Lecture 2: Linear Models for Regression

Tao LIN

SoE, Westlake University

September 16, 2025

WESTLAKE UNIVERSITY | SCHOOL OF ENGINEERING

This lecture:

- Basic concept of regression and classification
- Linear regression
  - Definition
  - Gradient Descent (GD) optimization
- Normal Equation and Least Squares (maybe?)

Next lecture:

- Normal Equation and Least Squares
- The probabilistic interpretation of Linear Regression
- Maximum Likelihood Estimation (MLE)
- Over-fitting and Under-fitting
- Polynomial Regression, Ridge Regression, and Lasso

Reading materials

- Chapter 1, Stanford CS 229 Lecture Notes,
  https://cs229.stanford.edu/notes2022fall/main_notes.pdf
- Chapter 3.1, Bishop, Pattern Recognition and Machine Learning
- Dr. Robert M. Gower, Expected smoothness is the key to understanding minibatching for stochastic gradient methods,
  https://gowerrobert.github.io/pdf/talks/expected_smoothness.pdf

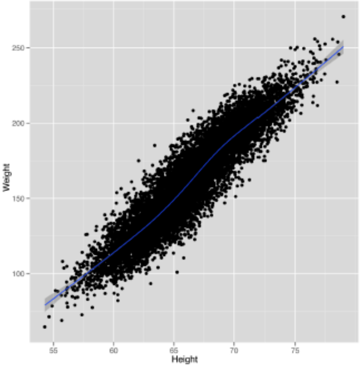
Reference

- EPFL, CS-433 Machine Learning, `https://github.com/epfml/ML_course`

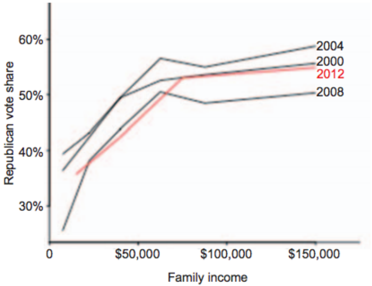
Table of Contents

Table of Contents

What is regression?

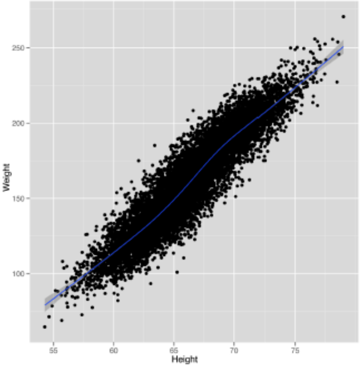


(a) Height is correlated with weight. Taken from “Machine Learning for Hackers"

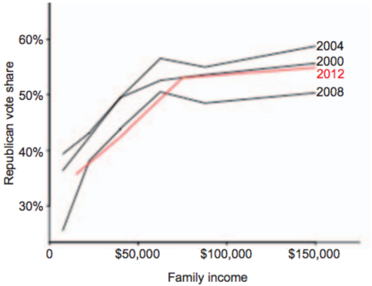


(b) Do rich people vote for republicans? Taken from Avi Feller et. al. 2013, Red state/blue state in 2012 elections.

What is regression?



(a) Height is correlated with weight. Taken from “Machine Learning for Hackers"

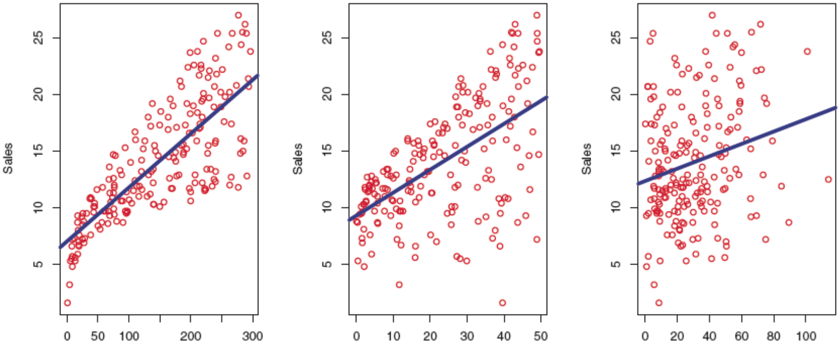


(b) Do rich people vote for republicans? Taken from Avi Feller et. al. 2013, Red state/blue state in 2012 elections.

Regression is to relate input variables to the output variable.

Dataset for regression

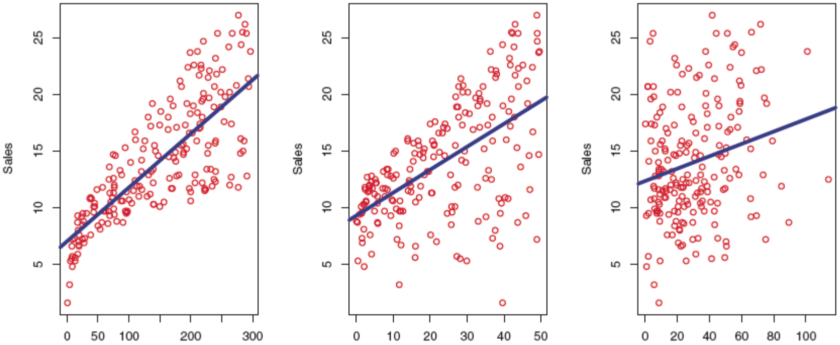
$$\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N} \in \mathcal{X} \times \mathcal{Y} \tag{1}$$



Dataset for regression

$$\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N} \in \mathcal{X} \times \mathcal{Y} \tag{1}$$

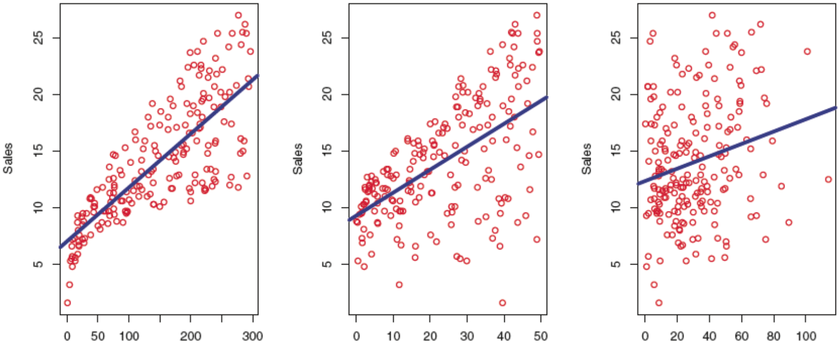
- data consists of pairs $(\mathbf{x}_n, y_n)$, where y_n is the n'th output and $\mathbf{x}_n$ is a vector of D inputs.



Dataset for regression

$$\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N} \in \mathcal{X} \times \mathcal{Y} \tag{1}$$

- data consists of pairs $(\mathbf{x}_n, y_n)$, where y_n is the n'th output and $\mathbf{x}_n$ is a vector of D inputs.
- The number of pairs N is the data-size and D is the dimensionality.



Two goals of regression

The regression function approximates the output y_n “well enough” given inputs $\mathbf{x}_n$.

$$y_n \approx f(\mathbf{x}_n), \text{ for all } n \tag{2}$$

Two goals of regression

The regression function approximates the output y_n “well enough” given inputs $\mathbf{x}_n$.

$$y_n \approx f(\mathbf{x}_n), \text{ for all } n \tag{2}$$

1. prediction: predict outputs for new inputs.

Two goals of regression

The regression function approximates the output y_n “well enough” given inputs $\mathbf{x}_n$.

$$y_n \approx f(\mathbf{x}_n), \text{ for all } n \tag{2}$$

1. prediction: predict outputs for new inputs.

   e.g., what is the weight of a person who is 170 cm tall?

Two goals of regression

The regression function approximates the output y_n "well enough" given inputs $\mathbf{x}_n$.

$$y_n \approx f(\mathbf{x}_n), \text{ for all } n \tag{2}$$

1. prediction: predict outputs for new inputs.

   e.g., what is the weight of a person who is 170 cm tall?

2. interpretation: understand the effect of the input on the output.

Two goals of regression

The regression function approximates the output y_n "well enough" given inputs $\mathbf{x}_n$.

$$y_n \approx f(\mathbf{x}_n), \text{ for all } n \tag{2}$$

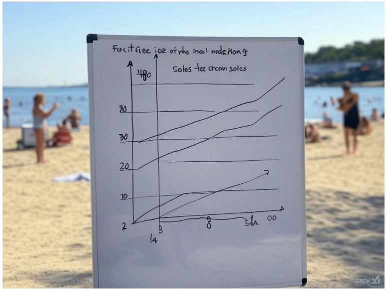
1. prediction: predict outputs for new inputs.

   e.g., what is the weight of a person who is 170 cm tall?

2. interpretation: understand the effect of the input on the output.

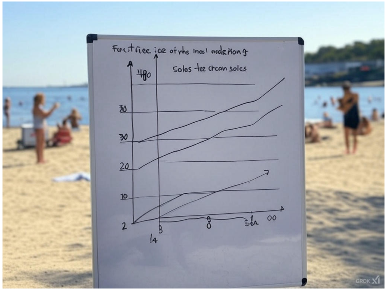
   e.g., are taller people heavier too?

Why correlation isn’t causation: spurious correlation



ice cream sales vs. drowning incidents

Why correlation isn't causation: spurious correlation

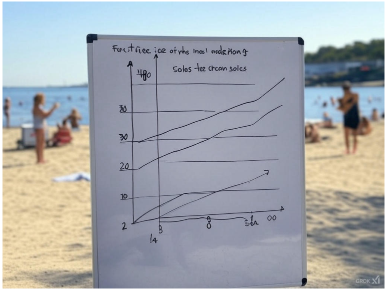


ice cream sales vs. drowning incidents

Remark 1 (Correlation $\neq$ Causation)

Regression finds a correlation not a causal relationship, so interpret your results with caution.

Why correlation isn't causation: spurious correlation



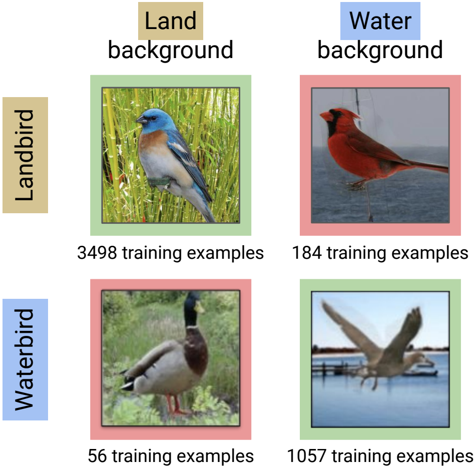
ice cream sales vs. drowning incidents

Brainstorm other examples!

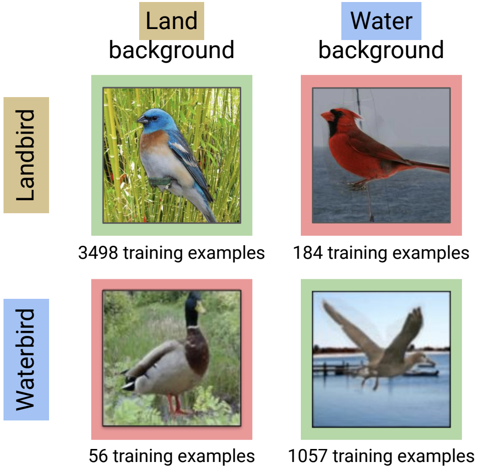
Remark 1 (Correlation $\neq$ Causation)

Regression finds a correlation not a causal relationship, so interpret your results with caution.

What is the concept of “spurious correlations”?

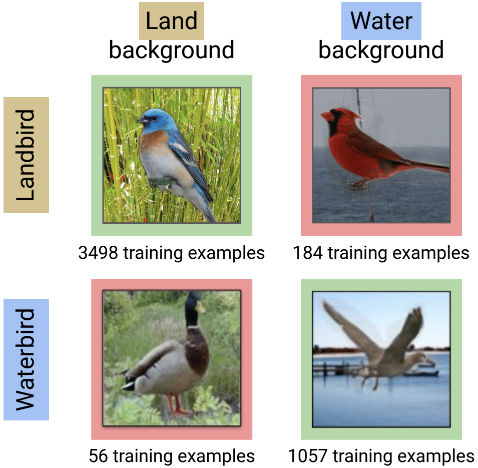


What is the concept of “spurious correlations”?



What is the concept of “spurious correlations”?

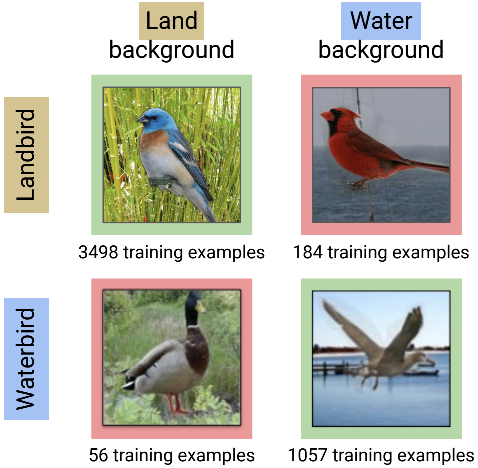
- The training set $\mathcal{D}_{tr}$ presents two spurious correlations, $\langle y, a\rangle$ and $\langle y', a\rangle$, i.e., a exists in two classes of images and can be mapped to two class labels.

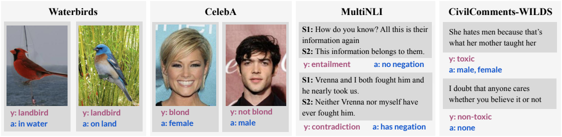


What is the concept of "spurious correlations"?

- The training set $\mathcal{D}_{tr}$ presents two spurious correlations, $\langle y, a \rangle$ and $\langle y', a \rangle$, i.e., a exists in two classes of images and can be mapped to two class labels.

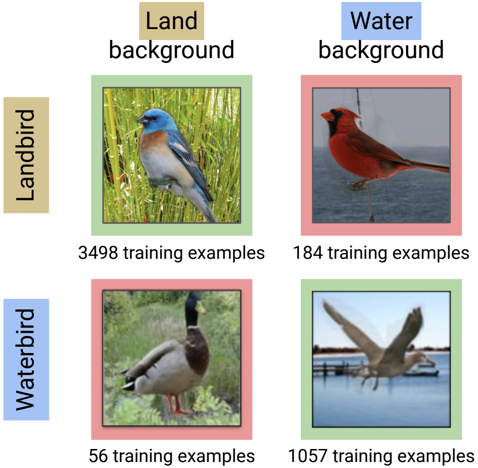
- A model trained on $\mathcal{D}_{tr}$ may simply use a to predict y, but this will result in incorrect predictions on samples with $\langle y', a \rangle$.

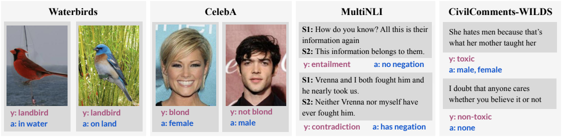


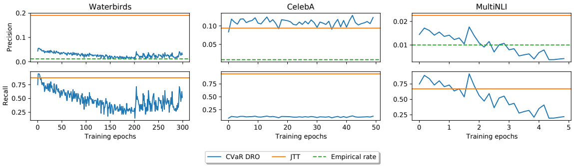


Remark 2 (Shortcut learning in Deep Learning)

Models may only learn spurious correlation (and thus sensitive to distribution shifts).







Remark 2 (Shortcut learning in Deep Learning)

Models may only learn spurious correlation (and thus sensitive to distribution shifts).

A formal definition of spurious correlation

Definition 3 (Spurious Correlation)

Let $\mathcal{D}_{tr} = \{(x_i, y_i)\}_{i=1}^{n}$ be the training set with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, where $\mathcal{X}$ denotes the set of all possible inputs, $\mathcal{Y}$ denotes the set of K classes.

