

Linear Regression and Gradient Descent

Prof. Mingkui Tan

South China University of Technology
Southern Artificial Intelligence Laboratory(SAIL)

December 1, 2018



Content

- 1 Basic Concepts about Machine Learning
- 2 Linear Regression
- 3 Closed-form solution
- 4 Gradient Descent

Contents

- 1 Basic Concepts about Machine Learning
- 2 Linear Regression
- 3 Closed-form solution
- 4 Gradient Descent

Basic Concepts about Machine Learning

What is Machine Learning?

Machine Learning compose of three parts:

- Data
- Model(function)
- Loss(prediction)

Basic Concepts about Machine Learning

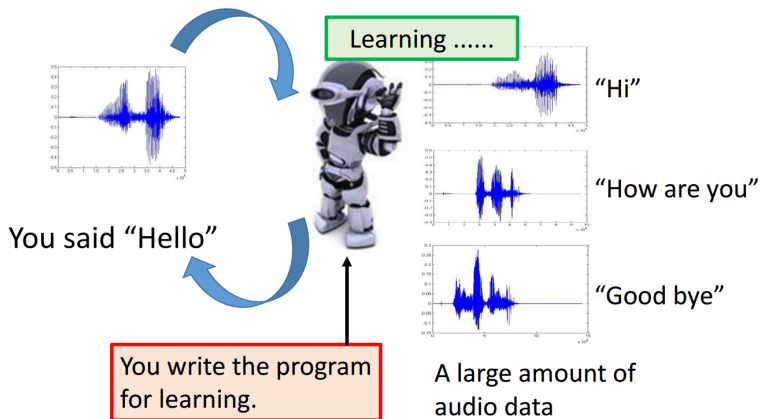


Figure: Speech Recognition

Basic Concepts about Machine Learning

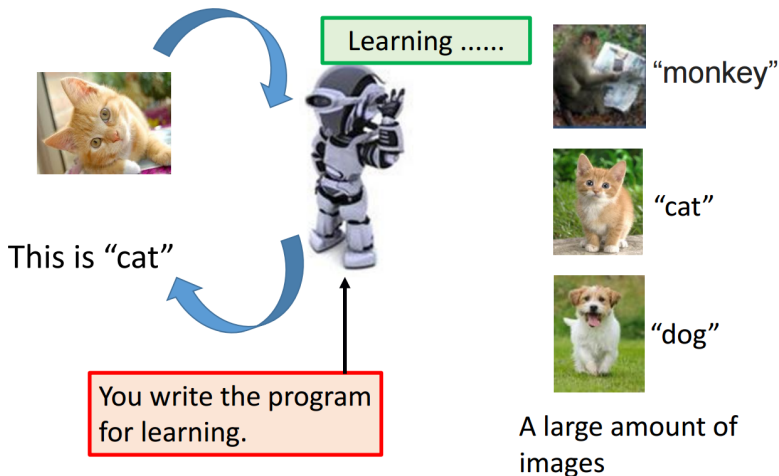


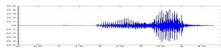
Figure: Image Recognition

Basic Concepts about Machine Learning

Machine Learning \approx Looking for a Function

- Speech Recognition

$$f(\text{audio waveform}) = \text{"How are you"}$$



- **Image Recognition**

$$f(\text{image of a cat}) = \text{"Cat"}$$



- Playing Go

$$f(\text{Go board state}) = \text{"5-5"} \quad (\text{next move})$$



- Dialogue System

$$f(\text{"Hi"} \mid \text{what the user said}) = \text{"Hello"} \quad (\text{system response})$$

Basic Concepts about Machine Learning Framework







f_1	) = "cat"	f_2	) = "money"
f_1	) = "dog"	f_2	) = "snake"

