Universidade de São Paulo
Escola Politécnica - Engenharia de Computação e Sistemas Digitais

Machine Learning Basics

Prof. Artur Jordão

The Many Faces of Artificial Intelligence

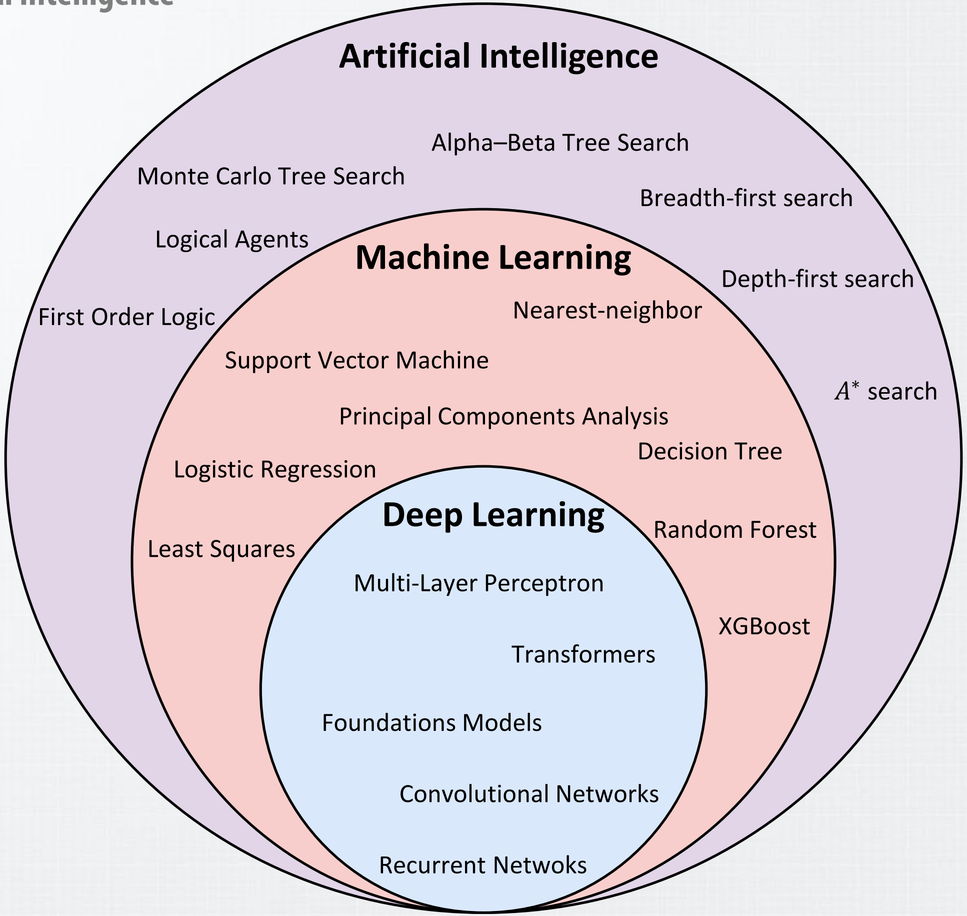
Definitions

The Many Faces of Artificial Intelligence

- Artificial Intelligence
  - Any technique that enables computers to mimic human behavior
- Machine Learning
  - Ability to learn without explicitly being programmed
- Deep Learning
  - Extract patterns from data using neural networks

Machine Learning and Deep Learning

The Many Faces of Artificial Intelligence

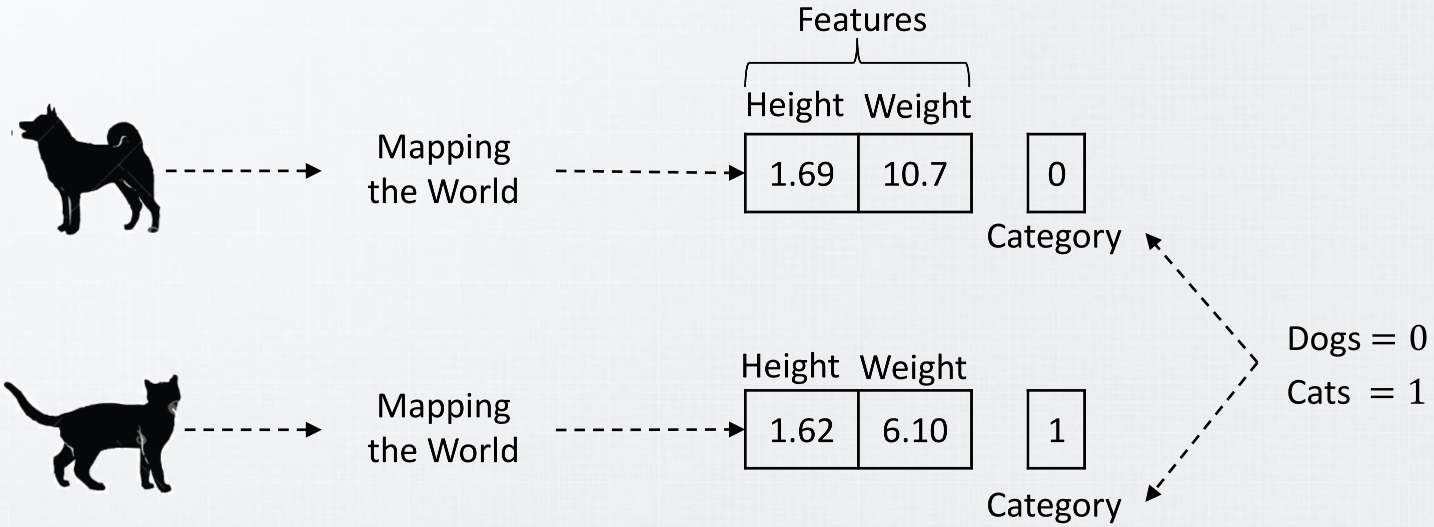


Independent (X) and Dependent (Y) Variables

Preliminaries

Independent and Dependent Variables

- Mapping the state of the world (features – m)
  - How can we represent the states (i.e., objects) of the world?
  - Numerical representation (features engineering)



Preliminaries

Independent and Dependent Variables

- Assume that we represent (map) n dogs and cats using m features

- Let $X \in \mathbb{R}^{n \times m}$
  - Independent variables
  - Data samples – each dog/cat composes a row in X

- Let $Y \in \mathbb{R}^{n \times c}$
  - c stands for the number of categories (i.e., 2)
  - Dependent variable
  - Labels/Classes

$$n\left\{\underbrace{\overset{X}{\begin{array}{|c|c|}\hline 1.69 & 10.7 \\ \hline 1.85 & 14.5 \\ \hline 2.67 & 20.3 \\ \hline .... & .... \\ \hline 1.62 & 6.1 \\ \hline 1.11 & 4.0 \\ \hline \end{array}}}_{m}\right. \qquad \overset{Y}{\begin{array}{|c|}\hline 0 \\ \hline 0 \\ \hline 0 \\ \hline . \\ \hline 1 \\ \hline 1 \\ \hline \end{array}}$$