- For each data point x_i with class label y_i, there is a spurious attribute $a_i \in \mathcal{A}$ of x_i, where a_i is non-predictive of y_i, and $\mathcal{A}$ denotes the set of all possible spurious attributes.

A formal definition of spurious correlation

Definition 3 (Spurious Correlation)

Let $\mathcal{D}_{tr} = \{(x_i, y_i)\}_{i=1}^{n}$ be the training set with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, where $\mathcal{X}$ denotes the set of all possible inputs, $\mathcal{Y}$ denotes the set of K classes.

- For each data point x_i with class label y_i, there is a spurious attribute $a_i \in \mathcal{A}$ of x_i, where a_i is non-predictive of y_i, and $\mathcal{A}$ denotes the set of all possible spurious attributes.
- A *spurious correlation*, denoted as $\langle y, a \rangle$, is the association between $y \in \mathcal{Y}$ and $a \in \mathcal{A}$, where y and a exist in a one-to-many mapping $\phi : \mathcal{A} \mapsto \mathcal{Y}^{K'}$ conditioned on $\mathcal{D}_{tr}$ with $1 < K' \leq K$.

A formal definition of spurious correlation

Definition 3 (Spurious Correlation)

Let $\mathcal{D}_{tr} = \{(x_i, y_i)\}_{i=1}^n$ be the training set with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, where $\mathcal{X}$ denotes the set of all possible inputs, $\mathcal{Y}$ denotes the set of K classes.

- For each data point x_i with class label y_i, there is a spurious attribute $a_i \in \mathcal{A}$ of x_i, where a_i is non-predictive of y_i, and $\mathcal{A}$ denotes the set of all possible spurious attributes.
- A *spurious correlation*, denoted as $\langle y, a \rangle$, is the association between $y \in \mathcal{Y}$ and $a \in \mathcal{A}$, where y and a exist in a one-to-many mapping $\phi : \mathcal{A} \mapsto \mathcal{Y}^{K'}$ conditioned on $\mathcal{D}_{tr}$ with $1 < K' \leq K$.
- A set of data with the spurious correlation $\langle y, a \rangle$ is annotated with the group label $g = (y, a) \in \mathcal{G}$, where $\mathcal{G} := \mathcal{Y} \times \mathcal{A}$ is the set of combinations of class labels and spurious attributes.

Wait!

Wait!

What is the definition of “classes”?

Table of Contents

Classification

We observe some data

$$\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N} \in \mathcal{X} \times \underbrace{\mathcal{Y}}_{\text{Discrete set}} \tag{3}$$

Classification

We observe some data

$$\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N} \in \mathcal{X} \times \underbrace{\mathcal{Y}}_{\text{Discrete set}} \tag{3}$$

- **Binary classification:** $y \in \{\mathcal{C}_1, \mathcal{C}_2\} \quad \Rightarrow \quad$ The $\mathcal{C}_i$ are called class labels or classes.

Classification

We observe some data

$$\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N} \in \mathcal{X} \times \underbrace{\mathcal{Y}}_{\text{Discrete set}} \tag{3}$$

- **Binary classification:** $y \in \{\mathcal{C}_1, \mathcal{C}_2\} \quad \Rightarrow \quad$ The $\mathcal{C}_i$ are called class labels or classes.
- **Multi-class classification:** $y \in \{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{K-1}\}$ for a K-class problem.

Classification

We observe some data

$$\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N} \in \mathcal{X} \times \underbrace{\mathcal{Y}}_{\text{Discrete set}} \tag{3}$$

- **Binary classification:** $y \in \{\mathcal{C}_1, \mathcal{C}_2\} \quad \Rightarrow \quad$ The $\mathcal{C}_i$ are called class labels or classes.
- **Multi-class classification:** $y \in \{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{K-1}\}$ for a K-class problem.

Remark 4

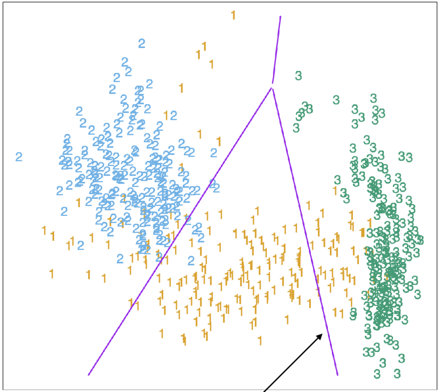
no ordering between classes.

Classifier

A classifier $f : \mathcal{X} \to \mathcal{Y}$

Classifier

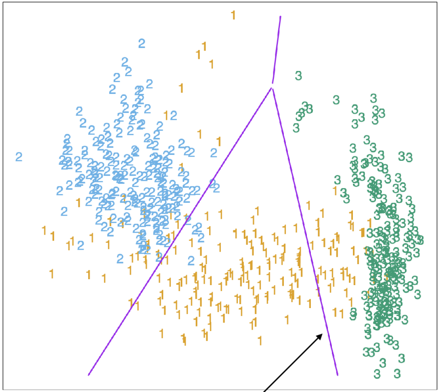
A classifier $f : \mathcal{X} \to \mathcal{Y}$ divides the input space into a collection of regions belonging to each class.



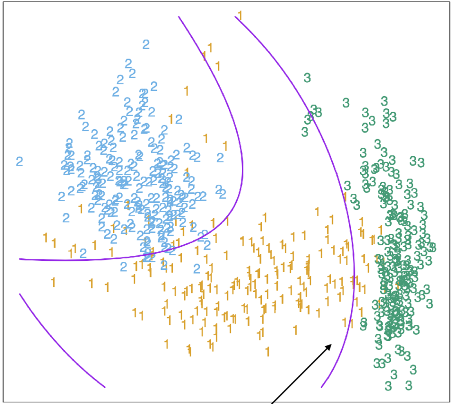
Linear Decision boundary

Classifier

A classifier $f : \mathcal{X} \to \mathcal{Y}$ divides the input space into a collection of regions belonging to each class.



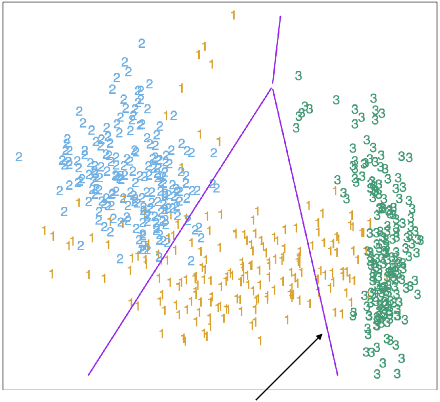
Linear Decision boundary



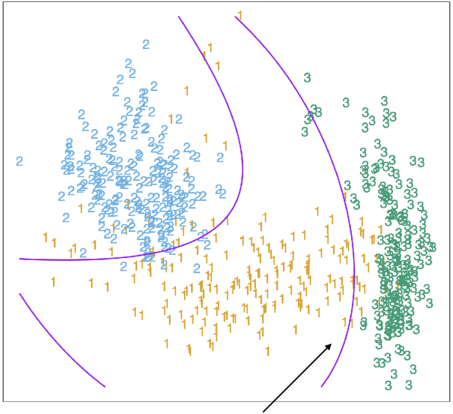
Nonlinear Decision boundary

Classifier

A classifier $f : \mathcal{X} \to \mathcal{Y}$ divides the input space into a collection of regions belonging to each class.



Linear Decision boundary



Nonlinear Decision boundary

The boundaries of these regions are called decision boundaries.

Classification: a special case of regression?

Classification is a **regression problem** with discrete labels:

$$(\mathbf{x}, y) \in \mathcal{X} \times \{0, 1\} \subset \mathcal{X} \times \mathbb{R} \tag{4}$$

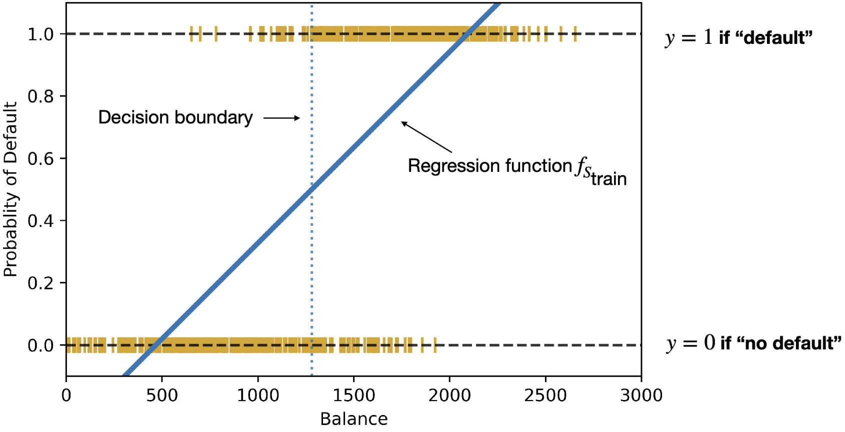
Classification: a special case of regression?

Classification is a **regression problem** with discrete labels:

$$(\mathbf{x}, y) \in \mathcal{X} \times \{0, 1\} \subset \mathcal{X} \times \mathbb{R} \tag{4}$$

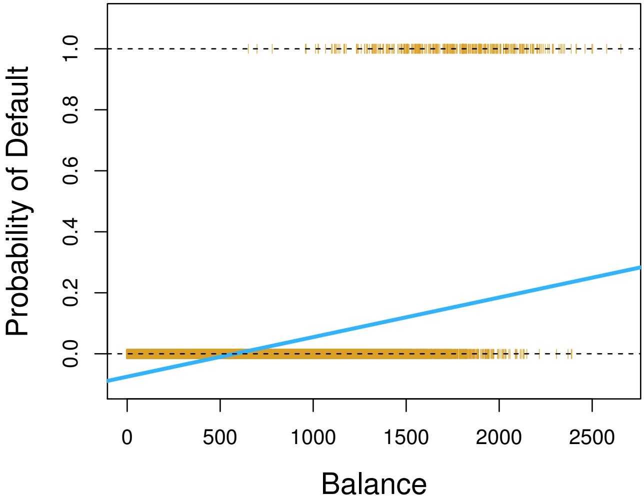
Could we use previously seen regression methods to solve it?

Is it a good idea to use regression methods for classification tasks?



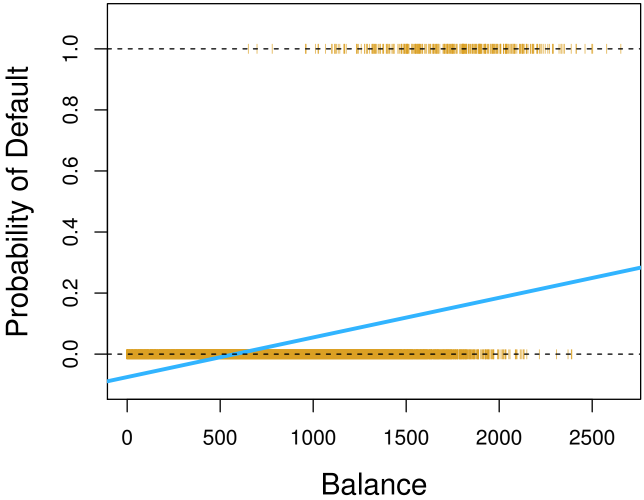
We label the output as probability for sake of interpretation

Classification is not just a special form of regression

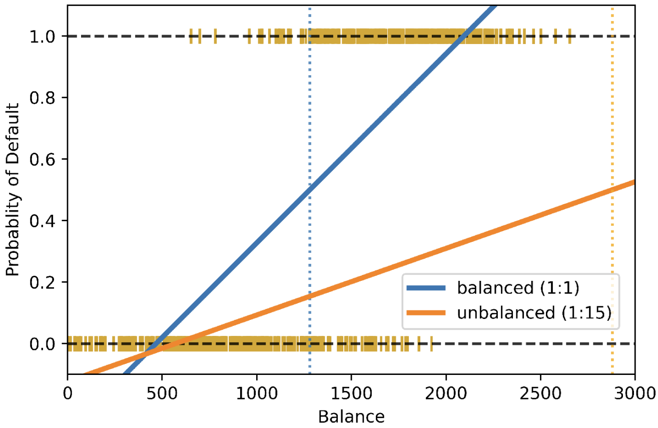


Classification is not just a special form of regression

- The predicted values are not probabilities (not in $[0, 1]$).

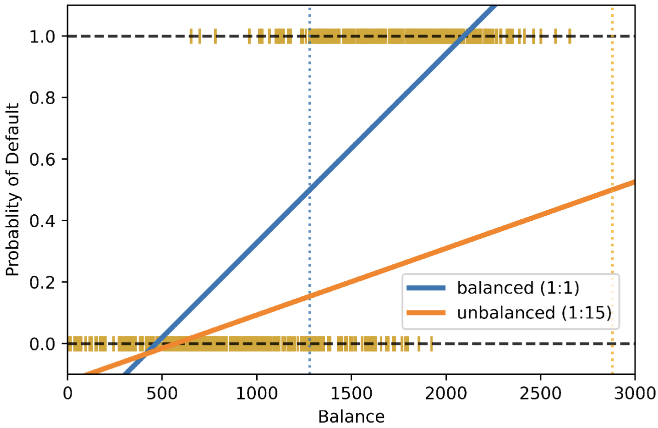


Classification is not just a special form of regression



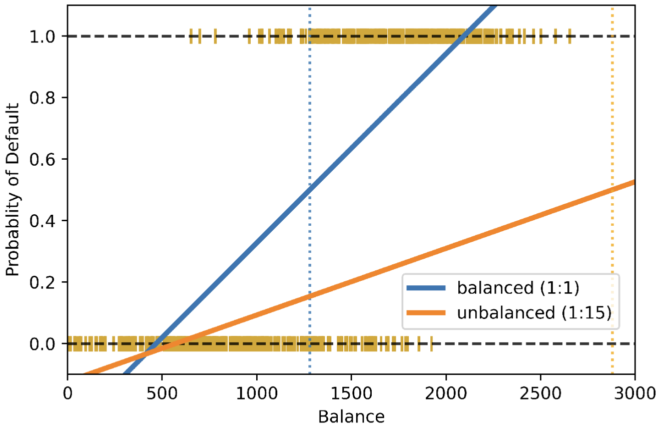
Classification is not just a special form of regression

- Sensitivity to unbalanced data.



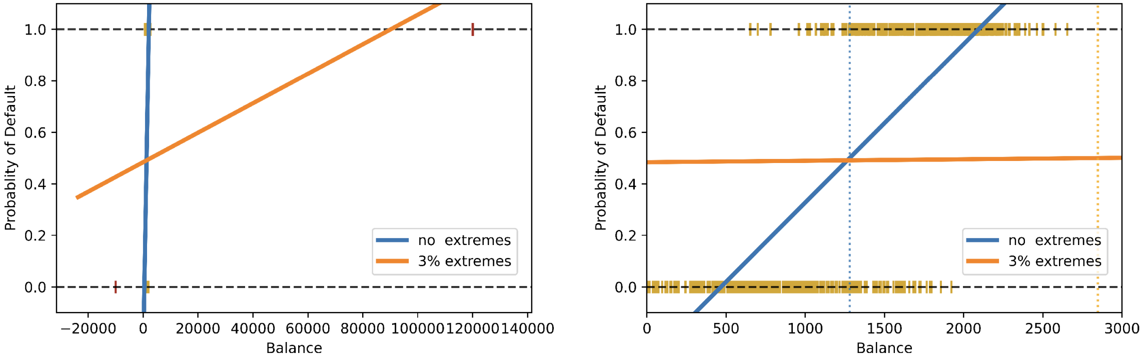
Classification is not just a special form of regression

- Sensitivity to unbalanced data.