Figure: Image Recognition

Three Main Elements of Machine Learning

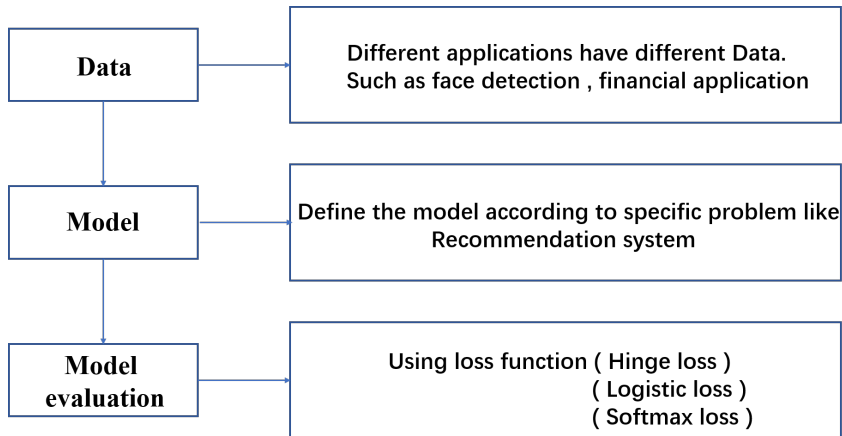


Figure: Three main elements of machine learning

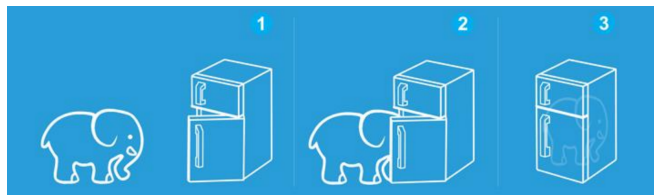
Basic Concepts about Machine Learning

Framework

Machine Learning is so simple ...



Just like putting an elephant into the fridge ...

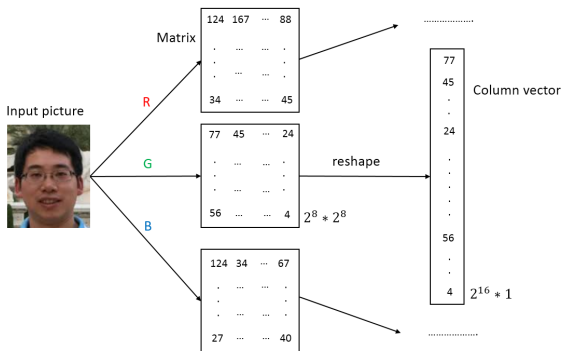


Column Vector

- Data:

$$D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$$

- \mathbf{x} is input, and we usually present it as column vector, y is output (for example: name of a person), n is number.
- For example, \mathbf{x} may be a picture stored as a matrix:



Basic Concepts about Machine Learning

- We want to use a function predicting y :

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

- However, the prediction may be inconsistent with the groundtruth. We calculate the differences by loss function:

$$\mathcal{L}_D = \sum_{i=1}^n l(\hat{y}_i, y_i)$$

Regression

Loss:

- Absolute value loss:

$$l(\hat{y}_i, y_i) = |\hat{y}_i - y_i|$$

- Least squares loss:

$$l(\hat{y}_i, y_i) = \frac{1}{2}(\hat{y}_i - y_i)^2$$

Total loss function:

$$\mathcal{L}_D(\mathbf{w}) = \sum_{i=1}^n l(\hat{y}_i, y_i)$$

Regression

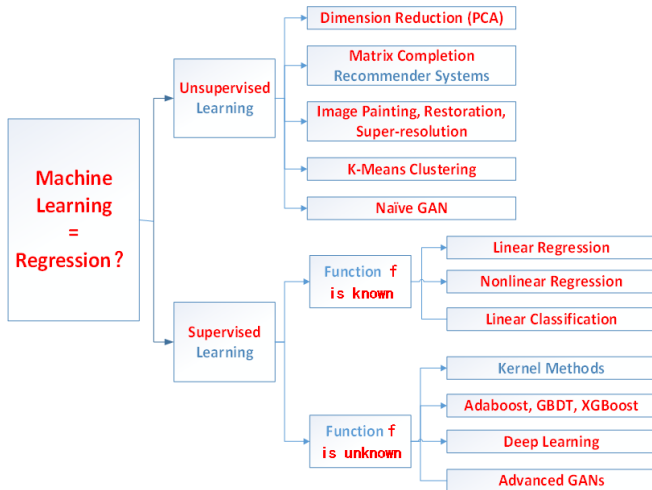
- The smaller value of \mathcal{L}_D is the better, and loss function(\mathcal{L}_D) plays a major role in machine learning

