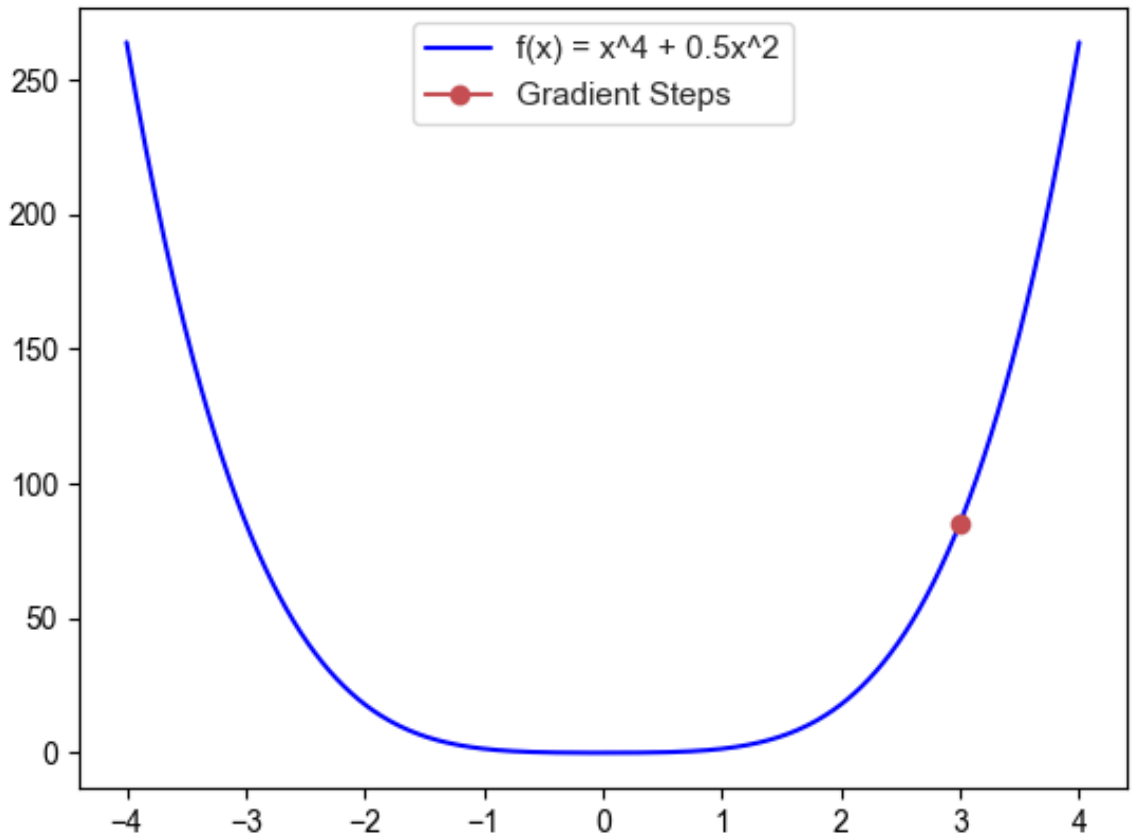
Exists Rich Theory: For instance [1], if F is convex and γ-smooth[1], then, for $\eta_t = 1/\gamma$:

$$F(\boldsymbol{\beta}_t) - F(\boldsymbol{\beta}^*) \leq \frac{2\gamma \|\boldsymbol{\beta}_1 - \boldsymbol{\beta}^*\|^2}{t-1}$$

- Bound is larger if starting distance from optimum $\|\boldsymbol{\beta}_1 - \boldsymbol{\beta}^*\|$ is larger
- Bound decreases with number of steps $t \Rightarrow$ closer to optimum

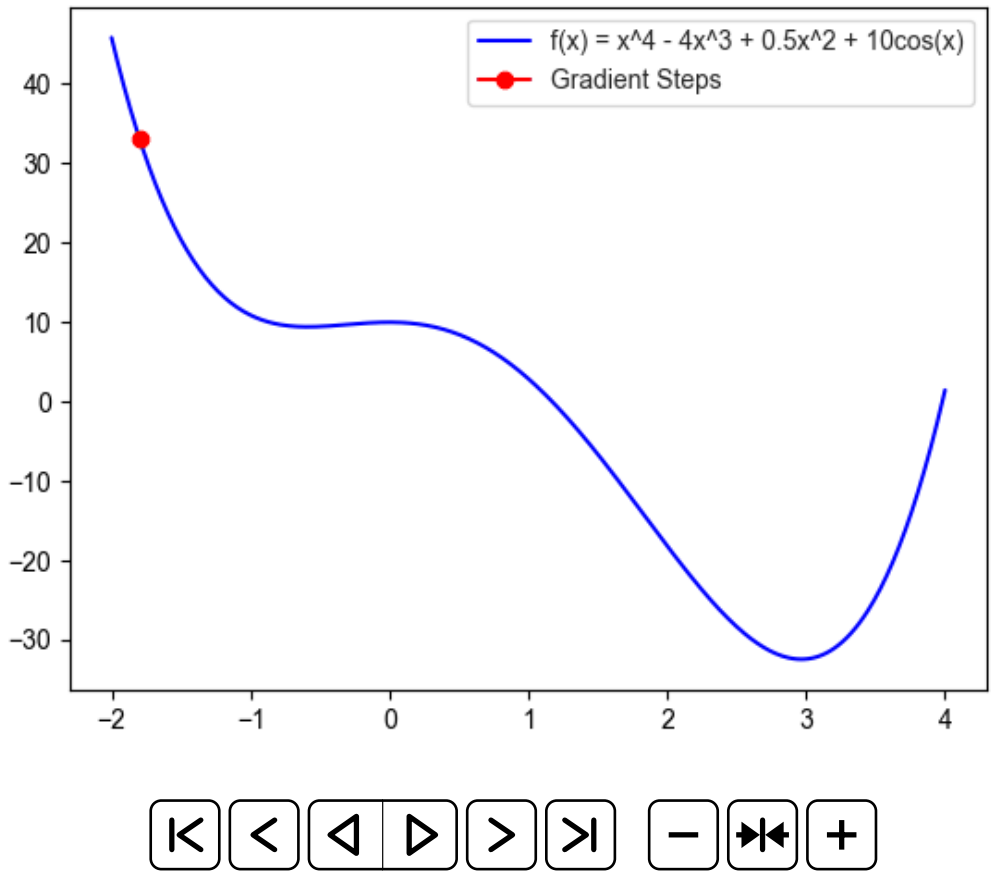
[1] γ-Smooth means that the second derivative in any direction is at most γ or, equivalently, that the gradient is γ-Lipschitz continuous.