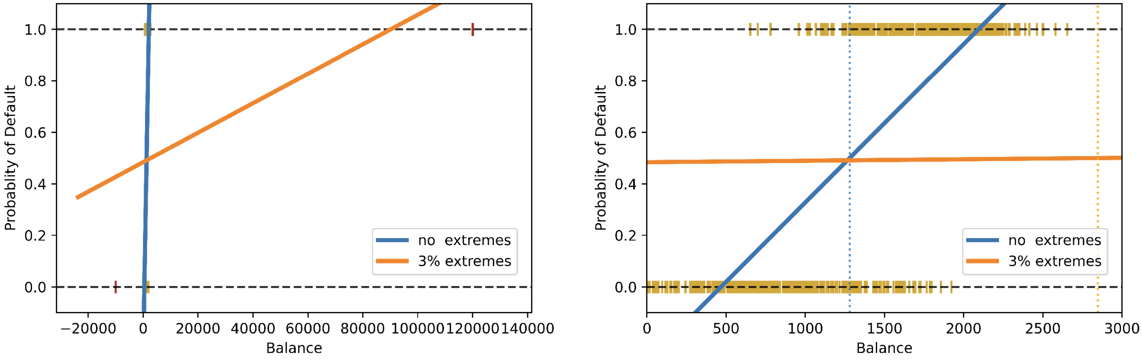
The position of the line depends crucially on how many points are in each class.

Classification is not just a special form of regression



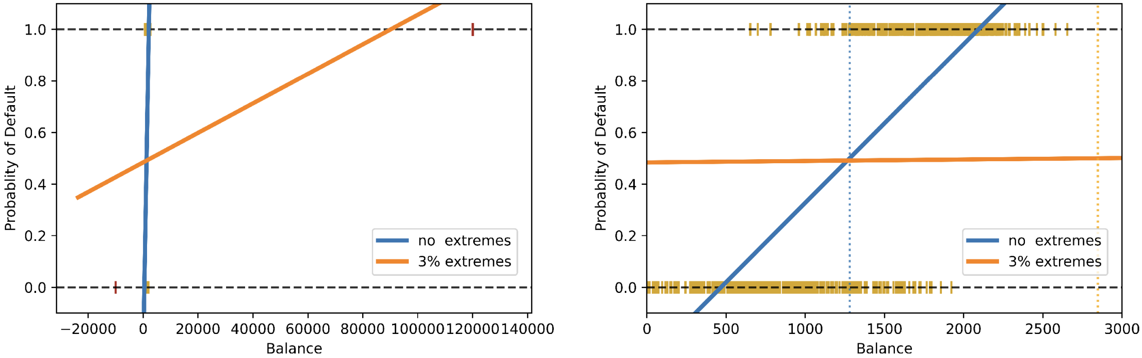
Classification is not just a special form of regression

- Sensitivity to extreme values:



Classification is not just a special form of regression

- Sensitivity to extreme values:



The position of the line depends crucially on where the points lie.

How to perform classification?

- A lot of different approaches have been developed

How to perform classification?

- A lot of different approaches have been developed
- We will not detail them today

How to perform classification?

- A lot of different approaches have been developed
- We will not detail them today
- Rather we will provide quick introductions

How to perform classification?

- A lot of different approaches have been developed
- We will not detail them today
- Rather we will provide quick introductions
- Fundamental task of classification:

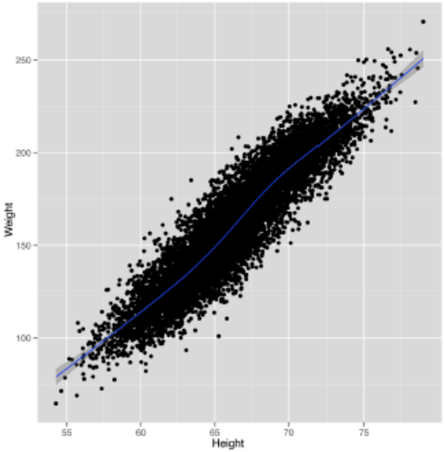
> **separate the space into various decision regions**

Table of Contents

Table of Contents

Definition

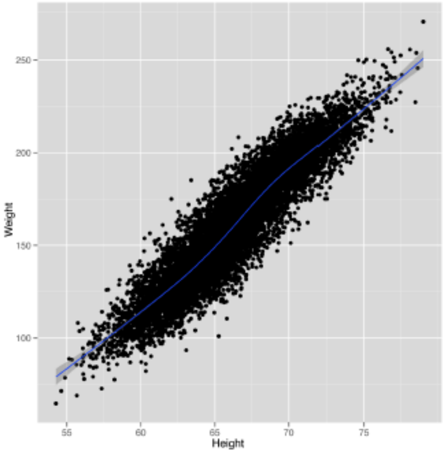
Linear regression is a model:



Definition

Linear regression is a model:

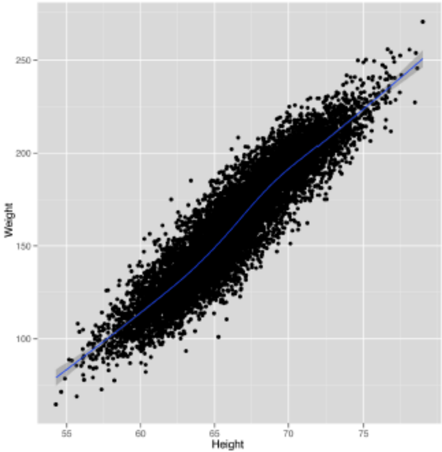
- $y_n \approx f(\mathbf{x}_n)$ for all n



Definition

Linear regression is a model:

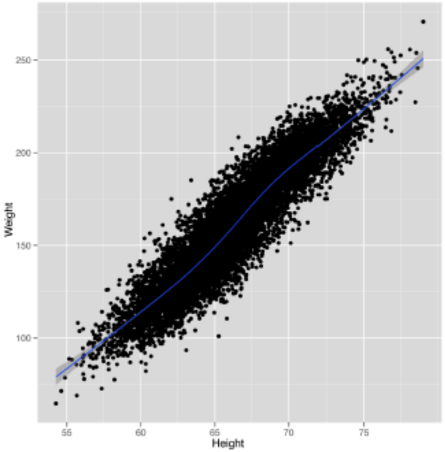
- $y_n \approx f(\mathbf{x}_n)$ for all n
- $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N} \in \mathcal{X} \times \mathcal{Y}$



Definition

Linear regression is a model:

- $y_n \approx f(\mathbf{x}_n)$ for all n
- $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N} \in \mathcal{X} \times \mathcal{Y}$
- a linear relationship is assumed for f



Detailed definition

Simple linear regression (w/ only one input dimension):

$$y_n \approx f(\mathbf{x}_n) := w_0 + w_1 x_{n1} \tag{5}$$

Detailed definition

Simple linear regression (w/ only one input dimension):

$$y_n \approx f(\mathbf{x}_n) := w_0 + w_1 x_{n1} \tag{5}$$

Here, $\mathbf{w} = (w_0, w_1)$ are the two parameters of the model. They describe f.

Multiple linear regression (multiple input dimension):

$$y_n \approx f(\mathbf{x}_n) := w_0 + w_1 x_{n1} + \ldots + w_D x_{nD} \tag{6}$$

(8)

Detailed definition

Simple linear regression (w/ only one input dimension):

$$y_n \approx f(\mathbf{x}_n) := w_0 + w_1 x_{n1} \tag{5}$$

Here, $\mathbf{w} = (w_0, w_1)$ are the two parameters of the model. They describe f.

Multiple linear regression (multiple input dimension):

$$\begin{aligned} y_n \approx f(\mathbf{x}_n) &:= w_0 + w_1 x_{n1} + \ldots + w_D x_{nD} \quad (6)\\ &= w_0 + \mathbf{x}_n^\top \begin{pmatrix} w_1 \\ \vdots \\ w_D \end{pmatrix} \quad (7)\\ & \quad (8) \end{aligned}$$

Detailed definition

Simple linear regression (w/ only one input dimension):

$$y_n \approx f(\mathbf{x}_n) := w_0 + w_1 x_{n1} \tag{5}$$

Here, $\mathbf{w} = (w_0, w_1)$ are the two parameters of the model. They describe f.

Multiple linear regression (multiple input dimension):

$$\begin{aligned} y_n \approx f(\mathbf{x}_n) &:= w_0 + w_1 x_{n1} + \ldots + w_D x_{nD} && (6)\\ &= w_0 + \mathbf{x}_n^\top \begin{pmatrix} w_1 \\ \vdots \\ w_D \end{pmatrix} && (7)\\ &=: \tilde{\mathbf{x}}_n^\top \tilde{\mathbf{w}} && (8) \end{aligned}$$

Detailed definition

Simple linear regression (w/ only one input dimension):

$$y_n \approx f(\mathbf{x}_n) := w_0 + w_1 x_{n1} \tag{5}$$

Here, $\mathbf{w} = (w_0, w_1)$ are the two parameters of the model. They describe f.

Multiple linear regression (multiple input dimension):

$$\begin{aligned} y_n \approx f(\mathbf{x}_n) &:= w_0 + w_1 x_{n1} + \ldots + w_D x_{nD} && (6) \\ &= w_0 + \mathbf{x}_n^\top \begin{pmatrix} w_1 \\ \vdots \\ w_D \end{pmatrix} && (7) \\ &=: \tilde{\mathbf{x}}_n^\top \tilde{\mathbf{w}} && (8) \end{aligned}$$

We add a tilde over the input vector & weights, to indicate containing the additional offset term (a.k.a. bias term).

Detailed definition

Simple linear regression (w/ only one input dimension):

$$y_n \approx f(\mathbf{x}_n) := w_0 + w_1 x_{n1} \tag{5}$$

Here, $\mathbf{w} = (w_0, w_1)$ are the two parameters of the model. They describe f.

Multiple linear regression (multiple input dimension):

$$\begin{aligned} y_n \approx f(\mathbf{x}_n) &:= w_0 + w_1 x_{n1} + \ldots + w_D x_{nD} && (6) \\ &= w_0 + \mathbf{x}_n^\top \begin{pmatrix} w_1 \\ \vdots \\ w_D \end{pmatrix} && (7) \\ &=: \tilde{\mathbf{x}}_n^\top \tilde{\mathbf{w}} && (8) \end{aligned}$$

We add a tilde over the input vector & weights, to indicate containing the additional offset term (a.k.a. bias term).

Goal of “learning/estimation/fitting”

Detailed definition

Simple linear regression (w/ only one input dimension):

$$y_n \approx f(\mathbf{x}_n) := w_0 + w_1 x_{n1} \tag{5}$$

Here, $\mathbf{w} = (w_0, w_1)$ are the two parameters of the model. They describe f.

Multiple linear regression (multiple input dimension):

$$\begin{aligned} y_n \approx f(\mathbf{x}_n) &:= w_0 + w_1 x_{n1} + \ldots + w_D x_{nD} && (6) \\ &= w_0 + \mathbf{x}_n^\top \begin{pmatrix} w_1 \\ \vdots \\ w_D \end{pmatrix} && (7) \\ &=: \tilde{\mathbf{x}}_n^\top \tilde{\mathbf{w}} && (8) \end{aligned}$$

We add a tilde over the input vector & weights, to indicate containing the additional offset term (a.k.a. bias term).

Goal of "learning/estimation/fitting"

Given data $\mathcal{D}$, we would like to find $\tilde{\mathbf{w}} = [w_0, w_1, \ldots, w_D]$.

Detailed definition

Simple linear regression (w/ only one input dimension):

$$y_n \approx f(\mathbf{x}_n) := w_0 + w_1 x_{n1} \tag{5}$$

Here, $\mathbf{w} = (w_0, w_1)$ are the two parameters of the model. They describe f.

Multiple linear regression (multiple input dimension):

$$\begin{aligned} y_n \approx f(\mathbf{x}_n) &:= w_0 + w_1 x_{n1} + \ldots + w_D x_{nD} && (6) \\ &= w_0 + \mathbf{x}_n^\top \begin{pmatrix} w_1 \\ \vdots \\ w_D \end{pmatrix} && (7) \\ &=: \tilde{\mathbf{x}}_n^\top \tilde{\mathbf{w}} && (8) \end{aligned}$$

We add a tilde over the input vector & weights, to indicate containing the additional offset term (a.k.a. bias term).

Goal of "learning/estimation/fitting"

Given data $\mathcal{D}$, we would like to find $\tilde{\mathbf{w}} = [w_0, w_1, \ldots, w_D]$.

We need an optimization algorithm!

Why learn about *linear* regression?

- simple

Why learn about *linear* regression?

- simple
- easy to understand

Why learn about *linear* regression?

- simple
- easy to understand
- widely used

Why learn about *linear* regression?

- simple
- easy to understand
- widely used
- easily generalized to non-linear models

Why learn about *linear* regression?

- simple
- easy to understand
- widely used
- easily generalized to non-linear models
- we can learn almost all fundamental concepts of ML with regression alone

Table of Contents

Motivation

Consider the following models.

$$\text{1-parameter model: } y_n \approx w_0$$
$$\text{2-parameter model: } y_n \approx w_0 + w_1 x_{n1}$$

Motivation

Consider the following models.

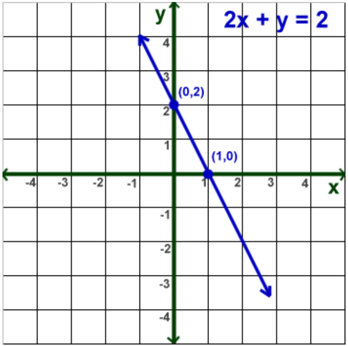
$$\text{1-parameter model: } y_n \approx w_0$$
$$\text{2-parameter model: } y_n \approx w_0 + w_1 x_{n1}$$

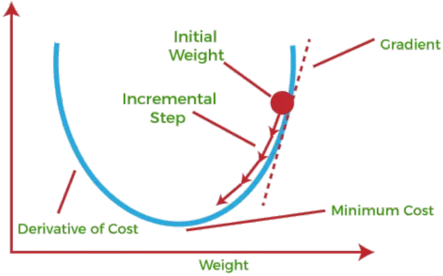
Q: How can we estimate values of $\mathbf{w}$ given the data $\mathcal{D}$?

Analytical vs. Numerical solutions

- ***Analytical solutions*** provide precise closed-form solutions but are limited to simple problems.