Find the best f by solving the following optimization problem:

$$f^* = \min_f \sum_{i=1}^n l(f(\mathbf{x}), y_i)$$

Supervised Optimization for Deep Learning

Learning Map



Supervised Optimization for Deep Learning

Learning Map

Supervised learning is the machine learning task of inferring a function from **labeled training data**

Labelled
data



cat



dog

Unlabeled
data



Figure: Images of cats and dogs

Data Sets for Supervised Learning

- Libsvm dataset

<http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>

LIBSVM Data: Classification, Regression, and Multi-label

This page contains many classification, regression, multi-label and string data sets stored in LIBSVM format. Many are from UCI, Statlog, StatLib and other col scale each attribute to [-1,1] or [0,1]. The testing data (if provided) is adjusted accordingly. Some training data are further separated to "training" (tr) and "vali each data set. To read data via MATLAB, you can use "libsvmread" in LIBSVM package.

A summary of all data sets is in the following. If you have used LIBSVM with these sets, and find them useful, please cite our work as:

Chih-Chung Chang and Chih-Jen Lin, LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

Please also cite the source of the data sets (references given below).

Go to pages of classification ([binary](#), [multi-class](#)), [regression](#), [multi-label](#), and [string](#). Those interested in hierarchical data with many classes can visit [LSHTC](#) pa

Some sets are large and the connection may fail. On Linux you can use

```
> wget -t inf URL_address_of_data
```

to retry infinitely many times. If it still fails, add -c to continually get a partially-downloaded set. You can also use

```
> lftp -c 'pget -c URL_address_of_data'
```

to have several connections for reducing the downloading time.

	name	source	type	class	training size	testing size	feature
a1a		UCI	classification	2	1,605	30,956	123
a2a		UCI	classification	2	2,265	30,296	123
a3a		UCI	classification	2	3,185	29,376	123

Contents

- 1 Basic Concepts about Machine Learning
- 2 Linear Regression**
- 3 Closed-form solution
- 4 Gradient Descent

Linear Regression

Simple linear regression describes the linear relationship between a variable, plotted on the x -axis, and a response variable y , plotted on the y -axis

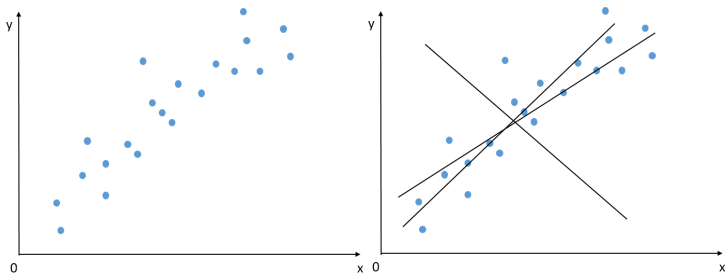


Figure: Simple linear 1D regression

Problem Setup for Regression

- Inputs
Input space: $\mathbf{X} \in \mathbb{R}^m$ features
- Outputs
Output space: \mathbf{Y}
- Goal: Learn a hypothesis/model $f : \mathbf{X} \mapsto \mathbf{Y}$

Linear Regression

Learn $f(\mathbf{x}; \mathbf{w})$ with

- Parameters: $\mathbf{w} \in \mathbb{R}^m, b \in \mathbb{R}$
- Input: \mathbf{x} where $x_j \in \mathbb{R}$ features for $j \in 1, \dots, m$
- Model Function:

$$\begin{aligned} f(\mathbf{x}; b, \mathbf{w}) &= b + w_1 x_1 + \dots + w_m x_m \\ &= \sum_{j=1}^m w_j x_j + b \\ &= \mathbf{w}^\top \mathbf{x} + b \end{aligned}$$

Linear Regression

- What makes a good model?