Preliminaries

Independent and Dependent Variables

- Let $x_i \in \mathbb{R}^{1 \times m}$ be the ith sample (example) of X
  - We can express x_i in terms of its features $x_i = x_i^1, x_i^2 \ldots x_i^m$
- Let $y_i \in \mathbb{R}^{1 \times 1}$ be the ith label of Y

	X		Y	
	1.69	10.7	0	
	1.85	14.5	0	
x_i	2.67	20.3	0	y_i
	....	....	.	
	1.62	6.1	1	
	1.11	4.0	1	

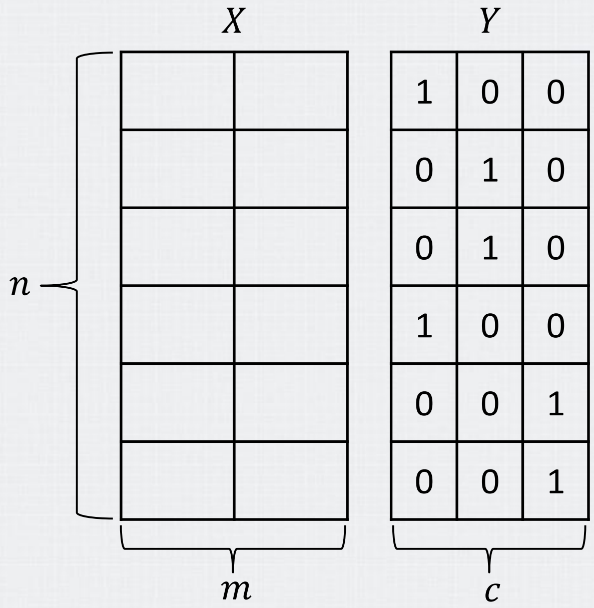
Multiclass Problems

Independent and Dependent Variables

- Problems involving more than two classes
  - In this case, we cannot use 0 and 1
  - For example, dogs, cats and bears

- One-hot encoding
  - Transform the c classes into a zero-one vector
  - The cth entry equal to 1 and the rest 0

- $Y \in \mathbb{R}^{n \times c}$

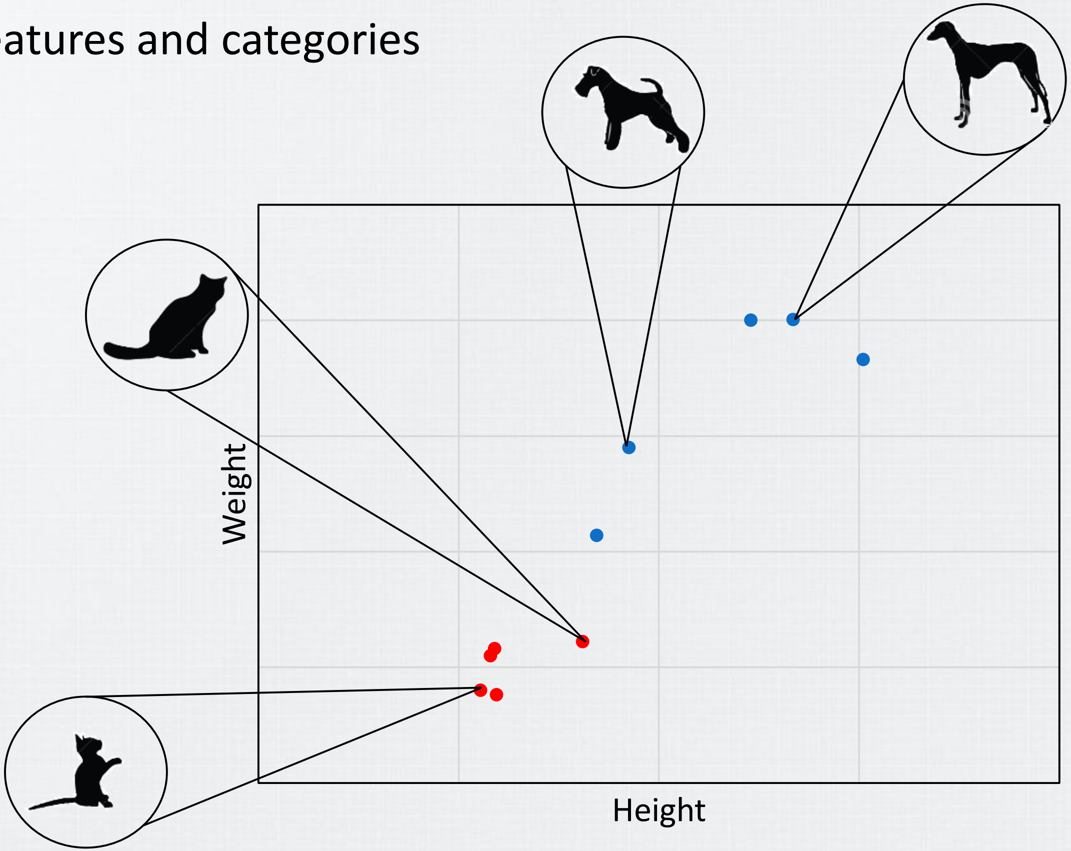


Feature Space

Independent and Dependent Variables

- Projects the data using its features and categories

- Technical details
  - Each feature (m) is an axis
  - Each sample (n) is a point
  - Each category (y) is a color



Problem Statement

Independent and Dependent Variables

- Given X, the problem is to predict Y to new samples
  - Given the features, we want to predict the category to which samples belong

- Assume $\mathcal{F}(\cdot,\cdot)$ a model parameterized by a set of parameters/weights θ that receives x and outputs $\bar{y}$
  - Formally, $\bar{y} = \mathcal{F}(x, \theta)$ (or $\bar{Y} = \mathcal{F}(X, \theta)$)

- **The problem is, therefore, to discover θ**

X

1.33	8.7
2.78	3.5
2.29	7.3
....	....
1.62	9.1
2.15	10.0

$\bar{Y}$ (unseen)

?
?
?
.
?
?

Problem Statement

Independent and Dependent Variables

- Training (or learning) phase
  - Estimate θ using X and Y (seen)

- Testing phase
  - Employ θ onto X to predict $\bar{Y}$ (unseen)
  - Unseen means new samples

Training Phase

X		Y (seen)
1.69	10.7	0
1.85	14.5	0
....	....	.
1.62	6.1	1
1.11	4.0	1

Testing Phase

X		$\bar{Y}$ (unseen)
1.33	8.7	?
2.78	3.5	?
....	....	.
1.62	9.1	?
2.15	10.0	?