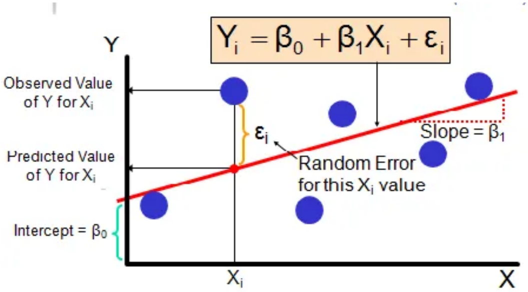
- ***Gradient Descent*** (GD) is a numerical method that approximates the optimal solution iteratively for complex problems.





https://medium.com/@msong507/gradient-descent-intuitive-overview-43e7869e1545

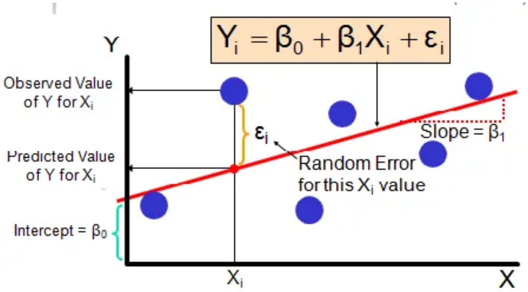
Approximate optimal solution $\mathbf{w}^\star$: cost function



- The cost function quantifies the difference between the model's predictions and the actual data.

https://medium.com/@msong507/gradient-descent-intuitive-overview-43e7869e1545

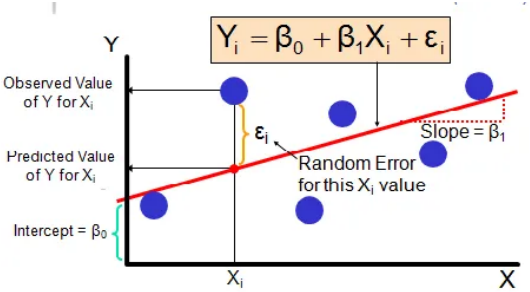
Approximate optimal solution $\mathbf{w}^\star$: cost function



- The cost function quantifies the difference between the model's predictions and the actual data.
- Two desirable properties of cost functions

https://medium.com/@msong507/gradient-descent-intuitive-overview-43e7869e1545

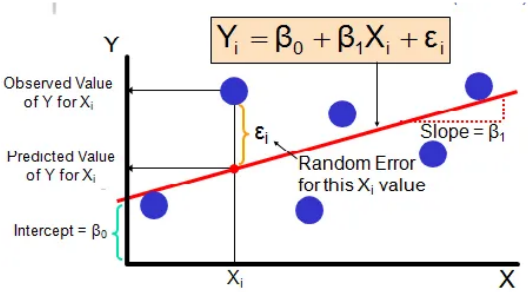
Approximate optimal solution $\mathbf{w}^\star$: cost function



- The cost function quantifies the difference between the model's predictions and the actual data.
- Two desirable properties of cost functions (let's brainstorm here).

https://medium.com/@msong507/gradient-descent-intuitive-overview-43e7869e1545

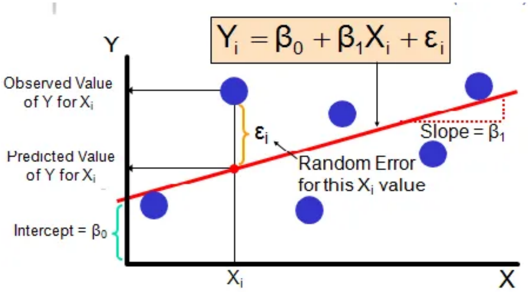
Approximate optimal solution $\mathbf{w}^\star$: cost function



- The cost function quantifies the difference between the model's predictions and the actual data.
- Two desirable properties of cost functions (let's brainstorm here).
  - the cost is symmetric around 0 (penalize positive and negative errors equally)

https://medium.com/@msong507/gradient-descent-intuitive-overview-43e7869e1545

Approximate optimal solution $\mathbf{w}^\star$: cost function



- The cost function quantifies the difference between the model's predictions and the actual data.
- Two desirable properties of cost functions (let's brainstorm here).
  - the cost is symmetric around 0 (penalize positive and negative errors equally)
  - the cost penalizes "large" mistakes and "very-large" mistakes similarly

https://medium.com/@msong507/gradient-descent-intuitive-overview-43e7869e1545

Mean Squared Error (MSE) and outliers

$$\text{MSE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}\left[y_n - f_{\mathbf{w}}(\mathbf{x}_n)\right]^2 \tag{9}$$

Mean Squared Error (MSE) and outliers

$$\text{MSE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}\left[y_n - f_{\mathbf{w}}(\mathbf{x}_n)\right]^2 \tag{9}$$

Does this cost function have both mentioned properties?

Mean Squared Error (MSE) and outliers

$$\text{MSE}(\mathbf{w}) := \frac{1}{N} \sum_{n=1}^{N} \left[y_n - f_{\mathbf{w}}(\mathbf{x}_n) \right]^2 \tag{9}$$

Does this cost function have both mentioned properties? No!

Mean Squared Error (MSE) and outliers

$$\text{MSE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}\left[y_n - f_{\mathbf{w}}(\mathbf{x}_n)\right]^2 \tag{9}$$

Does this cost function have both mentioned properties? **No!**

Definition 5 (Outliers)

Outliers are data examples that are far away from most of the other examples.

Mean Squared Error (MSE) and outliers

$$\text{MSE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}\left[y_n - f_{\mathbf{w}}(\mathbf{x}_n)\right]^2 \tag{9}$$

Does this cost function have both mentioned properties? No!

Definition 5 (Outliers)

Outliers are data examples that are far away from most of the other examples.

MSE is not a good cost function when outliers are present.

Mean Squared Error (MSE) and outliers

$$\text{MSE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}\left[y_n - f_{\mathbf{w}}(\mathbf{x}_n)\right]^2 \tag{9}$$

Does this cost function have both mentioned properties? No!

Definition 5 (Outliers)

Outliers are data examples that are far away from most of the other examples.

MSE is not a good cost function when outliers are present. Why?

Mean Squared Error (MSE) and outliers

$$\text{MSE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}\left[y_n - f_{\mathbf{w}}(\mathbf{x}_n)\right]^2 \tag{9}$$

Does this cost function have both mentioned properties? No!

Definition 5 (Outliers)

Outliers are data examples that are far away from most of the other examples.

MSE is not a good cost function when outliers are present. Why?

- **Pros:** It ensures that *trained model has no outlier predictions with huge errors*.

Mean Squared Error (MSE) and outliers

$$\text{MSE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}\left[y_n - f_{\mathbf{w}}(\mathbf{x}_n)\right]^2 \tag{9}$$

Does this cost function have both mentioned properties? No!

Definition 5 (Outliers)

Outliers are data examples that are far away from most of the other examples.

MSE is not a good cost function when outliers are present. Why?

- **Pros:** It ensures that *trained model has no outlier predictions with huge errors*.
- **Cons:** It is very sensitive to outliers.

Mean Absolute Error (MAE)

Handling outliers well is a desired *statistical* property.

Mean Absolute Error (MAE)

Handling outliers well is a desired *statistical* property.

$$\text{MAE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N} |y_n - f_{\mathbf{w}}(\mathbf{x}_n)| \tag{10}$$

Mean Absolute Error (MAE)

Handling outliers well is a desired *statistical* property.

$$\text{MAE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N} |y_n - f_{\mathbf{w}}(\mathbf{x}_n)| \tag{10}$$

+ MAE is more robust to outliers.

Mean Absolute Error (MAE)

Handling outliers well is a desired *statistical* property.

$$\text{MAE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N} |y_n - f_{\mathbf{w}}(\mathbf{x}_n)| \tag{10}$$

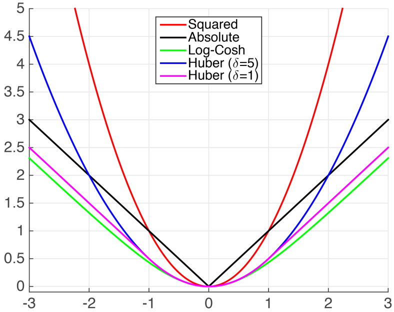
+ MAE is more robust to outliers.
– MAE is not differentiable at zero.

Discussion on different cost functions

$$\text{MSE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}\left[a := y_n - f_{\mathbf{w}}(\mathbf{x}_n)\right]^2 \tag{MSE}$$

$$\text{MAE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}|a| \tag{MAE}$$

$$\text{Huber}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}\begin{cases}\frac{1}{2}(a)^2 & \text{if } |a| \leq \delta, \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right) & \text{if } |a| > \delta,\end{cases} \tag{Huber}$$



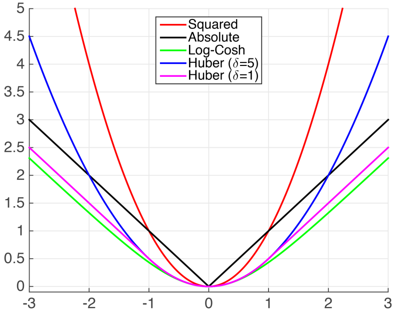
Discussion on different cost functions

$$\text{MSE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}\left[a := y_n - f_{\mathbf{w}}(\mathbf{x}_n)\right]^2 \qquad \text{(MSE)}$$

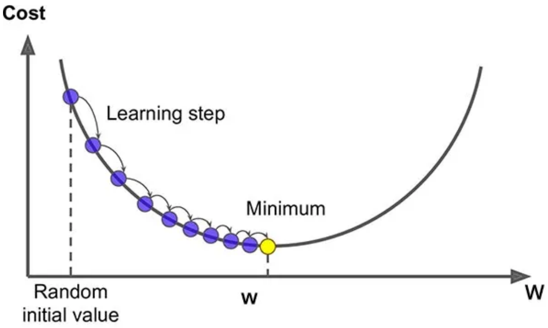
$$\text{MAE}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}|a| \qquad \text{(MAE)}$$

$$\text{Huber}(\mathbf{w}) := \frac{1}{N}\sum_{n=1}^{N}\begin{cases}\frac{1}{2}(a)^2 & \text{if } |a| \leq \delta, \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right) & \text{if } |a| > \delta,\end{cases} \qquad \text{(Huber)}$$

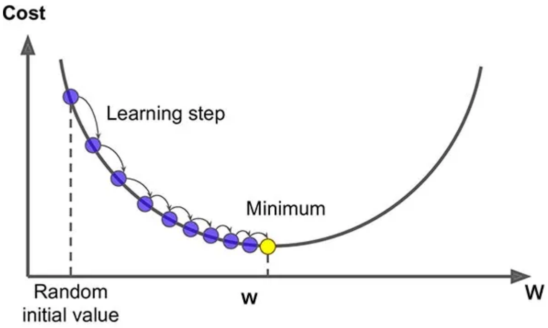
where Huber loss uses MSE logic for “small” residual values and MAE logic for “large” residual values.



Learning / Estimation / Fitting



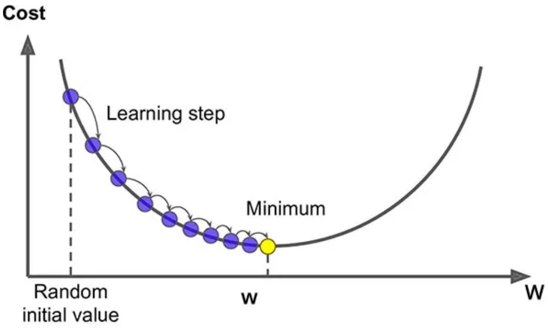
Learning / Estimation / Fitting



Definition 6 (*Learning* problem can be formulated as optimization problem)

Given a cost function $\mathcal{L}(\mathbf{w})$, we wish to find $\mathbf{w}^\star$ which minimizes the cost:

Learning / Estimation / Fitting

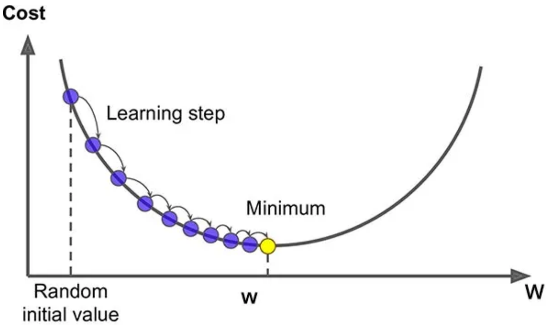


Definition 6 (*Learning* problem can be formulated as optimization problem)

Given a cost function $\mathcal{L}(\mathbf{w})$, we wish to find $\mathbf{w}^\star$ which minimizes the cost:

$$\min_{\mathbf{w}} \ \mathcal{L}(\mathbf{w}) \quad \text{subject to } \mathbf{w} \in \mathbb{R}^D \tag{11}$$

Learning / Estimation / Fitting



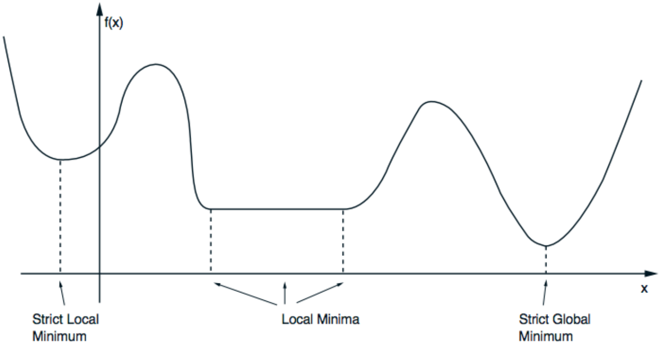
Definition 6 (*Learning* problem can be formulated as optimization problem)

Given a cost function $\mathcal{L}(\mathbf{w})$, we wish to find $\mathbf{w}^\star$ which minimizes the cost:

$$\min_{\mathbf{w}} \ \mathcal{L}(\mathbf{w}) \quad \text{subject to } \mathbf{w} \in \mathbb{R}^D \tag{11}$$

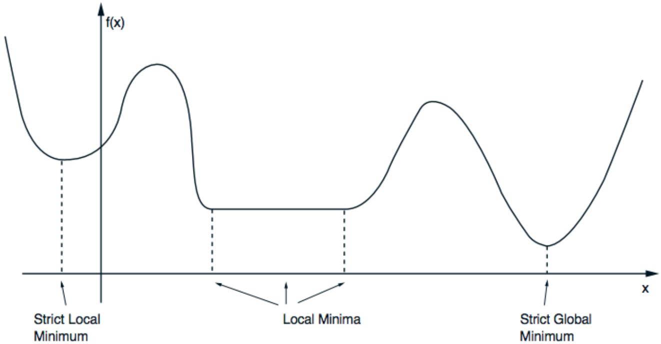
We will use an optimization algorithm like GD to solve the problem (to find a good $\mathbf{w}$).

Optimization landscapes



The above figure is taken from Bertsekas, Nonlinear programming.

Optimization landscapes

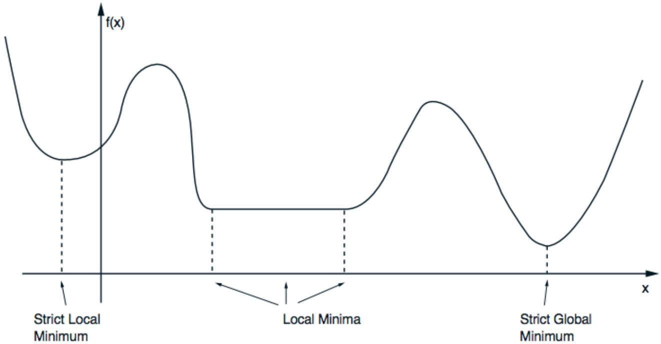


The above figure is taken from Bertsekas, Nonlinear programming.

- A vector $\mathbf{w}^\star$ is a local minimum of $\mathcal{L}$ if it is no worse than its neighbors; i.e. there exists an $\epsilon > 0$ such that,

$$\mathcal{L}(\mathbf{w}^\star) \leq \mathcal{L}(\mathbf{w}), \quad \forall \mathbf{w} \text{ with } \|\mathbf{w} - \mathbf{w}^\star\| < \epsilon$$

Optimization landscapes



The above figure is taken from Bertsekas, Nonlinear programming.

- A vector $\mathbf{w}^\star$ is a local minimum of $\mathcal{L}$ if it is no worse than its neighbors; i.e. there exists an $\epsilon > 0$ such that,

$$\mathcal{L}(\mathbf{w}^\star) \leq \mathcal{L}(\mathbf{w}), \quad \forall \mathbf{w} \text{ with } \|\mathbf{w} - \mathbf{w}^\star\| < \epsilon$$

- A vector $\mathbf{w}^\star$ is a global minimum of $\mathcal{L}$ if it is no worse than all others,

$$\mathcal{L}(\mathbf{w}^\star) \leq \mathcal{L}(\mathbf{w}), \quad \forall \mathbf{w} \in \mathbb{R}^D$$

Optimization landscapes

So what is the loss landscape of the following cost function?