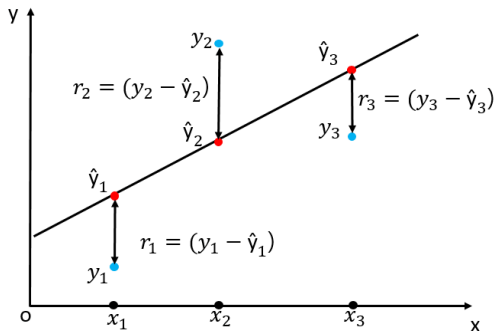


Figure: Contributing loss terms for 1D regression

Performance Measure for Regression

- Least squared loss

$$\begin{aligned}\mathcal{L}_D(\mathbf{w}) &= \frac{1}{2} \sum_{i=1}^n (y_i - f(\mathbf{x}_i; \mathbf{w}))^2 \\ &= \frac{1}{2} \sum_{i=1}^n (y_i - \hat{y}_i)^2\end{aligned}$$

Training: find minimizer of least squared loss

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})$$

Contents

- 1 Basic Concepts about Machine Learning
- 2 Linear Regression
- 3 Closed-form solution**
- 4 Gradient Descent

Matrix Presentation

$$\begin{aligned}\mathcal{L}_D(\mathbf{w}) &= \frac{1}{2} \sum_{i=1}^n (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 = \frac{1}{2} (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2\end{aligned}$$

Matrix Presentation

- Proof:

$$\begin{aligned} & \frac{1}{2} \sum_{i=1}^n (y_i - \mathbf{x}_i^\top \mathbf{w})^2 \\ &= \frac{1}{2} \begin{bmatrix} y_1 - \mathbf{x}_1^\top \mathbf{w} \\ \vdots \\ y_n - \mathbf{x}_n^\top \mathbf{w} \end{bmatrix}^\top \begin{bmatrix} y_1 - \mathbf{x}_1^\top \mathbf{w} \\ \vdots \\ y_n - \mathbf{x}_n^\top \mathbf{w} \end{bmatrix} \\ &= \frac{1}{2} \left(\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \mathbf{w} \right)^\top \left(\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \mathbf{w} \right) \\ &= \frac{1}{2} (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) \end{aligned}$$

Analytical Solution

How to address the linear regression question?

- Closed-form solution to linear regression:

$$\begin{aligned}\mathcal{L}_D(\mathbf{w}) &= \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \frac{1}{2}(\mathbf{y}^\top \mathbf{y} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w}) \\ \frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}} &= \frac{1}{2} \left(\frac{\partial \mathbf{y}^\top \mathbf{y}}{\partial \mathbf{w}} - \frac{\partial 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y}}{\partial \mathbf{w}} + \frac{\partial \mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w}}{\partial \mathbf{w}} \right) \\ &= \frac{1}{2}(-2\mathbf{X}^\top \mathbf{y} + (\mathbf{X}^\top \mathbf{X} + (\mathbf{X}^\top \mathbf{X})^\top) \mathbf{w}) \\ &= -\mathbf{X}^\top \mathbf{y} + \mathbf{X}^\top \mathbf{X} \mathbf{w}\end{aligned}$$

Analytical Solution

$$\begin{aligned}\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} &= -\mathbf{X}^\top \mathbf{y} + \mathbf{X}^\top \mathbf{X} \mathbf{w} = 0 \\ \Rightarrow \mathbf{X}^\top \mathbf{X} \mathbf{w} &= \mathbf{X}^\top \mathbf{y} \\ \Rightarrow \mathbf{w} &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}\end{aligned}$$

Solve for optimal parameters \mathbf{w}^*

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \arg \min_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})$$