Classification vs. Regression

Independent and Dependent Variables

- Classification
  - The goal is to predict a category between c possible categories
- Regression
  - The goal is to predict a real-valued target
- For both classification and regression, the problem is to discover θ
  - $\bar{Y} = \mathcal{F}(X, \theta)$

X (features)	Task	Y (target)
Weight, Height (Animal features)	Classification	Dogs, Cats, Bears
	Regression	Lifespan
Area, location, number of bedrooms (House features)	Classification	House Quality (good, bad, medium)
	Regression	House Price

Preliminaries

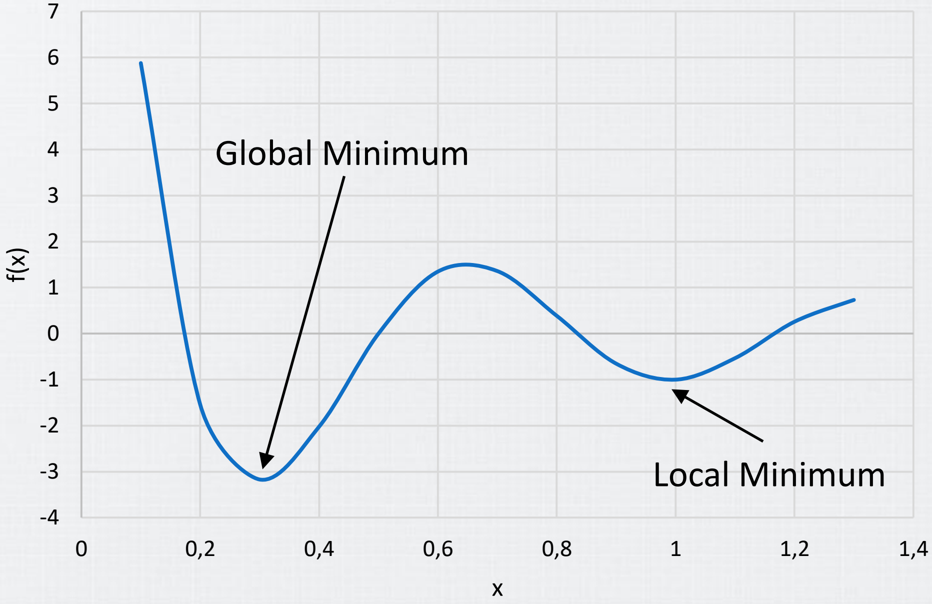
Functions

Preliminaries

- A function is a relation that associates each element x to **a single** element $f(x)$

- Global minimum

- Local minimum

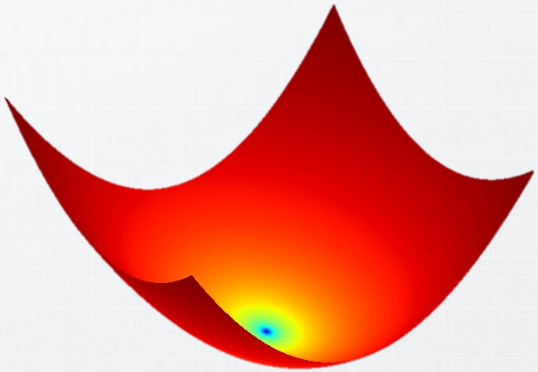


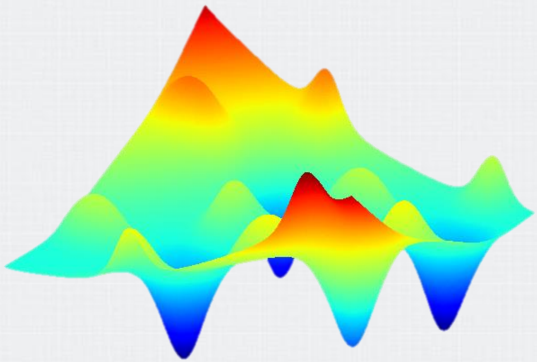
Convex and Non-Convex Problems

Preliminaries

- Convex Problems: **only one** global minimum (or maximum)
  - It facilitates the optimization process (i.e., discovering θ)

- Non-Convex Problems: multiple local minimum (or maximum)
  - Multiple optimal local solutions
  - Most practical problems belong to this category – deep learning problems



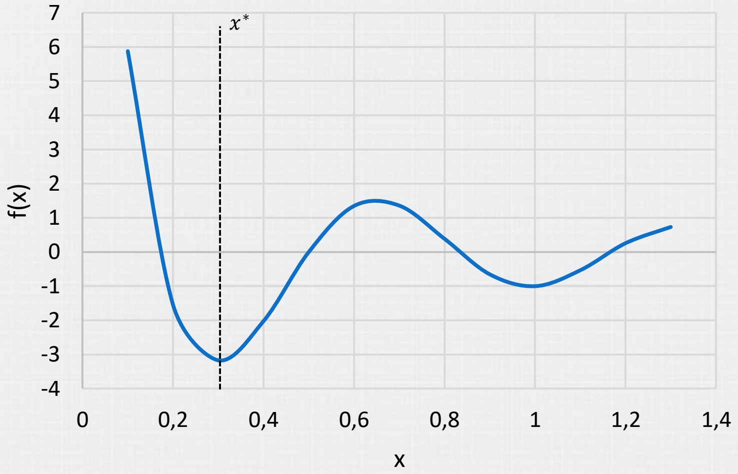


Loss Function

Preliminaries

- The function we want to minimize or maximize is called the objective function or criterion
  - When we are minimizing a function, we may also call it the **cost function**, **loss function**, or **error function**

- We often denote the value that minimizes (or maximizes) a function with a superscript *
  - $x^* = argmin\, f(x)$



Loss Function

Preliminaries

- Quantify the distance between the real (Y) and predicted values of the target
  - Real values: y or Y
  - Predicted values: $\mathcal{F}(x, \theta) = \hat{y}$ or $\mathcal{F}(X, \theta) = \hat{Y}$

- Loss function properties
  - Monotonicity: The better the model gets, the lower the value of the loss function
  - Differentiability: Differentiable with respect to θ

Common Losses	Short Description
Squared error	$\frac{1}{2}(\hat{y} - y)^2$
Mean Square Error	$\frac{1}{n}\sum_{i=0}^{n}(\hat{y}_i - y_i)^2$
Cross Entropy	$-\sum_{i=0}^{c} y_i \log(\hat{y}_i)$

The Role of the Derivative

Preliminaries

- Define a function $y = f(x)$ and $f'(x)$ the derivative of f
  - x and y are both real numbers

- The derivative $f'(x)$ gives the slope of $f(x)$ at the point x
  - It specifies how to scale a small change in the input in order to obtain the corresponding change in the output: $f(x+\epsilon) \approx f(x) + \epsilon f'(x)$

Change in the value of quantity

Quantity

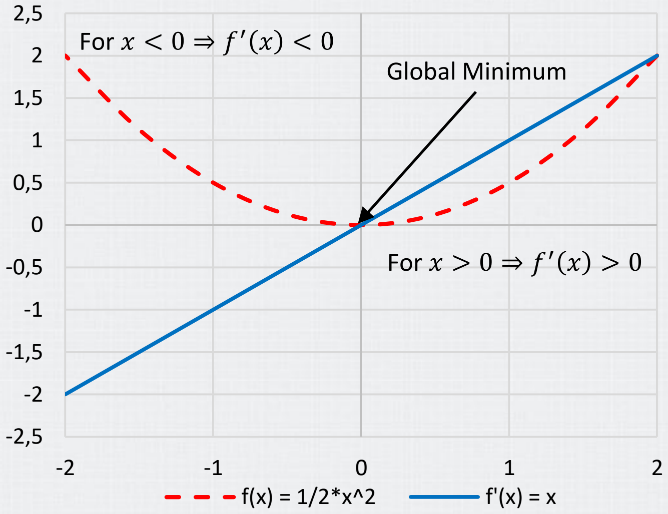
New value Old value

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{\overbrace{f(x+h) - f(x)}^{\text{Change in the value of quantity}}}{h}$$

w.r.t

Time interval

Tends to zero (becomes very small)