Optimization landscapes

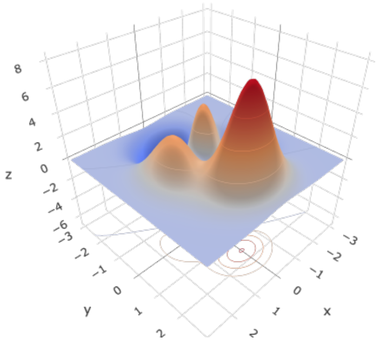
So what is the loss landscape of the following cost function?

$$f(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10(\frac{x}{5} - x^3 - y^5)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2} \tag{12}$$

Optimization landscapes

So what is the loss landscape of the following cost function?

$$f(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2} \qquad (12)$$

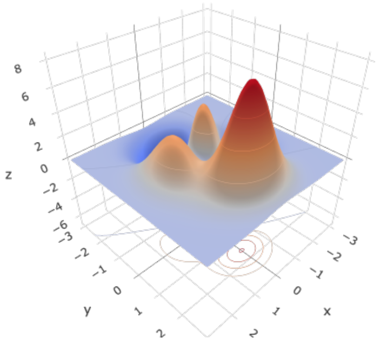


This is where gradient descent can help.

Optimization landscapes

So what is the loss landscape of the following cost function? Let's checkout this website.

$$f(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10(\frac{x}{5} - x^3 - y^5)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2} \tag{12}$$



This is where gradient descent can help.

Smooth optimization: follow the gradient

Smooth optimization: follow the gradient

Definition 7 (Gradient)

A gradient $\nabla\mathcal{L}(\mathbf{w})$ (at a point) is the slope of the ***tangent*** to the function (at that point):

Smooth optimization: follow the gradient

Definition 7 (Gradient)

A gradient $\nabla\mathcal{L}(\mathbf{w})$ (at a point) is the slope of the ***tangent*** to the function (at that point):

$$\nabla\mathcal{L}(\mathbf{w}) := \left[\frac{\partial\mathcal{L}(\mathbf{w})}{\partial w_1}, \ldots, \frac{\partial\mathcal{L}(\mathbf{w})}{\partial w_D}\right]^\top \in \mathbb{R}^D\,, \tag{13}$$

Smooth optimization: follow the gradient

Definition 7 (Gradient)

A gradient $\nabla\mathcal{L}(\mathbf{w})$ (at a point) is the slope of the ***tangent*** to the function (at that point):

$$\nabla\mathcal{L}(\mathbf{w}) := \left[\frac{\partial\mathcal{L}(\mathbf{w})}{\partial w_1}, \ldots, \frac{\partial\mathcal{L}(\mathbf{w})}{\partial w_D}\right]^\top \in \mathbb{R}^D, \tag{13}$$

where it points to the direction of the largest increase of the function.

Smooth optimization: follow the gradient

Definition 7 (Gradient)

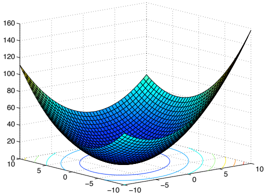
A gradient $\nabla\mathcal{L}(\mathbf{w})$ (at a point) is the slope of the ***tangent*** to the function (at that point):

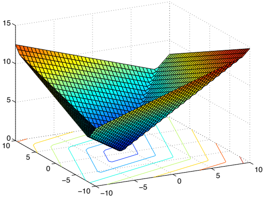
$$\nabla\mathcal{L}(\mathbf{w}) := \left[\frac{\partial\mathcal{L}(\mathbf{w})}{\partial w_1}, \ldots, \frac{\partial\mathcal{L}(\mathbf{w})}{\partial w_D}\right]^\top \in \mathbb{R}^D, \tag{13}$$

where it points to the direction of the largest increase of the function.

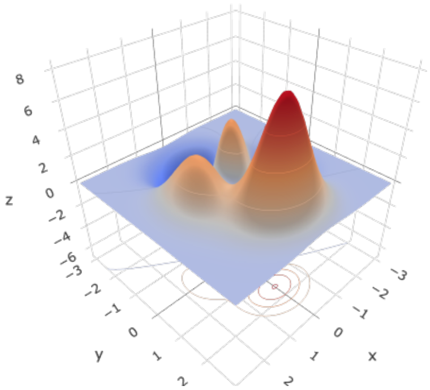
For a 2-parameter model, MSE$(\mathbf{w})$ and MAE$(\mathbf{w})$ are shown below.

(We used $\mathbf{y}_n \approx w_0 + w_1 x_{n1}$ with $\mathbf{y}^\top = [2, -1, 1.5]$ and $\mathbf{x}^\top = [-1, 1, -1]$).

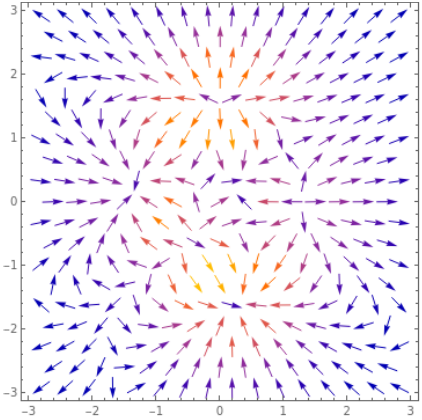




Example:



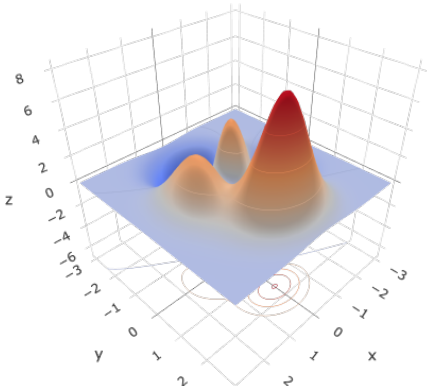
Cost function $f(x, y)$.



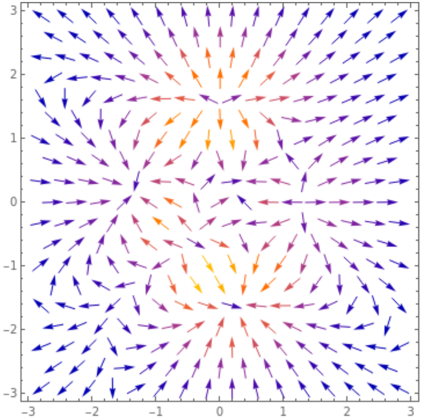
Gradient of f plotted as a vector field:

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x}(x, y) \\ \frac{\partial f}{\partial y}(x, y) \end{bmatrix}$$

Example: revisit this link for more details.



Cost function $f(x, y)$.



Gradient of f plotted as a vector field:

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f}{\partial x}(x,y) \\ \frac{\partial f}{\partial y}(x,y) \end{bmatrix}$$

Gradient Descent (GD)

Definition 8 (Gradient Descent)

To minimize the cost function, we iteratively take a step in the (opposite) direction of the gradient

Gradient Descent (GD)

Definition 8 (Gradient Descent)

To minimize the cost function, we iteratively take a step in the (opposite) direction of the gradient

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \nabla \mathcal{L}(\mathbf{w}^{(t)}) \tag{14}$$

Gradient Descent (GD)

Definition 8 (Gradient Descent)

To minimize the cost function, we iteratively take a step in the (opposite) direction of the gradient

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \nabla \mathcal{L}(\mathbf{w}^{(t)}) \tag{14}$$

where $\gamma > 0$ is the step-size (or learning rate). Then repeat with the next t.

Table of Contents

Gradient Descent for linear regression with MSE

Considering a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ and learnable weights $\mathbf{w} \in \mathbb{R}^D$ for $f_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w}$.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N, \qquad \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix} \in \mathbb{R}^{N \times D} \tag{15}$$

Gradient Descent for linear regression with MSE

Considering a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ and learnable weights $\mathbf{w} \in \mathbb{R}^D$ for $f_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w}$.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N, \qquad \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix} \in \mathbb{R}^{N \times D} \tag{15}$$

We define the error vector $\mathbf{e}$:

$$\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{w} = \begin{pmatrix} e_1 \\ \vdots \\ e_n \\ \vdots \\ e_N \end{pmatrix} \in \mathbb{R}^N, \tag{16}$$

where $e_n := y_n - \mathbf{x}_n^\top \mathbf{w}$.

Gradient Descent for linear regression with MSE

Considering a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ and learnable weights $\mathbf{w} \in \mathbb{R}^D$ for $f_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w}$.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N, \qquad \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix} \in \mathbb{R}^{N \times D} \tag{15}$$

We define the error vector $\mathbf{e}$:

$$\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{w} = \begin{pmatrix} e_1 \\ \vdots \\ e_n \\ \vdots \\ e_N \end{pmatrix} \in \mathbb{R}^N, \tag{16}$$

where $e_n := y_n - \mathbf{x}_n^\top \mathbf{w}$. The MSE is defined as $\mathcal{L}(\mathbf{w}) := \frac{1}{2N} \sum_{n=1}^{N} \left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2$,

Gradient Descent for linear regression with MSE

Considering a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ and learnable weights $\mathbf{w} \in \mathbb{R}^D$ for $f_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w}$.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N, \qquad \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix} \in \mathbb{R}^{N \times D} \tag{15}$$

We define the error vector $\mathbf{e}$:

$$\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{w} = \begin{pmatrix} e_1 \\ \vdots \\ e_n \\ \vdots \\ e_N \end{pmatrix} \in \mathbb{R}^N, \tag{16}$$

where $e_n := y_n - \mathbf{x}_n^\top \mathbf{w}$. The MSE is defined as $\mathcal{L}(\mathbf{w}) := \frac{1}{2N} \sum_{n=1}^{N} \left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 = \frac{1}{2N} \mathbf{e}^\top \mathbf{e}$, and then the gradient is given by

$$\nabla \mathcal{L}(\mathbf{w}) = -\frac{1}{N} \mathbf{X}^\top \mathbf{e}$$

.

Stochastic Gradient Descent (SGD)

- **Sum Objectives:** In machine learning, most cost functions are formulated as a sum over the training examples, that is

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(\mathbf{w}) ,$$

Stochastic Gradient Descent (SGD)

- **Sum Objectives:** In machine learning, most cost functions are formulated as a sum over the training examples, that is

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(\mathbf{w}) \ ,$$

where $\mathcal{L}_n$ is the cost contributed by the n-th training example.

Stochastic Gradient Descent (SGD)

- **Sum Objectives:** In machine learning, most cost functions are formulated as a sum over the training examples, that is

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(\mathbf{w}) \ ,$$

where $\mathcal{L}_n$ is the cost contributed by the n-th training example.

- **The SGD Algorithm:** The Stochastic Gradient Descent (SGD) algorithm is given by the following update rule, at step t:

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma\,\nabla\mathcal{L}_n(\mathbf{w}^{(t)}) \, .$$

Stochastic Gradient Descent (SGD)

- **Sum Objectives:** In machine learning, most cost functions are formulated as a sum over the training examples, that is
$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}(\mathbf{w}) \ ,$$
where $\mathcal{L}_n$ is the cost contributed by the n-th training example.

- **The SGD Algorithm:** The Stochastic Gradient Descent (SGD) algorithm is given by the following update rule, at step t:
$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \, \nabla \mathcal{L}_n(\mathbf{w}^{(t)}) \, .$$

- **Theoretical Motivation:** In expectation over the random choice of n, we have
$$\mathbb{E}[\nabla \mathcal{L}_n(\mathbf{w})] = \nabla \mathcal{L}(\mathbf{w})$$
which is the true gradient direction.

- **Mini-batch SGD:** There is an intermediate version, using the update direction being

$$\mathbf{g} := \frac{1}{|B|} \sum_{n \in B} \nabla \mathcal{L}_n(\mathbf{w}^{(t)})$$

- **Mini-batch SGD:** There is an intermediate version, using the update direction being

$$\mathbf{g} := \frac{1}{|B|} \sum_{n \in B} \nabla \mathcal{L}_n(\mathbf{w}^{(t)})$$

again with

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \, \mathbf{g} \, .$$

- **Mini-batch SGD:** There is an intermediate version, using the update direction being

$$\mathbf{g} := \frac{1}{|B|} \sum_{n \in B} \nabla \mathcal{L}_n(\mathbf{w}^{(t)})$$

  again with

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \, \mathbf{g} \, .$$

- We have randomly chosen a subset $B \subseteq [N]$ of the training examples.

- **Mini-batch SGD:** There is an intermediate version, using the update direction being

$$\mathbf{g} := \frac{1}{|B|} \sum_{n \in B} \nabla \mathcal{L}_n(\mathbf{w}^{(t)})$$

  again with

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \, \mathbf{g} \, .$$

- We have randomly chosen a subset $B \subseteq [N]$ of the training examples. For each of these selected examples n, we compute the respective gradient $\nabla \mathcal{L}_n$, at the same current point $\mathbf{w}^{(t)}$.

- **Mini-batch SGD:** There is an intermediate version, using the update direction being

$$\mathbf{g} := \frac{1}{|B|} \sum_{n \in B} \nabla \mathcal{L}_n(\mathbf{w}^{(t)})$$

again with

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \, \mathbf{g} \, .$$

- We have randomly chosen a subset $B \subseteq [N]$ of the training examples. For each of these selected examples n, we compute the respective gradient $\nabla \mathcal{L}_n$, at the same current point $\mathbf{w}^{(t)}$.
- The computation of $\mathbf{g}$ can be parallelized easily. This is how current deep-learning applications utilize GPUs (by running over $|B|$ threads in parallel).

- **Mini-batch SGD:** There is an intermediate version, using the update direction being

$$\mathbf{g} := \frac{1}{|B|} \sum_{n \in B} \nabla \mathcal{L}_n(\mathbf{w}^{(t)})$$

again with

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \, \mathbf{g} \, .$$

- We have randomly chosen a subset $B \subseteq [N]$ of the training examples. For each of these selected examples n, we compute the respective gradient $\nabla \mathcal{L}_n$, at the same current point $\mathbf{w}^{(t)}$.
- The computation of $\mathbf{g}$ can be parallelized easily. This is how current deep-learning applications utilize GPUs (by running over $|B|$ threads in parallel).
- Note that in the extreme case $B := [N]$, we obtain (batch) gradient descent, i.e. $\mathbf{g} = \nabla \mathcal{L}$.