Example: Gradient Descent for Convex Function



Animation code by Sdmuhsin, https://medium.com/@sdmuhsin3011/master-gradient-descent-through-visual-animation-8d2858afefa8

Example: Gradient Descent for Non-Convex Function



Animation code by Sdmuhsin, https://medium.com/@sdmuhsin3011/master-gradient-descent-through-visual-animation-8d2858afefa8

Stochastic Optimization

Suppose

$$F(\boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^{N} f_i(\boldsymbol{\beta})$$

for some (convex) functions $f_1, \ldots, f_N$.

Stochastic Optimization

Suppose

$$F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{\beta})$$

for some (convex) functions $f_1, \ldots, f_N$.

Then computing

$$\nabla F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\boldsymbol{\beta})$$

takes a **lot of computation when N is large.**

All examples are of this form!

- Every f_i corresponds to one data point (X_i, Y_i).
- For Ridge and Lasso take $f_i(\boldsymbol{\beta}) = (Y_i - X_i^\top \boldsymbol{\beta})^2 + \lambda \operatorname{pen}(\boldsymbol{\beta})$.

Stochastic Gradient Descent

$$\nabla F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\boldsymbol{\beta})$$

Pick one data point at random and do gradient descent step based on only that data point instead of the whole data:

Choose i_t randomly from $\{1, \ldots, N\}$

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \nabla f_{i_t}(\boldsymbol{\beta}_t)$$

Stochastic Gradient Descent

$$\nabla F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\boldsymbol{\beta})$$

Pick one data point at random and do gradient descent step based on only that data point instead of the whole data:

$$\text{Choose } i_t \text{ randomly from } \{1, \dots, N\}$$

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \nabla f_{i_t}(\boldsymbol{\beta}_t)$$

- Can do N steps in same computation time as one update of regular gradient descent

Stochastic Gradient Descent

$$\nabla F(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\boldsymbol{\beta})$$

Pick one data point at random and do gradient descent step based on only that data point instead of the whole data:

$$\text{Choose } i_t \text{ randomly from } \{1, \dots, N\}$$

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \nabla f_{i_t}(\boldsymbol{\beta}_t)$$

- Can do N steps in same computation time as one update of regular gradient descent
- $\nabla f_{i_t}(\boldsymbol{\beta}_t)$ is **unbiased** estimate of $\nabla F(\boldsymbol{\beta}_t)$:

$$\mathop{\mathbb{E}}_{i_t}[\nabla f_{i_t}(\boldsymbol{\beta}_t)] = \sum_{i_t=1}^{N} \frac{1}{N} \nabla f_{i_t}(\boldsymbol{\beta}_t) = \nabla F(\boldsymbol{\beta}_t).$$

- So we go in **right direction on average** with much less computation

Stochastic Gradient Descent

Choose i_t randomly from $\{1, \dots, N\}$

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \nabla f_{i_t}(\boldsymbol{\beta}_t)$$

- Can do N steps in same computation time as one update of regular gradient descent
- $\nabla f_{i_t}(\boldsymbol{\beta}_t)$ is **unbiased** estimate of $\nabla F(\boldsymbol{\beta}_t)$:

$$\mathop{\mathbb{E}}_{i_t}[\nabla f_{i_t}(\boldsymbol{\beta}_t)] = \sum_{i_t=1}^{N} \frac{1}{N} \nabla f_{i_t}(\boldsymbol{\beta}_t) = \nabla F(\boldsymbol{\beta}_t).$$

- So we go in **right direction on average** with much less computation

Theory:

- Analyse the effect of variance in the gradient estimates
- Methods that try to reduce the variance

References

S. Bubeck.
Convex optimization: Algorithms and complexity.
Foundations and Trends in Machine Learning, 8(3–4):231–358, 2015.