The Role of the Derivative

Preliminaries

- The **gradient** generalizes the notion of derivative to the case where the derivative is with respect to a vector

- The gradient of f is the vector containing all of the partial derivatives $\nabla_x f(x)$
  - Element i of the gradient is the partial derivative of f with respect to x_i

Linear Regression

Regression

Linear Regression

- Regression refers to a set of methods for modeling the relationship between one or more independent variables and a dependent variable

- Linear regression
  - The simplest and most popular among the standard tools for regression

Assumptions of Linear Regression

- Linearity assumption
  - The relationship between the independent variables x and the dependent variable y is linear
  - y can be expressed as a **weighted sum** of the elements in x, given some noise on the observations

- Any noise is well-behaved
  - Following a Gaussian distribution

Linear Model

Linear Regression

- $\hat{y} = w_1 * x^1 + w_2 * x^2 + \cdots + w_m * x^m + \boldsymbol{b}$
  - $W = w_1, w_2, \ldots, w_m$
  - We can organize W as column or row matrix

- Dot product form (single sample prediction)
  - $\hat{y} = xW^T + b$

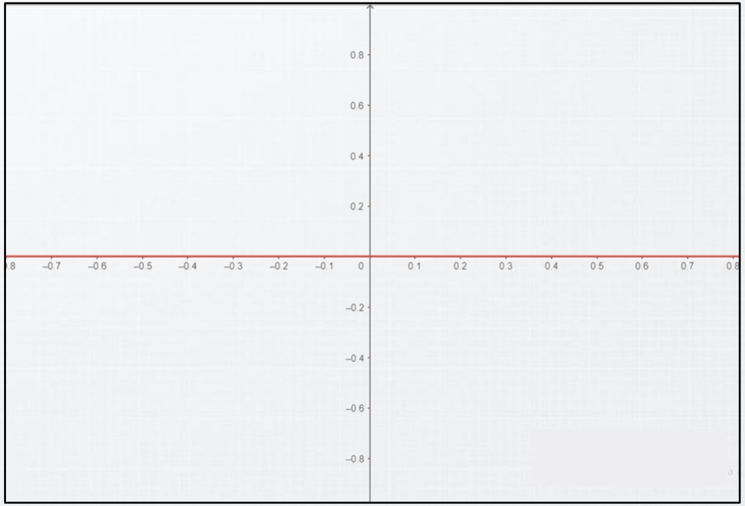
- Matrix vector product form (n samples prediction – all at once)
  - $\hat{Y} = XW^T + b$

The Bias Term

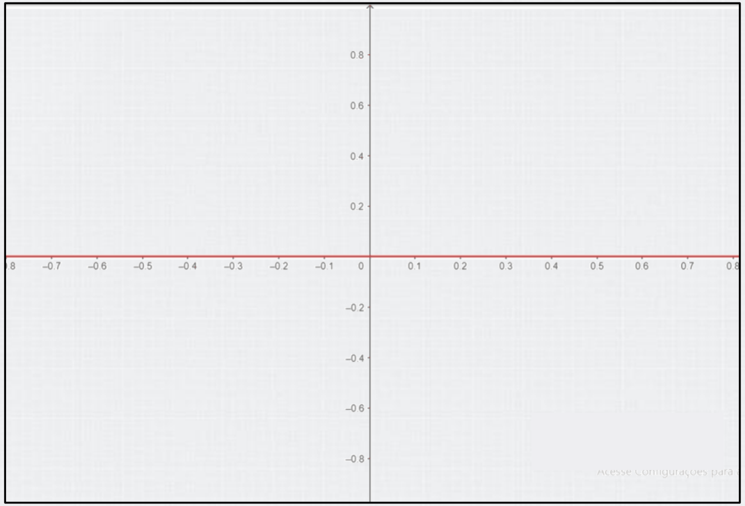
Linear Regression

- The bias term plays a role in the expressivity of the model
  - It allows the model to fit not only the data points that **pass through the origin but also those that do not**
  - It enables the model to capture and represent more complex relationships between the features and the target variable

Without Bias



With Bias



The Bias Term

Linear Regression

- Putting the bias into the parameter matrix w_i
  - We can subsume the bias b into the parameter matrix W by appending a column to the design matrix consisting of all **ones**

X

1.69	10.7	1
1.85	14.5	1
2.67	20.3	1
....	....	...
1.62	6.1	1
1.11	4.0	1

Y (observable)

0
0
0
.
1
1

X

1.33	8.7	1
2.78	3.5	1
2.29	7.3	1
....	....	...
1.62	9.1	1
2.15	10.0	1

$\bar{Y}$ (unseen)

?
?
?
.
?
?

Problem Definition

Linear Regression

- How can we discover W (and b) that minimizes the total loss across all training examples?
  - Formally: $\mathrm{W}^*, b^* = argmin\, \mathcal{L}(W, b) \Rightarrow \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\big(\overbrace{\boldsymbol{x_i}\boldsymbol{W^T} + \boldsymbol{b}}^{\hat{y}} - y_i\big)^2$
  - Here $\mathcal{L}(\cdot)$ means a loss function

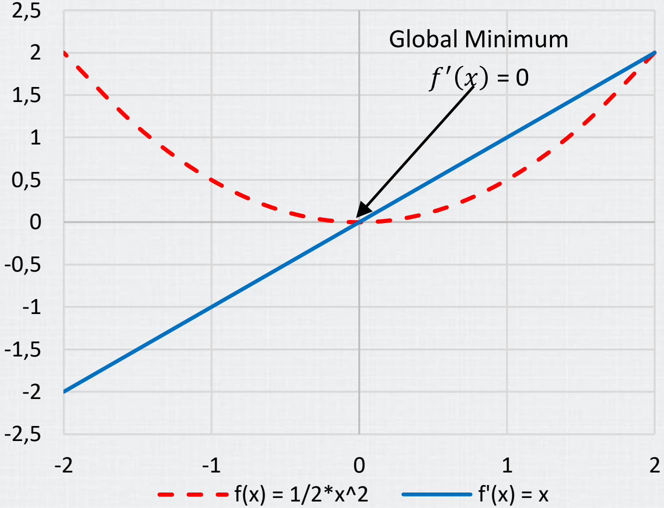
- Techniques
  - Analytic Solution
  - Gradient-Based Optimization