Problem about The Analytical Solution

There are two questions left to address about the analytical solution $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$:

- **Many matrices** are not invertible
- The inverse of a large matrix needs huge memory, which takes $O(m^3)$ to compute

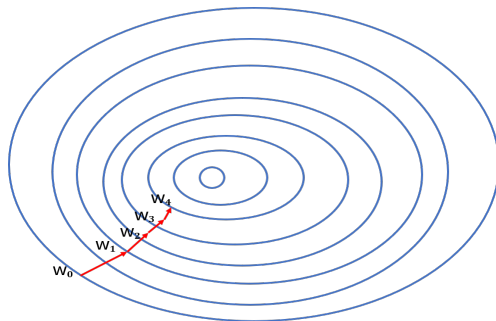
Contents

- 1 Basic Concepts about Machine Learning
- 2 Linear Regression
- 3 Closed-form solution
- 4 Gradient Descent**

Gradient Descent

- Get the best \mathbf{w} by minimizing a loss function $\mathcal{L}_D(\mathbf{w})$

$$\arg \min_{\mathbf{w}} \mathcal{L}_D(\mathbf{w})$$



Descent Direction

We use $\mathbf{d} = -\frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}}$ as the **direction** of optimization
Gradient (vector of partial derivatives)

$$\frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial \mathcal{L}_D(w_1)}{\partial w_1} \\ \frac{\partial \mathcal{L}_D(w_2)}{\partial w_2} \\ \vdots \\ \frac{\partial \mathcal{L}_D(w_m)}{\partial w_m} \end{bmatrix}$$

(We always write a vector into column form)

Why $\mathcal{L}_D(\mathbf{w}') = \mathcal{L}_D(\mathbf{w} + \eta \mathbf{d}) \leq \mathcal{L}_D(\mathbf{w})$ ($\eta \rightarrow 0$ & $\eta > 0$)

Descent Direction

Proof:

- By Taylor expansion, when $\eta \rightarrow 0$:

$$\begin{aligned}\mathcal{L}_D(\mathbf{w} + \eta \mathbf{d}) &= \mathcal{L}_D(\mathbf{w}) + \left[\frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}} \right]^\top \eta \mathbf{d} + o(\eta \mathbf{d}) \\ &= \mathcal{L}_D(\mathbf{w}) + \eta' \left[\frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}} \right]^\top \mathbf{d}\end{aligned}$$

Note that $\eta' > 0$ and

$$\eta' \left[\frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}} \right]^\top \mathbf{d} = -\eta' \mathbf{d}^\top \mathbf{d} \leq 0$$

We have:

$$\mathcal{L}_D(\mathbf{w}') = \mathcal{L}_D(\mathbf{w} + \eta \mathbf{d}) \leq \mathcal{L}_D(\mathbf{w})$$

Gradient Descent

update parameters

Minimize loss by repeated gradient steps (when no closed form):

- Compute gradient of loss with respect to parameters $\frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}}$
- Update parameters with rate η

$$\mathbf{w}' \rightarrow \mathbf{w} - \eta \frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}}$$

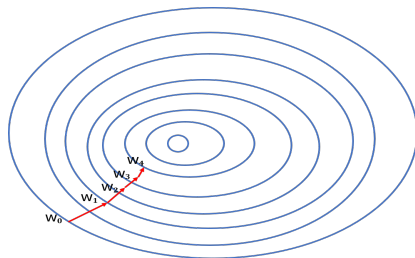


Figure: Gradient steps on a simple $m = 2$ loss function.

Learning Rate

Find out an appropriate size of step

Learning rate η has a large impact on convergence

- Too large $\eta \rightarrow$ oscillatory and may even diverge
- Too small $\eta \rightarrow$ too slow to converge

Adaptive learning rate(For example):

- Set larger learning rate at the begining
- Use relatively smaller learning rate in the later epochs
- Decrease the learning rate: $\eta^{t+1} = \frac{\eta^t}{t+1}$

THANK YOU!