An alternative view of SGD

Finite-sum empirical risk minimization problem:

$$f(\mathbf{x}^\star) = \min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} \left(f_i(\mathbf{x}) := \boxed{F(\mathbf{x}, \boldsymbol{\xi}_i)} \right) \right\}$$

An alternative view of SGD

Finite-sum empirical risk minimization problem:

$$f(\mathbf{x}^\star) = \min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^N \left(f_i(\mathbf{x}) := \boxed{F(\mathbf{x}, \boldsymbol{\xi}_i)} \right) \right\}$$

- The loss function of i-th data $\xi_i := (\mathbf{d}_i, y_i)$

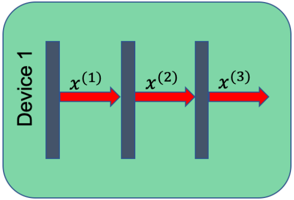
An alternative view of SGD

Finite-sum empirical risk minimization problem:

$$f(\mathbf{x}^\star) = \min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} \left(f_i(\mathbf{x}) := F(\mathbf{x}, \boldsymbol{\xi}_i) \right) \right\}$$

- The loss function of i-th data $\boldsymbol{\xi}_i := (\mathbf{d}_i, y_i)$
- Baseline method: Stochastic Gradient Descent (SGD)

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \nabla F(\mathbf{x}^{(t)}, \boldsymbol{\xi}_i)$$



An alternative view of SGD

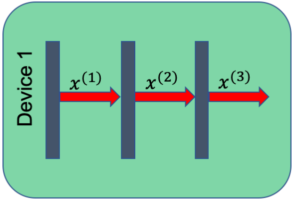
Finite-sum empirical risk minimization problem:

$$f(\mathbf{x}^\star) = \min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} \left(f_i(\mathbf{x}) := \boxed{F(\mathbf{x}, \boldsymbol{\xi}_i)} \right) \right\}$$

- The loss function of i-th data $\boldsymbol{\xi}_i := (\mathbf{d}_i, y_i)$
- Baseline method: Stochastic Gradient Descent (SGD)

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \boxed{\eta}\ \boxed{\nabla F(\mathbf{x}^{(t)}, \boldsymbol{\xi}_i)}$$

  - η is step-size/learning rate



An alternative view of SGD

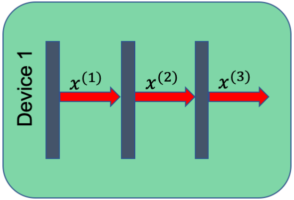
Finite-sum empirical risk minimization problem:

$$f(\mathbf{x}^\star) = \min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} \left(f_i(\mathbf{x}) := F(\mathbf{x}, \boldsymbol{\xi}_i) \right) \right\}$$

- The loss function of i-th data $\boldsymbol{\xi}_i := (\mathbf{d}_i, y_i)$
- Baseline method: Stochastic Gradient Descent (SGD)

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \, \nabla F(\mathbf{x}^{(t)}, \boldsymbol{\xi}_i)$$

  - η is step-size/learning rate
  - sampled i.i.d. $i \in \{1, \ldots, N\}$



An alternative view of SGD

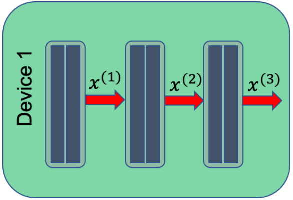
Finite-sum empirical risk minimization problem:

$$f(\mathbf{x}^\star) = \min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}) := \tfrac{1}{N}\sum_{i=1}^{N} \left(f_i(\mathbf{x}) := F(\mathbf{x}, \boldsymbol{\xi}_i) \right) \right\}$$

- The loss function of i-th data $\boldsymbol{\xi}_i := (\mathbf{d}_i, y_i)$
- Baseline method: Stochastic Gradient Descent (SGD)

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \frac{1}{B}\sum_{i\in\mathcal{B}} \eta \nabla f_i(\mathbf{x})$$

(Using *mini-batching*)



An alternative view of SGD

Finite-sum empirical risk minimization problem:

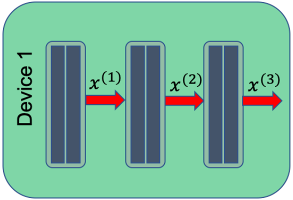
$$f(\mathbf{x}^\star) = \min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} \left(f_i(\mathbf{x}) := F(\mathbf{x}, \boldsymbol{\xi}_i) \right) \right\}$$

- The loss function of i-th data $\boldsymbol{\xi}_i := (\mathbf{d}_i, y_i)$
- Baseline method: Stochastic Gradient Descent (SGD)

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \frac{1}{B}\sum_{i\in\mathcal{B}} \eta \nabla f_i(\mathbf{x})$$

(Using *mini-batching*)

- $\mathcal{B} \in \{1, \ldots, N\}$ with $|\mathcal{B}| = B$.



Alternative view: Stochastic reformulation of finite-sum problems (SGD with arbitrary sampling)

Random sampling vector $\mathbf{v} = (v_1, \ldots, v_N) \sim \mathcal{D}$ with $\mathbb{E}[v_i] = 1 \qquad$ for $i = 1, \ldots, N$.

Alternative view: Stochastic reformulation of finite-sum problems (SGD with arbitrary sampling)

Random sampling vector $\mathbf{v} = (v_1, \ldots, v_N) \sim \mathcal{D}$ with $\mathbb{E}\left[v_i\right] = 1$ for $i = 1, \ldots, N$.

$f(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^{N} f_i(\mathbf{x})$ (Stochastic Reformulation)

Alternative view: Stochastic reformulation of finite-sum problems (SGD with arbitrary sampling)

Random sampling vector $\mathbf{v} = (v_1, \ldots, v_N) \sim \mathcal{D}$ with $\mathbb{E}[v_i] = 1$ for $i = 1, \ldots, N$.

$$f(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}[v_i] f_i(\mathbf{x}) \qquad \text{(Stochastic Reformulation)}$$

Alternative view: Stochastic reformulation of finite-sum problems (SGD with arbitrary sampling)

Random sampling vector $\mathbf{v} = (v_1, \ldots, v_N) \sim \mathcal{D}$ with $\mathbb{E}\left[v_i\right] = 1 \qquad$ for $i = 1, \ldots, N$.

$$f(\mathbf{x}) := \tfrac{1}{N}\textstyle\sum_{i=1}^{N} f_i(\mathbf{x}) = \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \mathbb{E}\left[v_i\right] f_i(\mathbf{x}) = \mathbb{E}\left[\underbrace{\frac{1}{N}\sum_{i=1}^{N} v_i f_i(\mathbf{x})}_{=: f_v(\mathbf{x})}\right] \qquad \text{(Stochastic Reformulation)}$$

Alternative view: Stochastic reformulation of finite-sum problems (SGD with arbitrary sampling)

Random sampling vector $\mathbf{v} = (v_1, \ldots, v_N) \sim \mathcal{D}$ with $\mathbb{E}\left[v_i\right] = 1$ for $i = 1, \ldots, N$.

$$f(\mathbf{x}) := \tfrac{1}{N}\textstyle\sum_{i=1}^{N} f_i(\mathbf{x}) = \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \mathbb{E}\left[v_i\right] f_i(\mathbf{x}) = \mathbb{E}\left[\underbrace{\frac{1}{N}\sum_{i=1}^{N} v_i f_i(\mathbf{x})}_{=: f_v(\mathbf{x})}\right] \qquad \text{(Stochastic Reformulation)}$$

Original Finite-sum problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x})$$

$\longleftrightarrow$

Stochastic Reformulation

$$\min_{\mathbf{x}\in\mathbb{R}^d} \mathbb{E}\left[f_v(\mathbf{x})\right]$$

Minimizing the expectation of **random linear combinations** of original function

Alternative view: Stochastic reformulation of finite-sum problems (SGD with arbitrary sampling)

Random sampling vector $\mathbf{v} = (v_1, \ldots, v_N) \sim \mathcal{D}$ with $\mathbb{E}[v_i] = 1$ for $i = 1, \ldots, N$.

$$f(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^N f_i(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^N \mathbb{E}[v_i] f_i(\mathbf{x}) = \mathbb{E}\left[\underbrace{\frac{1}{N}\sum_{i=1}^N v_i f_i(\mathbf{x})}_{=: f_v(\mathbf{x})}\right] \qquad \text{(Stochastic Reformulation)}$$

Original Finite-sum problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^N f_i(\mathbf{x})$$

$\longleftrightarrow$

Stochastic Reformulation

$$\min_{\mathbf{x}\in\mathbb{R}^d} \mathbb{E}[f_v(\mathbf{x})]$$

Minimizing the expectation of **random linear combinations** of original function

$\downarrow$

Sample $v_i^{(t)} \sim \mathcal{D}$

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \eta \nabla f_{v_i^{(t)}}(\mathbf{x}^{(t)})$$

The distribution $\mathcal{D}$ encodes any form of mini-batching/non-uniform sampling. Our analysis is done for any distribution $\mathcal{D}$.

Examples of arbitrary sampling: uniform single element

Random set $\mathcal{S} \subset \{1, \ldots, N\}$, $|\mathcal{S}| = 1$,
$\Pr[i \in \mathcal{S}] = \frac{1}{N}$, for $i = 1, \ldots, N$

$$\begin{aligned} \mathbb{E}[v_i] &= 1 \\ v_i = &\begin{cases} N & i \in \mathcal{S} \\ 0 & i \notin \mathcal{S} \end{cases} \end{aligned} \tag{17}$$

$$\nabla f_v(\mathbf{x}) = \nabla f_i(\mathbf{x}),$$

where $\mathbb{E}[\nabla f_v(\mathbf{x})] = \nabla f(\mathbf{x})$

Single element SGD

Sample $v^{(t)} \sim \mathcal{D}$

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \nabla f_{v^{(t)}}(\mathbf{x}^{(t)})$$

Examples of arbitrary sampling: uniform mini-batching

Random set $\mathcal{S} \subset \{1, \ldots, N\}$, $|\mathcal{S}| = B$,
$\Pr[i \in \mathcal{S}] = \frac{B}{N}$, for $i = 1, \ldots, N$

$$\mathbb{E}[v_i] = 1$$

$$v_i = \begin{cases} \frac{N}{B} & i \in \mathcal{S} \\ 0 & i \notin \mathcal{S} \end{cases} \quad (18)$$

$$\nabla f_v(\mathbf{x}) = \frac{1}{B} \sum_{i \in \mathcal{S}} \nabla f_i(\mathbf{x}),$$

where $\mathbb{E}[\nabla f_v(\mathbf{x})] = \nabla f(\mathbf{x})$

Mini-batch SGD w/o replacement

Sample $v^{(t)} \sim \mathcal{D}$

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \nabla f_{v^{(t)}}(\mathbf{x}^{(t)})$$

Examples of arbitrary sampling: non-uniform mini-batching

Random set $\mathcal{S} \subset \{1, \ldots, N\}$, $|\mathcal{S}| = B$,
$\Pr[i \in \mathcal{S}] = p_i$, for $i = 1, \ldots, N$

$$\mathbb{E}[v_i] = 1$$

$$v_i = \begin{cases} \frac{1}{p_i} & i \in \mathcal{S} \\ 0 & i \notin \mathcal{S} \end{cases} \tag{19}$$

$$\nabla f_v(\mathbf{x}) = \frac{n}{p_i} \sum_{i \in \mathcal{S}} \nabla f_i(\mathbf{x}),$$

where $\mathbb{E}[\nabla f_v(\mathbf{x})] = \nabla f(\mathbf{x})$

Arbitrary sampling SGD

Sample $v^{(t)} \sim \mathcal{D}$

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \nabla f_{v^{(t)}}(\mathbf{x}^{(t)})$$

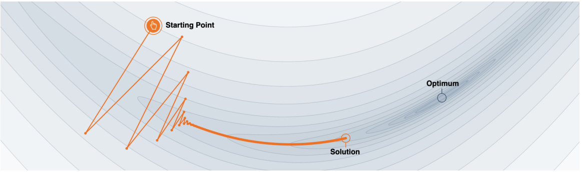
Includes all forms of SGD (and GD)

$$\min_{\mathbf{x} \in \mathbb{R}^d} \mathbb{E}\left[f_v(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^{N} v_i f_i(\mathbf{x})\right]$$

Table of Contents

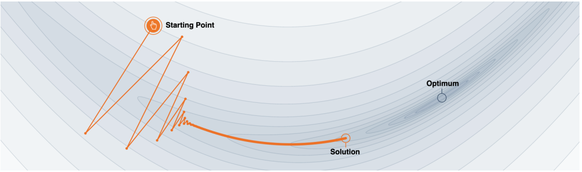
Motivation

Recall that *GD for Linear Regression with MSE loss* is an iterative optimization algorithm, computationally flexibility for handling large datasets.



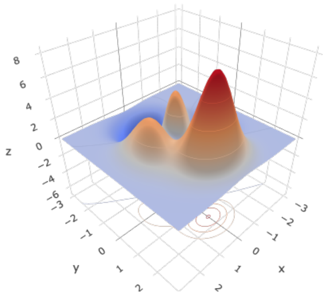
Motivation

Recall that *GD for Linear Regression with MSE loss* is an <u>iterative</u> optimization algorithm, computationally flexibility for handling large datasets.

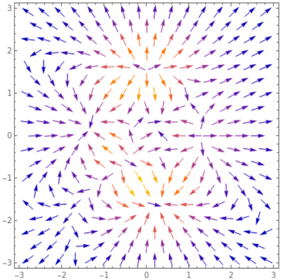


Can we compute the optimum of the cost function analytically?

- Linear regression using an MSE cost function is one such case.

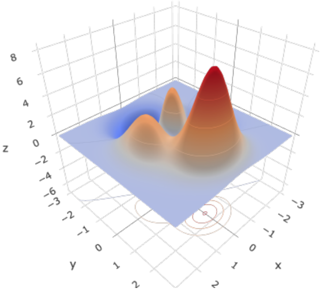


Cost function $f(x, y)$.

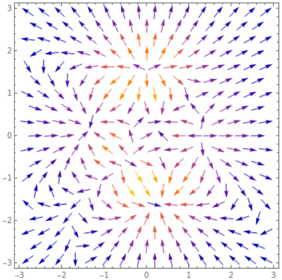


Gradient of f plotted as a vector field.

- Linear regression using an MSE cost function is one such case.

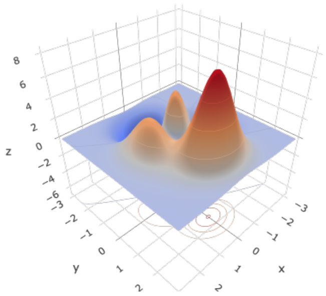


Cost function $f(x, y)$.

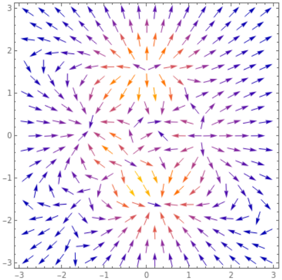


Gradient of f plotted as a vector field.

- Linear regression using an MSE cost function is one such case.
- Its solution can be obtained explicitly, **by solving a linear system of equations**.

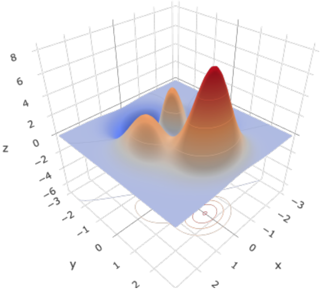


Cost function $f(x, y)$.

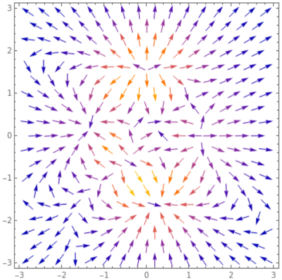

Gradient of f plotted as a vector field.

- Linear regression using an MSE cost function is one such case.
- Its solution can be obtained explicitly, **by solving a linear system of equations**.

$\Rightarrow$ These equations are sometimes called the normal equations.



Cost function $f(x, y)$.



Gradient of f plotted as a vector field.

- Linear regression using an MSE cost function is one such case.
- Its solution can be obtained explicitly, **by solving a linear system of equations**.

$\Rightarrow$ These equations are sometimes called the normal equations.

$\Rightarrow$ Solving the normal equations is called the least squares.

Recall that the cost function for linear regression with MSE is given by

Recall that the cost function for linear regression with MSE is given by

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 = \frac{1}{2N}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}), \tag{20}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N, \qquad \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix} \in \mathbb{R}^{N\times D}. \tag{21}$$

Recall that the cost function for linear regression with MSE is given by

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 = \frac{1}{2N}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}), \tag{20}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N, \qquad \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix} \in \mathbb{R}^{N\times D}. \tag{21}$$

What is the meaning of finding the analytical solution?