Analytic Solution

Linear Regression

- $\mathcal{L}(W) = ||XW^T - Y||$
  - The loss across all training samples

- $\nabla_w \mathcal{L}(W) = 2X^T(XW^T - Y) = 0$
  - We set the gradient to zero to find the points where the loss function reaches the minimum



- $W^* = \underbrace{(X^TX)^{-1}X^T}Y$

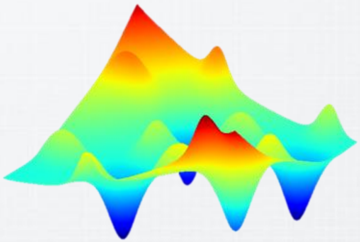
The **pseudoinverse** of X

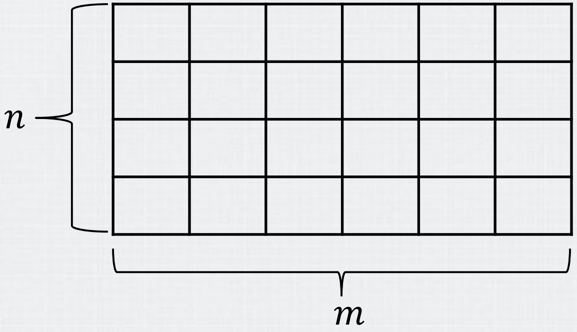
Analytic Solution

Linear Regression

- Problems
  - It does not work for complex (non-convex – multiple local optima) problems

- The sample size problem (a.k.a zero determinant or singularity)
  - The number of samples is smaller than the number of features $(n < m)$





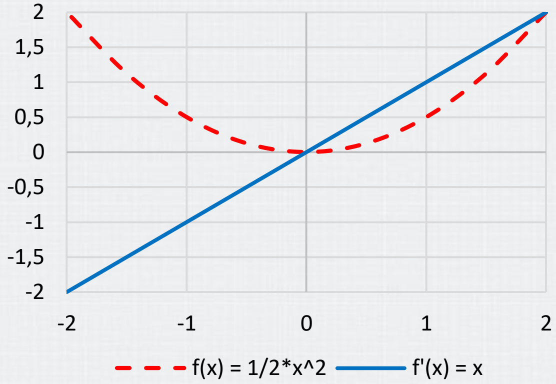
Gradient Descent

Linear Regression

- The gradient is useful for minimizing a function
  - It tells us how to change x in order to make a small improvement in y

- We can reduce $f(x)$ by moving x in small steps with **opposite sign of the gradient**

- The key consists of **iteratively** reducing the error by updating the parameters (weights) in the direction that incrementally lowers the loss function
  - **Gradient Descent**



Gradient Descent

Linear Regression

- Learning rate (η)
  - Positive scalar determining the size of the step
  - The rate of learning

- Convergence
  - Run for a defined number of iterations (**epochs**)
  - Stop when the loss does not decrease (early stop)

Gradient Descent Algorithm
$\boldsymbol{W} \leftarrow$ Random values
While not converged do
for each $\boldsymbol{w_i} \in \boldsymbol{W}$ do
$\boldsymbol{w_i} \leftarrow \boldsymbol{w_i} - \eta \frac{\partial}{\partial \boldsymbol{w_i}} \mathcal{L}(\boldsymbol{W})$

Gradient Descent

Linear Regression

- The Gradient Descent takes the derivative of the loss function, which is an **average of the losses** computed on every single example ($x \in X$)

- In practice, this can be extremely slow and memory costly
  - We must pass over the entire dataset before making a single update
  - The problem is compounded if n is larger than the processor's memory size

Stochastic Gradient Descent (SGD)

Linear Regression

- Stochastic Gradient Descent randomly selects a small number (**batch size** – β) of training examples at each step t, and updates according to

$$W \leftarrow W - \eta \frac{1}{|\beta|} \sum_{j=\beta_t}^{n} \frac{\partial}{\partial \boldsymbol{W}} \mathcal{L}^j(\boldsymbol{W})$$

Stochastic Gradient Descent Algorithm
$\boldsymbol{W} \leftarrow$ Random values
While not converged do
for each $\boldsymbol{w_i} \in \boldsymbol{W}$ do
$\boldsymbol{w_i} \leftarrow \boldsymbol{w_i} - \eta \frac{1}{\lvert\beta\rvert} \sum_{j=\beta_t}^{n} \frac{\partial}{\partial \boldsymbol{w_i}} \mathcal{L}^j(\boldsymbol{W})$

Learning and Testing Phase

Definitions

Learning and Testing Phase

- Define $D = \{(x_i, y_i)\}_{i=1}^{n}$ a dataset

- Let D_{train} be a subset of D such that $D_{train} \subseteq D$
  - Samples and their (seen) labels

- Let D_{test} be a subset of D such that $D_{test} \subseteq D$
  - In practice, unseen labels

- Important properties
  - $D = D_{train} \cup D_{test}$
  - $\boldsymbol{D_{train}} \cap \boldsymbol{D_{test}} = \emptyset$

Overview

Learning and Testing Phase

Learning Phase

(Here X and Y come from D_{train})

```
model.fit(X,Y)
```

$$(X^T X)^{-1} X^T Y$$

$$\boldsymbol{w_i} \leftarrow \boldsymbol{w_i} - \eta \frac{\partial}{\partial \boldsymbol{w_i}} \mathcal{L}(\boldsymbol{W})$$

$$\boldsymbol{w_i} \leftarrow \boldsymbol{w_i} - \eta \frac{1}{|\beta|} \sum_{j=\beta_t}^{n} \frac{\partial}{\partial \boldsymbol{w_i}} \mathcal{L}^j(\boldsymbol{W})$$

Testing Phase

(Here X comes from D_{test})

```
y_pred = model.predict(X)
```

$$\hat{Y} = XW^T + b$$

Quality of the Learning Trajectory

Learning and Testing Phase

- How to measure the quality of the training trajectory?
  - Dynamics of training

- Loss curve

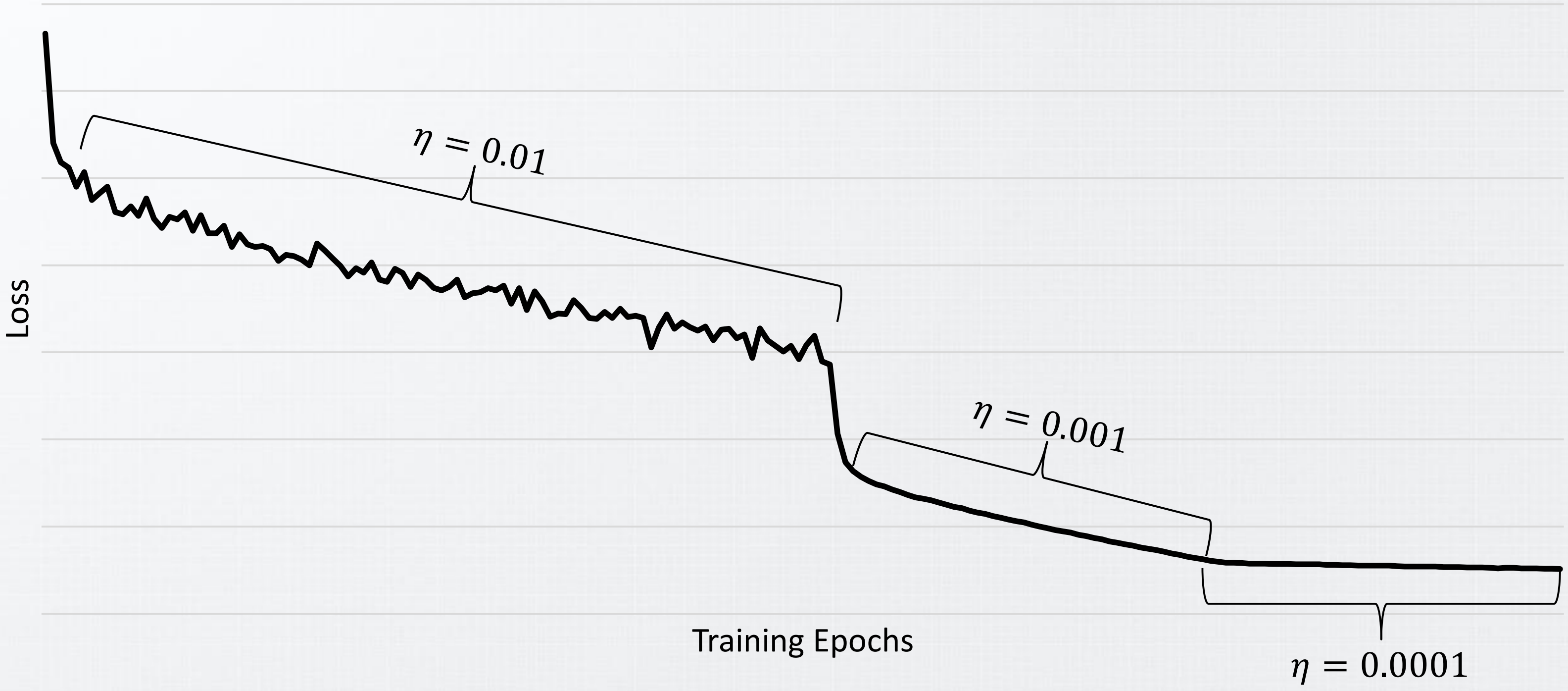
- Loss landscape (Li et al., 2018)

Li et al. *Visualizing the Loss Landscape of Neural Nets*. In Neural Information Processing Systems (NeurIPS), 2018

Loss Curve

Learning and Testing Phase

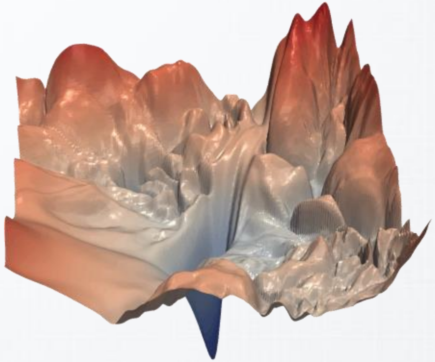
- Early stop

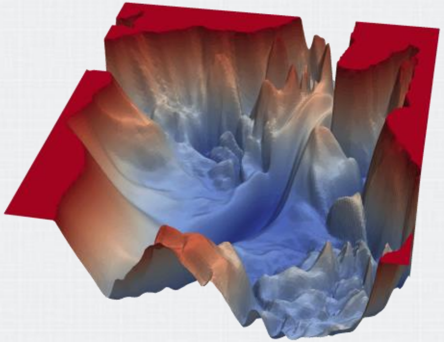


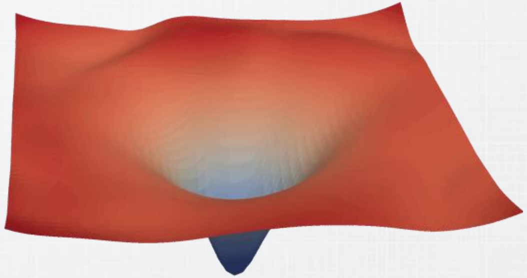
Loss Landscape

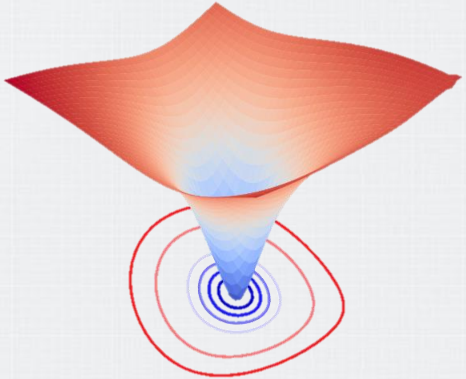
Learning and Testing Phase

- http://www.telesens.co/loss-landscape-viz/viewer.html









Li et al. *Visualizing the Loss Landscape of Neural Nets*. In Neural Information Processing Systems (NeurIPS), 2018

Testing Phase

Learning and Testing Phase

- Once the training is done, how can we measure the predictive ability of the model?

- Predictive ability (quantitative) metrics
  - Accuracy
  - Confusion Matrix
  - Loss
  - Peason Correlation
  - Pair-wise

- The metric depends on the application
  - Some benchmarks have their own metrics
  - For example, accuracy on CIFAR-10 and Top-5 accuracy/error on ImageNet

Overfitting, Underfitting, Generalization and No Free Lunch

Generalization

Overfitting, Underfitting, Generalization and No Free Lunch

- Generalization
    - The quality of the model in predicting new (unseen) data

- Overfitting
    - The model performs well on training but poorly on testing/validation
    - Complex models tend to (we can avoid/handle this) overfit the data

- Underfitting
    - The model fails to find a pattern in the data

The Bias-Variance Tradeoff

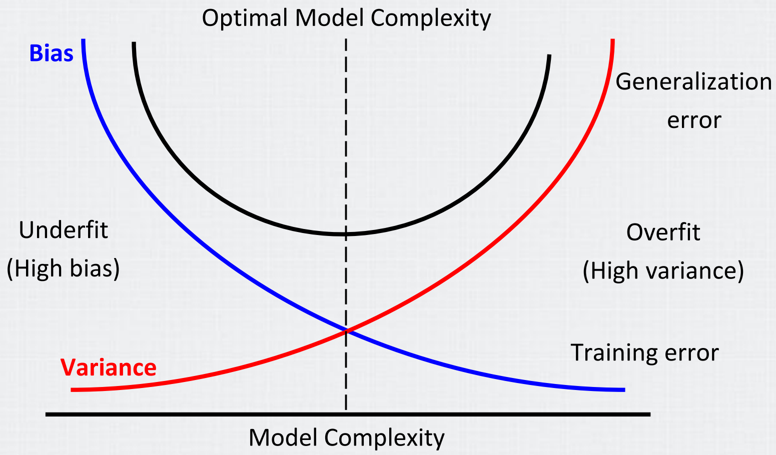
Overfitting, Underfitting, Generalization and No Free Lunch