Steps to form normal equations

Steps to form normal equations

Let's start with some tools:

Steps to form normal equations

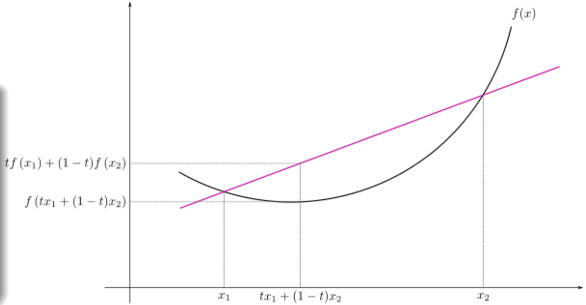
Let's start with some tools:

Definition 9 (Convexity)

Steps to form normal equations

Let's start with some tools:

Definition 9 (Convexity)



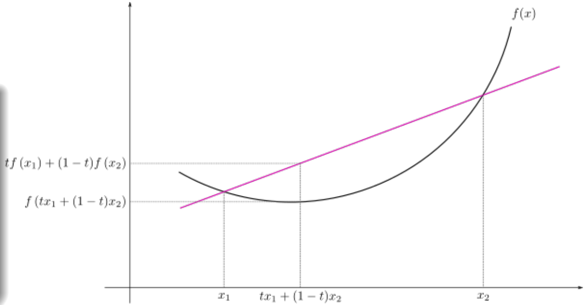
Steps to form normal equations

Let's start with some tools:

Definition 9 (Convexity)

A function $h(\mathbf{u})$ with $\mathbf{u} \in \mathbb{R}^D$ is convex, if for any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^D$ and for any $0 \leq \lambda \leq 1$, we have:

$$h\left(\lambda \mathbf{u} + (1-\lambda)\mathbf{v}\right) \leq \lambda h(\mathbf{u}) + (1-\lambda)h(\mathbf{v}) \quad (22)$$



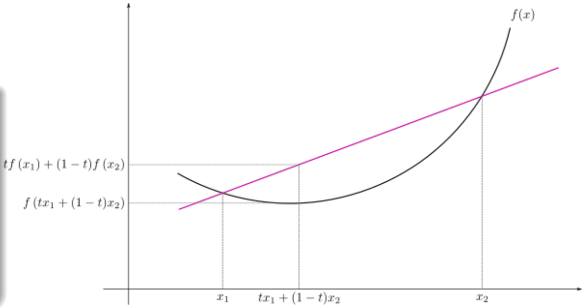
Steps to form normal equations

Let's start with some tools:

Definition 9 (Convexity)

A function $h(\mathbf{u})$ with $\mathbf{u} \in \mathbb{R}^D$ is convex, if for any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^D$ and for any $0 \leq \lambda \leq 1$, we have:

$$h\left(\lambda \mathbf{u} + (1-\lambda)\mathbf{v}\right) \leq \lambda h(\mathbf{u}) + (1-\lambda) h(\mathbf{v}) \quad (22)$$



To derive the normal equations,

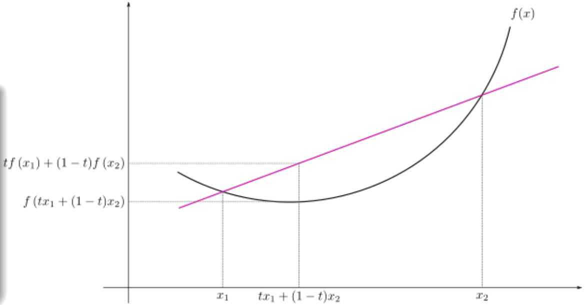
Steps to form normal equations

Let's start with some tools:

Definition 9 (Convexity)

A function $h(\mathbf{u})$ with $\mathbf{u} \in \mathbb{R}^D$ is convex, if for any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^D$ and for any $0 \leq \lambda \leq 1$, we have:

$$h\left(\lambda \mathbf{u} + (1-\lambda)\mathbf{v}\right) \leq \lambda h(\mathbf{u}) + (1-\lambda) h(\mathbf{v}) \quad (22)$$



To derive the normal equations,

1. we first show that the problem is convex.

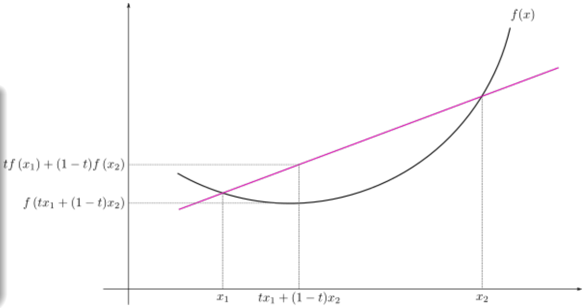
Steps to form normal equations

Let's start with some tools:

Definition 9 (Convexity)

A function $h(\mathbf{u})$ with $\mathbf{u} \in \mathbb{R}^D$ is convex, if for any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^D$ and for any $0 \le \lambda \le 1$, we have:

$$h\left(\lambda \mathbf{u} + (1-\lambda)\mathbf{v}\right) \le \lambda h(\mathbf{u}) + (1-\lambda) h(\mathbf{v}) \quad (22)$$



To derive the normal equations,

1. we first show that the problem is convex.
2. we then use the optimality conditions for convex functions, i.e.,

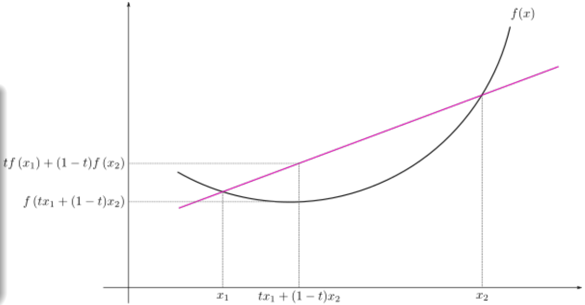
Steps to form normal equations

Let's start with some tools:

Definition 9 (Convexity)

A function $h(\mathbf{u})$ with $\mathbf{u} \in \mathbb{R}^D$ is convex, if for any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^D$ and for any $0 \leq \lambda \leq 1$, we have:

$$h(\lambda \mathbf{u} + (1-\lambda)\mathbf{v}) \leq \lambda h(\mathbf{u}) + (1-\lambda) h(\mathbf{v}) \qquad (22)$$



To derive the normal equations,

1. we first show that the problem is convex.
2. we then use the optimality conditions for convex functions, i.e.,

$$\nabla \mathcal{L}(\mathbf{w}^\star) = \mathbf{0}\,, \qquad (23)$$

where $\mathbf{w}^\star$ corresponds to the parameter at the optimum point.

Derivation (step 1): the MSE is *convex* in the $\mathbf{w}$

There are several ways of proving this:

Derivation (step 1): the MSE is *convex* in the $\mathbf{w}$

There are several ways of proving this:

1. **Way 1.** Recall the definition of $\mathcal{L}$

$$\mathcal{L} := \frac{1}{2N} \sum_{n=1}^{N} \left(\mathcal{L}_n := y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 , \tag{24}$$

   where each $\mathcal{L}_n$ is the composition of a linear function with a convex function.

Derivation (step 1): the MSE is *convex* in the $\mathbf{w}$

There are several ways of proving this:

1. **Way 1.** Recall the definition of $\mathcal{L}$

$$\mathcal{L} := \frac{1}{2N} \sum_{n=1}^{N} \left(\mathcal{L}_n := y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 \,, \tag{24}$$

   where each $\mathcal{L}_n$ is the composition of a linear function with a convex function.
   $\Rightarrow$ We conclude the proof by

Derivation (step 1): the MSE is *convex* in the $\mathbf{w}$

There are several ways of proving this:

1. **Way 1.** Recall the definition of $\mathcal{L}$

$$\mathcal{L} := \frac{1}{2N} \sum_{n=1}^{N} \left(\mathcal{L}_n := y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 , \tag{24}$$

   where each $\mathcal{L}_n$ is the composition of a linear function with a convex function.
   $\Rightarrow$ We conclude the proof by “the sum of convex functions is still a convex function”.

Derivation (step 1): the MSE is *convex* in the $\mathbf{w}$

There are several ways of proving this:

1. **Way 1.** Recall the definition of $\mathcal{L}$

$$\mathcal{L} := \frac{1}{2N} \sum_{n=1}^{N} \left(\mathcal{L}_n := y_n - \mathbf{x}_n^\top \mathbf{w} \right)^2 \,, \tag{24}$$

   where each $\mathcal{L}_n$ is the composition of a linear function with a convex function.
   $\Rightarrow$ We conclude the proof by "the sum of convex functions is still a convex function".

2. **Way 2.** By verifying the definition of convexity, that for any $\lambda \in [0,1]$ and $\mathbf{w}, \mathbf{w}'$,

$$\mathcal{L}(\lambda \mathbf{w} + (1-\lambda)\mathbf{w}') - (\lambda \mathcal{L}(\mathbf{w}) + (1-\lambda)\mathcal{L}(\mathbf{w}')) \leq 0 \,. \tag{25}$$

Derivation (step 1): the MSE is *convex* in the $\mathbf{w}$

There are several ways of proving this:

1. **Way 1.** Recall the definition of $\mathcal{L}$

$$\mathcal{L} := \frac{1}{2N} \sum_{n=1}^{N} (\mathcal{L}_n := y_n - \mathbf{x}_n^\top \mathbf{w})^2 \, , \tag{24}$$

   where each $\mathcal{L}_n$ is the composition of a linear function with a convex function.
   $\Rightarrow$ We conclude the proof by "the sum of convex functions is still a convex function".

2. **Way 2.** By verifying the definition of convexity, that for any $\lambda \in [0, 1]$ and $\mathbf{w}, \mathbf{w}'$,

$$\mathcal{L}(\lambda \mathbf{w} + (1 - \lambda)\mathbf{w}') - (\lambda \mathcal{L}(\mathbf{w}) + (1 - \lambda)\mathcal{L}(\mathbf{w}')) \leq 0 \, . \tag{25}$$

   Exercise.

Derivation (step 1): the MSE is *convex* in the $\mathbf{w}$

There are several ways of proving this:

1. **Way 1.** Recall the definition of $\mathcal{L}$

$$\mathcal{L} := \frac{1}{2N} \sum_{n=1}^{N} \left(\mathcal{L}_n := y_n - \mathbf{x}_n^\top \mathbf{w} \right)^2 , \tag{24}$$

where each $\mathcal{L}_n$ is the composition of a linear function with a convex function.
$\Rightarrow$ We conclude the proof by "the sum of convex functions is still a convex function".

2. **Way 2.** By verifying the definition of convexity, that for any $\lambda \in [0,1]$ and $\mathbf{w}, \mathbf{w}'$,

$$\mathcal{L}(\lambda \mathbf{w} + (1-\lambda)\mathbf{w}') - (\lambda \mathcal{L}(\mathbf{w}) + (1-\lambda)\mathcal{L}(\mathbf{w}')) \leq 0 \,. \tag{25}$$

Exercise. The LHS of our case $-\frac{1}{2N}\lambda(1-\lambda)\left\|\mathbf{X}(\mathbf{w}-\mathbf{w}')\right\|_2^2$ indeed is non-positive.

Derivation (step 1): the MSE is *convex* in the $\mathbf{w}$

There are several ways of proving this:

1. **Way 1.** Recall the definition of $\mathcal{L}$

$$\mathcal{L} := \frac{1}{2N} \sum_{n=1}^{N} \left(\mathcal{L}_n := y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 , \tag{24}$$

where each $\mathcal{L}_n$ is the composition of a linear function with a convex function.
$\Rightarrow$ We conclude the proof by "the sum of convex functions is still a convex function".

2. **Way 2.** By verifying the definition of convexity, that for any $\lambda \in [0, 1]$ and $\mathbf{w}, \mathbf{w}'$,

$$\mathcal{L}(\lambda \mathbf{w} + (1-\lambda)\mathbf{w}') - (\lambda \mathcal{L}(\mathbf{w}) + (1-\lambda)\mathcal{L}(\mathbf{w}')) \leq 0 . \tag{25}$$

Exercise. The LHS of our case $-\frac{1}{2N}\lambda(1-\lambda)\left\|\mathbf{X}(\mathbf{w}-\mathbf{w}')\right\|_2^2$ indeed is non-positive.

3. **Way 3.** Check the second derivative (the Hessian) and show that it is positive semi-definite (all its eigenvalues are non-negative).

Derivation (step 1): the MSE is *convex* in the $\mathbf{w}$

There are several ways of proving this:

1. **Way 1.** Recall the definition of $\mathcal{L}$

$$\mathcal{L} := \frac{1}{2N} \sum_{n=1}^{N} \left(\mathcal{L}_n := y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 , \tag{24}$$

   where each $\mathcal{L}_n$ is the composition of a linear function with a convex function.
   $\Rightarrow$ We conclude the proof by "the sum of convex functions is still a convex function".

2. **Way 2.** By verifying the definition of convexity, that for any $\lambda \in [0,1]$ and $\mathbf{w}, \mathbf{w}'$,

$$\mathcal{L}(\lambda \mathbf{w} + (1-\lambda)\mathbf{w}') - (\lambda \mathcal{L}(\mathbf{w}) + (1-\lambda)\mathcal{L}(\mathbf{w}')) \leq 0 . \tag{25}$$

   Exercise. The LHS of our case $-\frac{1}{2N}\lambda(1-\lambda) \left\|\mathbf{X}(\mathbf{w}-\mathbf{w}')\right\|_2^2$ indeed is non-positive.

3. **Way 3.** Check the second derivative (the Hessian) and show that it is positive semi-definite (all its eigenvalues are non-negative). Exercise.

Derivation (step 1): the MSE is *convex* in the $\mathbf{w}$

There are several ways of proving this:

1. **Way 1.** Recall the definition of $\mathcal{L}$

$$\mathcal{L} := \frac{1}{2N} \sum_{n=1}^{N} \left(\mathcal{L}_n := y_n - \mathbf{x}_n^\top \mathbf{w} \right)^2 \,, \tag{24}$$

   where each $\mathcal{L}_n$ is the composition of a linear function with a convex function.
   $\Rightarrow$ We conclude the proof by "the sum of convex functions is still a convex function".

2. **Way 2.** By verifying the definition of convexity, that for any $\lambda \in [0,1]$ and $\mathbf{w}, \mathbf{w}'$,

$$\mathcal{L}(\lambda \mathbf{w} + (1-\lambda)\mathbf{w}') - (\lambda \mathcal{L}(\mathbf{w}) + (1-\lambda)\mathcal{L}(\mathbf{w}')) \leq 0 \,. \tag{25}$$

   Exercise. The LHS of our case $-\frac{1}{2N}\lambda(1-\lambda) \left\| \mathbf{X}(\mathbf{w} - \mathbf{w}') \right\|_2^2$ indeed is non-positive.

3. **Way 3.** Check the second derivative (the Hessian) and show that it is positive semi-definite (all its eigenvalues are non-negative). Exercise.
   $\Rightarrow$ the Hessian has the form $\frac{1}{N}\mathbf{X}^\top \mathbf{X}$,

Derivation (step 1): the MSE is *convex* in the $\mathbf{w}$

There are several ways of proving this:

1. **Way 1.** Recall the definition of $\mathcal{L}$

$$\mathcal{L} := \frac{1}{2N} \sum_{n=1}^{N} \left(\mathcal{L}_n := y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 , \tag{24}$$

where each $\mathcal{L}_n$ is the composition of a linear function with a convex function.
$\Rightarrow$ We conclude the proof by "the sum of convex functions is still a convex function".

2. **Way 2.** By verifying the definition of convexity, that for any $\lambda \in [0,1]$ and $\mathbf{w}, \mathbf{w}'$,

$$\mathcal{L}(\lambda \mathbf{w} + (1-\lambda)\mathbf{w}') - (\lambda \mathcal{L}(\mathbf{w}) + (1-\lambda)\mathcal{L}(\mathbf{w}')) \leq 0 . \tag{25}$$

Exercise. The LHS of our case $-\frac{1}{2N}\lambda(1-\lambda)\left\|\mathbf{X}(\mathbf{w}-\mathbf{w}')\right\|_2^2$ indeed is non-positive.