- Bias
  - Low bias: the model predicts well the samples of the training data
  - High bias: the model makes many mistakes in the training data

- Variance
  - Error of the model due to its sensitivity to small fluctuations in the training set

- Bias–variance tradeoff
  - Low-bias hypotheses that fit the training data well
  - Low-variance hypotheses that **may** generalize better

- U-shaped curve



No Free Lunch Theorem

- The *no free lunch theorem* states that every learning algorithm is as good as any other when averaged over all possible problems (Wolpert, 1996)
  - **There is no universal learning algorithm able to solve all tasks precisely**

- Under a uniform distribution over problems (search/learning problems), all algorithms perform equally
  - A particular model or algorithm is **better** than average on some problems, it must be **worse** than average on others

Wolpert. *The lack of a prior distinctions between learning algorithms and the existence of a priori distinctions between learning algorithms*. Neural Computation, 1996

Normalization

Z-Score

Normalization

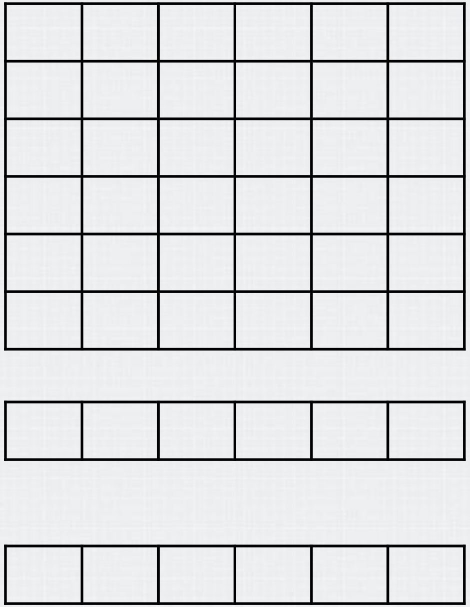
- Suppose we are mapping the world using the following features
  - $x^1 \in [0, 100], x^2 \in [0, 1], \dots, x^m \in [-\infty, +\infty]$

$[0, 100]$ $[0, 1]$ $[-\infty, +\infty]$

$$\hat{y} = w_1 * x^1 + w_2 * x^2 + \cdots + w_m * x^m + \boldsymbol{b}$$

- Z-score normalization
  - $X \leftarrow \frac{X - \mu}{\sigma}$

$X \in \mathbb{R}^{n \times m}$

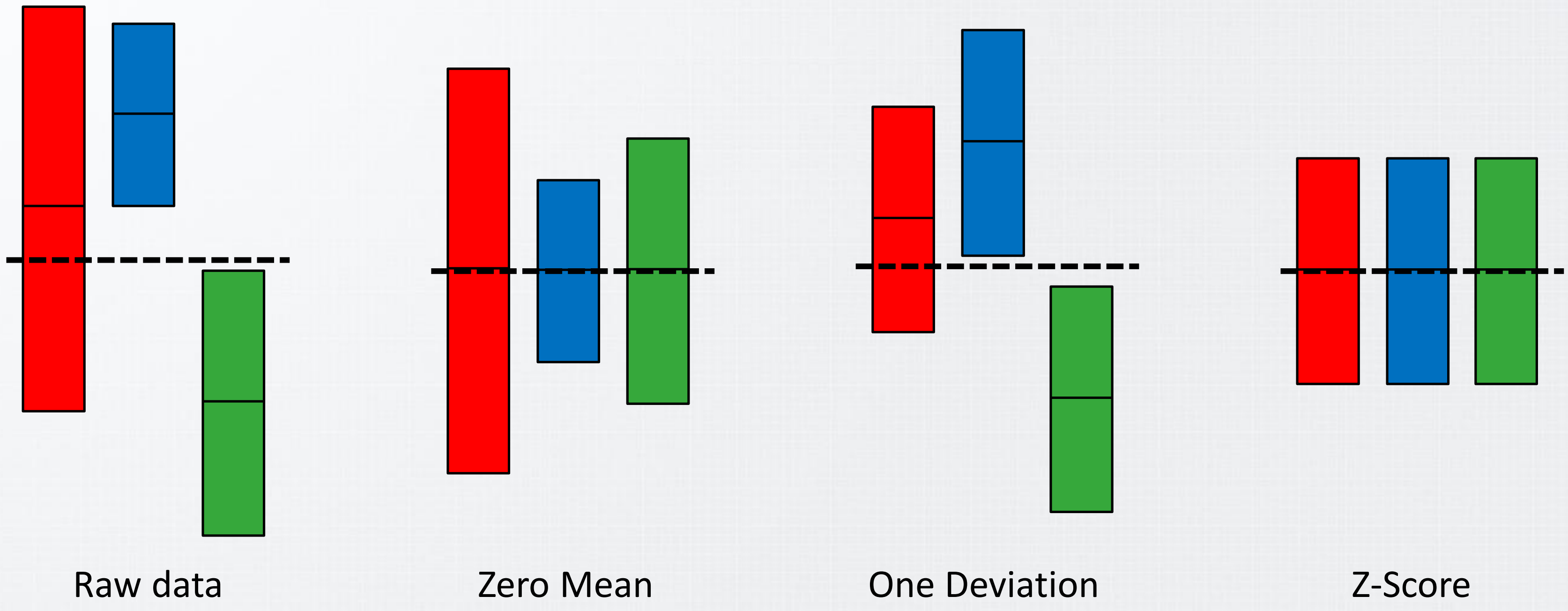


$\mu \in \mathbb{R}^{1 \times m}$
(average sample)