3. **Way 3.** Check the second derivative (the Hessian) and show that it is positive semi-definite (all its eigenvalues are non-negative). Exercise.
$\Rightarrow$ the Hessian has the form $\frac{1}{N}\mathbf{X}^\top \mathbf{X}$, which is indeed positive semi-definite.

Derivation (step 1): the MSE is *convex* in the $\mathbf{w}$

There are several ways of proving this:

1. **Way 1.** Recall the definition of $\mathcal{L}$

$$\mathcal{L} := \frac{1}{2N} \sum_{n=1}^{N} \left(\mathcal{L}_n := y_n - \mathbf{x}_n^\top \mathbf{w} \right)^2 , \tag{24}$$

   where each $\mathcal{L}_n$ is the composition of a linear function with a convex function.
   $\Rightarrow$ We conclude the proof by "the sum of convex functions is still a convex function".

2. **Way 2.** By verifying the definition of convexity, that for any $\lambda \in [0,1]$ and $\mathbf{w}, \mathbf{w}'$,

$$\mathcal{L}(\lambda \mathbf{w} + (1-\lambda)\mathbf{w}') - (\lambda \mathcal{L}(\mathbf{w}) + (1-\lambda)\mathcal{L}(\mathbf{w}')) \leq 0 \,. \tag{25}$$

   Exercise. The LHS of our case $-\frac{1}{2N}\lambda(1-\lambda) \left\| \mathbf{X}(\mathbf{w} - \mathbf{w}') \right\|_2^2$ indeed is non-positive.

3. **Way 3.** Check the second derivative (the Hessian) and show that it is positive semi-definite (all its eigenvalues are non-negative). Exercise.
   $\Rightarrow$ the Hessian has the form $\frac{1}{N}\mathbf{X}^\top \mathbf{X}$, which is indeed positive semi-definite.
   $\Rightarrow$ its non-zero eigenvalues are the squares of the non-zero singular values of the matrix $\mathbf{X}$.

Derivation (step 2): finding the minimum of a convex function

By taking the gradient of $\mathcal{L}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 = \frac{1}{2N}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w})$, we have

$$\nabla\mathcal{L}(\mathbf{w}) = -\frac{1}{N}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w})\,. \tag{26}$$

Derivation (step 2): finding the minimum of a convex function

By taking the gradient of $\mathcal{L}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 = \frac{1}{2N}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w})$, we have

$$\nabla\mathcal{L}(\mathbf{w}) = -\frac{1}{N}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w})\,. \tag{26}$$

Given the property of convexity $\nabla\mathcal{L}(\mathbf{w}^\star) = \mathbf{0}$,

Derivation (step 2): finding the minimum of a convex function

By taking the gradient of $\mathcal{L}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 = \frac{1}{2N}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w})$, we have

$$\nabla\mathcal{L}(\mathbf{w}) = -\frac{1}{N}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w})\,. \tag{26}$$

Given the property of convexity $\nabla\mathcal{L}(\mathbf{w}^\star) = \mathbf{0}$, we can get the normal equations for linear regression:

$$\mathbf{X}^\top \underbrace{(\mathbf{y} - \mathbf{X}\mathbf{w})}_{\text{error}} = \mathbf{0}\,, \tag{27}$$

where the error $\mathbf{e} := \mathbf{y} - \mathbf{X}\mathbf{w}$ is orthogonal to all columns of $\mathbf{X}$.

Geometric interpretation of Normal Equations

Definition 10 (Span of a set of vectors)

The **span** of a set of vectors, $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, is the set of all possible linear combinations of these vectors; i.e. $\text{span}\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} = \{\alpha_1\mathbf{x}_1 + \ldots + \alpha_k\mathbf{x}_k \,|\, \alpha_1, \ldots, \alpha_k \in \mathbb{R}\}$.

Geometric interpretation of Normal Equations

Definition 10 (Span of a set of vectors)

The **span** of a set of vectors, $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, is the set of all possible linear combinations of these vectors; i.e. span$\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} = \{\alpha_1 \mathbf{x}_1 + \ldots + \alpha_k \mathbf{x}_k \,|\, \alpha_1, \ldots, \alpha_k \in \mathbb{R}\}$.

- The span of $\mathbf{X} \in \mathbb{R}^{N \times D}$ is the space spanned by the columns of $\mathbf{X}$

$$\mathcal{S} := span(\mathbf{X}) = \{\mathbf{u} := \mathbf{X}\mathbf{w} | \mathbf{w} \in \mathbb{R}^D\}$$

Geometric interpretation of Normal Equations

Definition 10 (Span of a set of vectors)

The **span** of a set of vectors, $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, is the set of all possible linear combinations of these vectors; i.e. span$\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} = \{\alpha_1 \mathbf{x}_1 + \ldots + \alpha_k \mathbf{x}_k \mid \alpha_1, \ldots, \alpha_k \in \mathbb{R}\}$.

- The span of $\mathbf{X} \in \mathbb{R}^{N \times D}$ is the space spanned by the columns of $\mathbf{X}$

$$\mathcal{S} := span(\mathbf{X}) = \{\mathbf{u} := \mathbf{X}\mathbf{w} | \mathbf{w} \in \mathbb{R}^D\}$$

Which element $\mathbf{u}$ of $span(\mathbf{X})$ shall we take such that the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$?

Geometric interpretation of Normal Equations

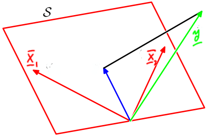
Definition 10 (Span of a set of vectors)

The **span** of a set of vectors, $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, is the set of all possible linear combinations of these vectors; i.e. span$\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} = \{\alpha_1 \mathbf{x}_1 + \ldots + \alpha_k \mathbf{x}_k \,|\, \alpha_1, \ldots, \alpha_k \in \mathbb{R}\}$.

- The span of $\mathbf{X} \in \mathbb{R}^{N \times D}$ is the space spanned by the columns of $\mathbf{X}$

$$\mathcal{S} := span(\mathbf{X}) = \{\mathbf{u} := \mathbf{X}\mathbf{w} | \mathbf{w} \in \mathbb{R}^D\}$$

Which element $\mathbf{u}$ of $span(\mathbf{X})$ shall we take such that the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$?



(taken from Bishop's book)

Geometric interpretation of Normal Equations

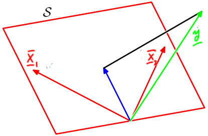
Definition 10 (Span of a set of vectors)

The **span** of a set of vectors, $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, is the set of all possible linear combinations of these vectors; i.e. span$\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} = \{\alpha_1 \mathbf{x}_1 + \ldots + \alpha_k \mathbf{x}_k \mid \alpha_1, \ldots, \alpha_k \in \mathbb{R}\}$.

- The span of $\mathbf{X} \in \mathbb{R}^{N \times D}$ is the space spanned by the columns of $\mathbf{X}$

$$\mathcal{S} := span(\mathbf{X}) = \{\mathbf{u} := \mathbf{X}\mathbf{w} | \mathbf{w} \in \mathbb{R}^D\}$$

Which element $\mathbf{u}$ of $span(\mathbf{X})$ shall we take such that the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$?



(taken from Bishop's book)

From $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$, we have:

Geometric interpretation of Normal Equations

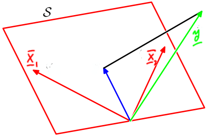
Definition 10 (Span of a set of vectors)

The **span** of a set of vectors, $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, is the set of all possible linear combinations of these vectors; i.e. $\text{span}\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} = \{\alpha_1 \mathbf{x}_1 + \ldots + \alpha_k \mathbf{x}_k \mid \alpha_1, \ldots, \alpha_k \in \mathbb{R}\}$.

- The span of $\mathbf{X} \in \mathbb{R}^{N \times D}$ is the space spanned by the columns of $\mathbf{X}$

$$\mathcal{S} := span(\mathbf{X}) = \{\mathbf{u} := \mathbf{X}\mathbf{w} | \mathbf{w} \in \mathbb{R}^D\}$$

Which element $\mathbf{u}$ of $span(\mathbf{X})$ shall we take such that the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$?



(taken from Bishop's book)

From $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$, we have:

- the optimum choice for $\mathbf{u}$, i.e. $\mathbf{u}^\star$, requires $\mathbf{y} - \mathbf{u}^\star$ to be orthogonal to $span(\mathbf{X})$.

Geometric interpretation of Normal Equations

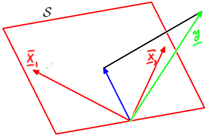
Definition 10 (Span of a set of vectors)

The **span** of a set of vectors, $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, is the set of all possible linear combinations of these vectors; i.e. span$\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} = \{\alpha_1 \mathbf{x}_1 + \ldots + \alpha_k \mathbf{x}_k \,|\, \alpha_1, \ldots, \alpha_k \in \mathbb{R}\}$.

- The span of $\mathbf{X} \in \mathbb{R}^{N \times D}$ is the space spanned by the columns of $\mathbf{X}$

$$\mathcal{S} := span(\mathbf{X}) = \{\mathbf{u} := \mathbf{X}\mathbf{w} | \mathbf{w} \in \mathbb{R}^D\}$$

Which element $\mathbf{u}$ of $span(\mathbf{X})$ shall we take such that the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$?



(taken from Bishop's book)

From $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$, we have:

- the optimum choice for $\mathbf{u}$, i.e. $\mathbf{u}^\star$, requires $\mathbf{y} - \mathbf{u}^\star$ to be orthogonal to $span(\mathbf{X})$.
- $\mathbf{u}^\star$ should be equal to *the projection of* $\mathbf{y}$ *onto* $span(\mathbf{X})$.

Least Squares

We need to solve the linear system of the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$, where

$$\mathbf{X}^\top y = \underbrace{\mathbf{X}^\top \mathbf{X}}_{\text{Gram matrix}} \mathbf{w} \tag{28}$$

Least Squares

We need to solve the linear system of the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$, where

$$\mathbf{X}^\top y = \underbrace{\mathbf{X}^\top\mathbf{X}}_{\text{Gram matrix}} \mathbf{w} \tag{28}$$

If the Gram matrix is invertible,

Least Squares

We need to solve the linear system of the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$, where

$$\mathbf{X}^\top y = \underbrace{\mathbf{X}^\top\mathbf{X}}_{\text{Gram matrix}} \mathbf{w} \tag{28}$$

If the Gram matrix is invertible, we can multiply the normal equation by the inverse of the Gram matrix from the left:

$$\mathbf{w}^\star = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}\,, \tag{29}$$

Least Squares

We need to solve the linear system of the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$, where

$$\mathbf{X}^\top y = \underbrace{\mathbf{X}^\top\mathbf{X}}_{\text{Gram matrix}} \mathbf{w} \tag{28}$$

If the Gram matrix is invertible, we can multiply the normal equation by the inverse of the Gram matrix from the left:

$$\mathbf{w}^\star = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}\,, \tag{29}$$

where we can get a closed-form expression for the minimum.

Least Squares

We need to solve the linear system of the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$, where

$$\mathbf{X}^\top y = \underbrace{\mathbf{X}^\top\mathbf{X}}_{\text{Gram matrix}} \mathbf{w} \tag{28}$$

If the Gram matrix is invertible, we can multiply the normal equation by the inverse of the Gram matrix from the left:

$$\mathbf{w}^\star = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}\,, \tag{29}$$

where we can get a closed-form expression for the minimum.

We can use this model to predict a new value for an unseen datapoint (test point) $\mathbf{x}_m$:

$$\hat{y}_m := \mathbf{x}_m^\top\mathbf{w}^\star = \mathbf{x}_m^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}\,. \tag{30}$$

Least Squares

We need to solve the linear system of the normal equation $\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$, where

$$\mathbf{X}^\top y = \underbrace{\mathbf{X}^\top\mathbf{X}}_{\text{Gram matrix}} \mathbf{w} \tag{28}$$

If the Gram matrix is invertible, we can multiply the normal equation by the inverse of the Gram matrix from the left:

$$\mathbf{w}^\star = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}\,, \tag{29}$$

where we can get a closed-form expression for the minimum.

We can use this model to predict a new value for an unseen datapoint (test point) $\mathbf{x}_m$:

$$\hat{y}_m := \mathbf{x}_m^\top\mathbf{w}^\star = \mathbf{x}_m^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}\,. \tag{30}$$

Remark 11

The Gram matrix $\mathbf{X}^\top\mathbf{X} \in \mathbb{R}^{D\times D}$ is invertible if and only if $\mathbf{X}$ has full column rank, or in other words $rank(\mathbf{X}) = D$.

Rank deficiency and ill-conditioning

Unfortunately, in practice, $\mathbf{X} \in \mathbb{R}^{N \times D}$ is often rank deficient.

Rank deficiency and ill-conditioning

Unfortunately, in practice, $\mathbf{X} \in \mathbb{R}^{N \times D}$ is often rank deficient.

For example,

Rank deficiency and ill-conditioning

Unfortunately, in practice, $\mathbf{X} \in \mathbb{R}^{N \times D}$ is often rank deficient.

For example,

- If $D > N$ (namely over-parameterized), we always have $rank(\mathbf{X}) < D$

Rank deficiency and ill-conditioning

Unfortunately, in practice, $\mathbf{X} \in \mathbb{R}^{N \times D}$ is often rank deficient.

For example,

- If $D > N$ (namely over-parameterized), we always have $rank(\mathbf{X}) < D$ (since row rank = col. rank)

Rank deficiency and ill-conditioning

Unfortunately, in practice, $\mathbf{X} \in \mathbb{R}^{N \times D}$ is often rank deficient.

For example,

- If $D > N$ (namely over-parameterized), we always have $rank(\mathbf{X}) < D$ (since row rank = col. rank)
- If $D \leq N$, but some of the columns $\mathbf{x}_{:d}$ are (nearly) collinear

Rank deficiency and ill-conditioning

Unfortunately, in practice, $\mathbf{X} \in \mathbb{R}^{N \times D}$ is often rank deficient.

For example,

- If $D > N$ (namely over-parameterized), we always have $rank(\mathbf{X}) < D$ (since row rank = col. rank)
- If $D \leq N$, but some of the columns $\mathbf{x}_{:d}$ are (nearly) collinear $\Rightarrow \mathbf{X}$ is ill-conditioned, leading to numerical issues when solving the linear system.

Rank deficiency and ill-conditioning

Unfortunately, in practice, $\mathbf{X} \in \mathbb{R}^{N \times D}$ is often rank deficient.

For example,

- If $D > N$ (namely over-parameterized), we always have $rank(\mathbf{X}) < D$ (since row rank = col. rank)

- If $D \leq N$, but some of the columns $\mathbf{x}_{:d}$ are (nearly) collinear
$\Rightarrow \mathbf{X}$ is ill-conditioned, leading to numerical issues when solving the linear system.

Can we solve least squares if $\mathbf{X}$ is rank deficient?

Rank deficiency and ill-conditioning

Unfortunately, in practice, $\mathbf{X} \in \mathbb{R}^{N \times D}$ is often rank deficient.

For example,

- If $D > N$ (namely over-parameterized), we always have $rank(\mathbf{X}) < D$ (since row rank = col. rank)

- If $D \leq N$, but some of the columns $\mathbf{x}_{:d}$ are (nearly) collinear $\Rightarrow \mathbf{X}$ is ill-conditioned, leading to numerical issues when solving the linear system.

Can we solve least squares if $\mathbf{X}$ is rank deficient?
Yes, using a linear system solver, e.g., $\mathrm{np.linalg.solve}(\mathbf{X}, \mathbf{y})$.

This lecture:

- Basic concept of regression and classification
- Linear regression
  - Definition
  - Gradient Descent (GD) optimization
- Normal Equation and Least Squares

Next lecture:

- The probabilistic interpretation of Linear Regression
- Maximum Likelihood Estimation (MLE)
- Over-fitting and Under-fitting
- Polynomial Regression, Ridge Regression, and Lasso