$\sigma \in \mathbb{R}^{1 \times m}$

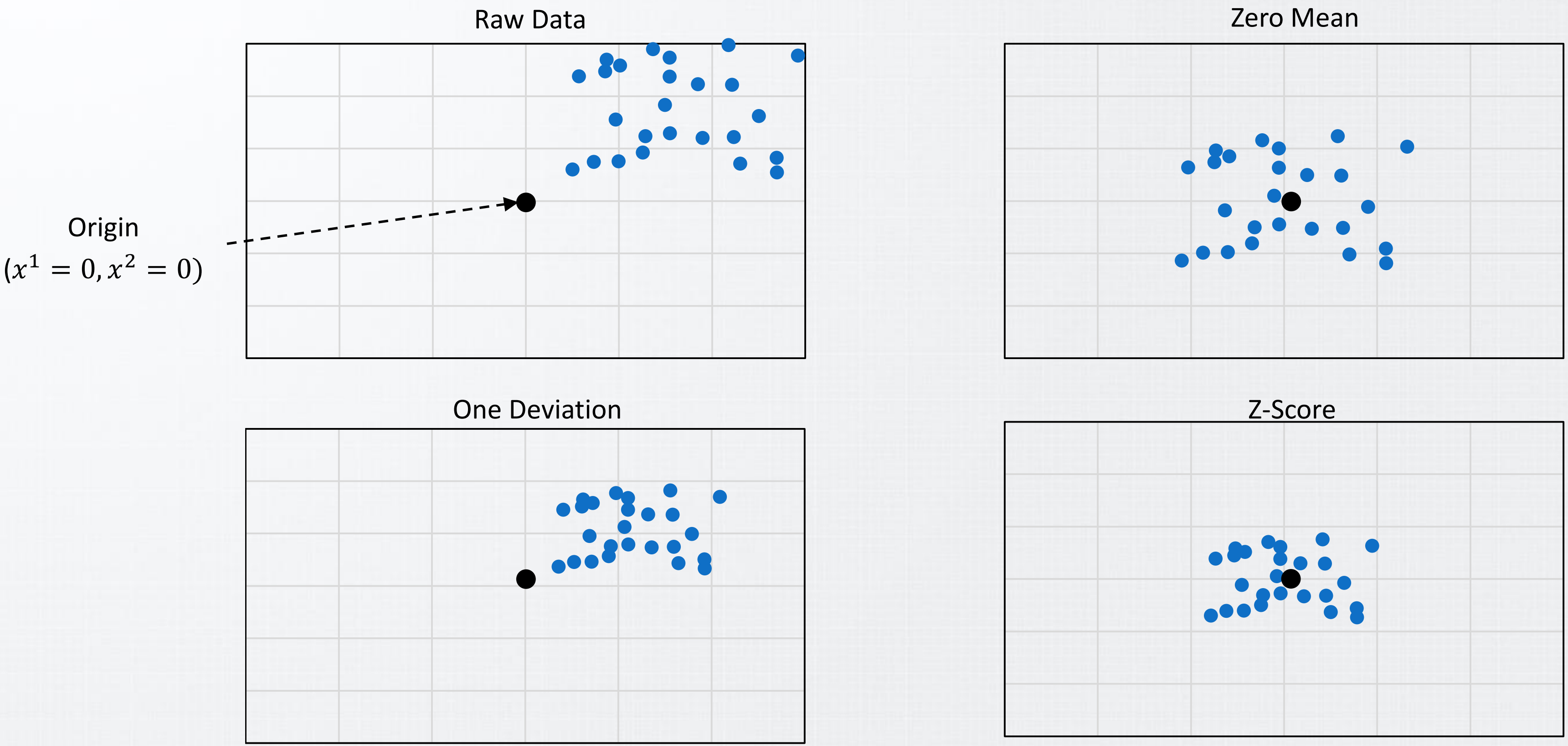
Z-Score

Normalization



Z-Score

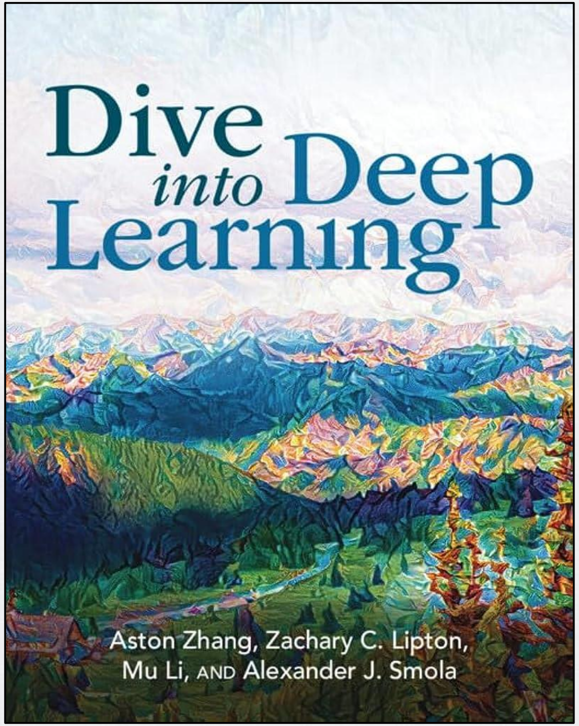
Normalization



Bibliography

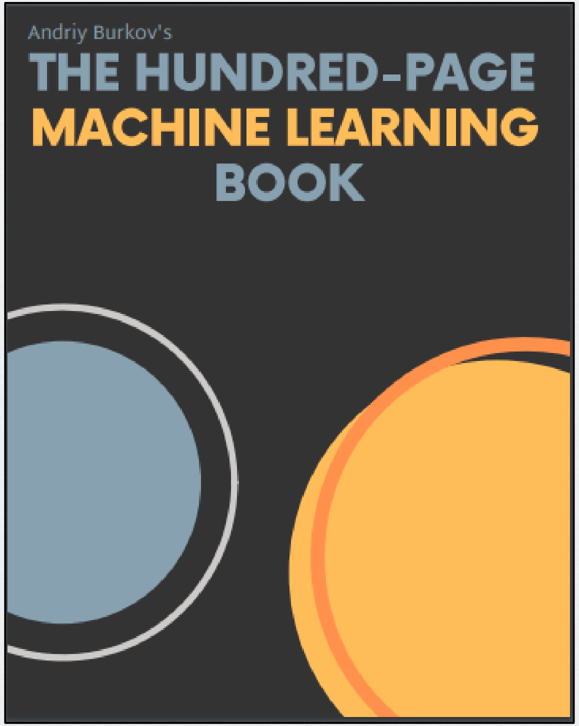
Bibliography

- Dive into Deep Learning
    - Chapter 3
        - 3.1 Linear Regression
        - 3.4 Linear Regression Implementation from Scratch



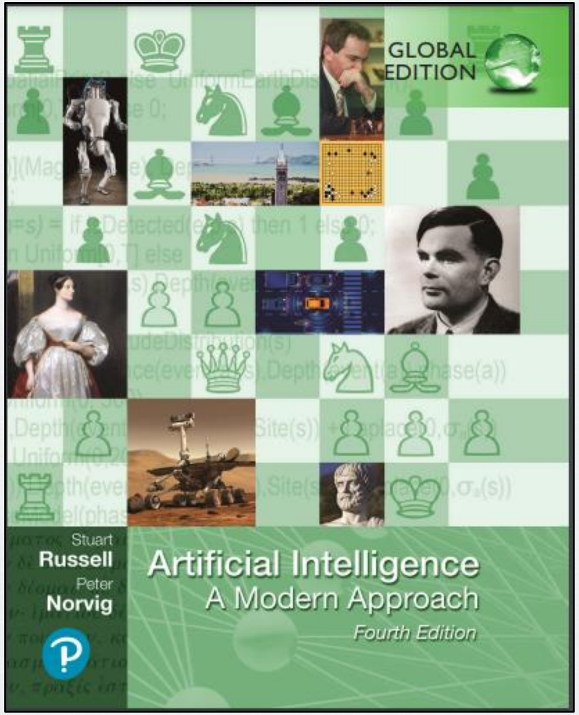
Bibliography

- The Hundred-page Machine Learning Book
  - Chapter 3 – Fundamental Algorithms
    - 3.1 Linear Regression



Bibliography

- Artificial Intelligence A Modern Approach Fourth Edition
  - Chapter 19.6 Linear Regression and Classification
    - 19.6.2 Gradient descent



Bibliography

- Deep Learning
  - Chapter 4 – Numerical Computation
    - 4.3 Gradient-Based Optimization