Basic Concepts about Machine Learning: Regression and Optimization

Prof. Mingkui Tan

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

July 23, 2018

Content

Contents

Basic Concepts about Machine Learning

What is Machine Learning?

Mchine Learning compose of three parts:

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

Basic Concepts about Machine Learning

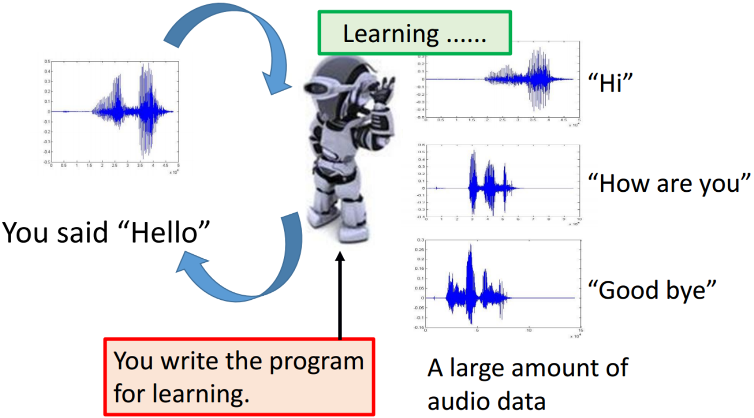


Figure: Speech Recognition

Basic Concepts about Machine Learning

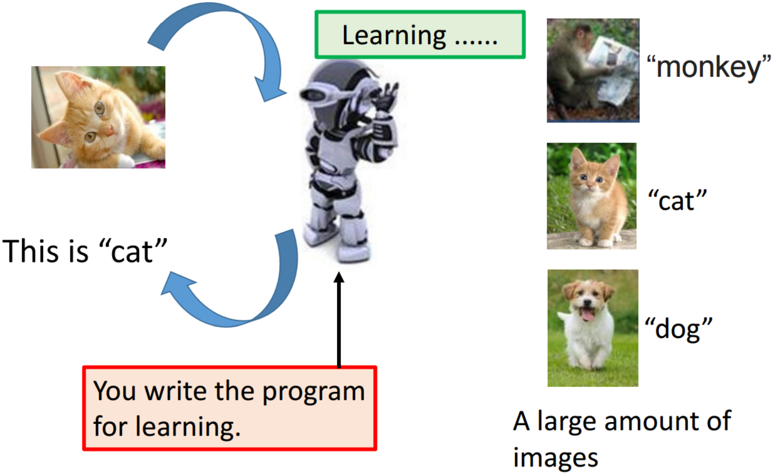
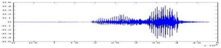


Figure: Image Recognition

Basic Concepts about Machine Learning

Machine Learning ≈ Looking for a Function

- Speech Recognition

  $f($  $)=$ "How are you"

- **Image Recognition**

  $f($  $)=$ "Cat"

- Playing Go

  $f($  $)=$ "5-5" (next move)

- Dialogue System

  $f($ "Hi" $)=$ "Hello"

  (what the user said) (system response)

Basic Concepts about Machine Learning

Framework

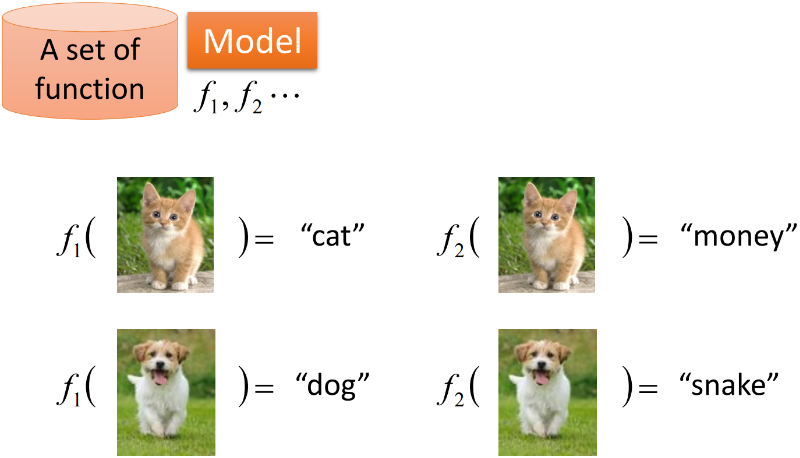


Figure: Image Recognition

Three Main Elements of Machine Learning

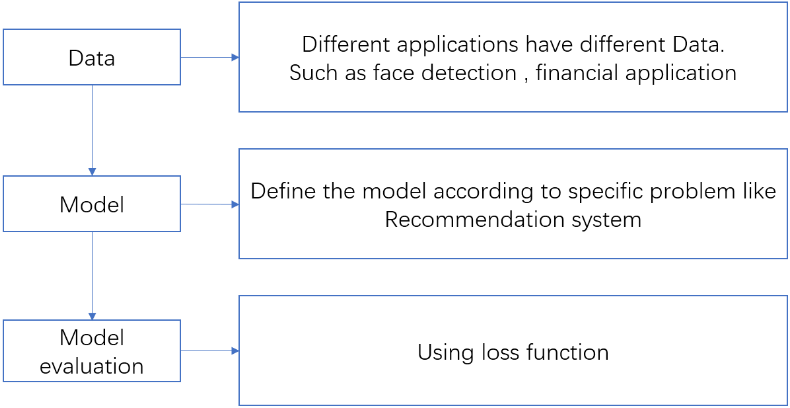


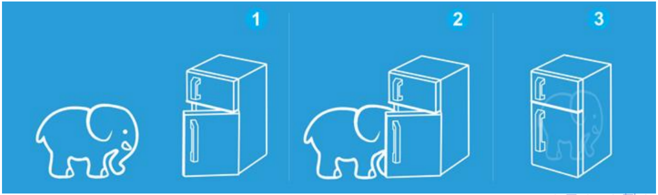
Figure: Three main elements of machine learning

Basic Concepts about Machine Learning

Framework

Machine Learning is so simple . . .

Just like putting an elephant into the fridge . . .

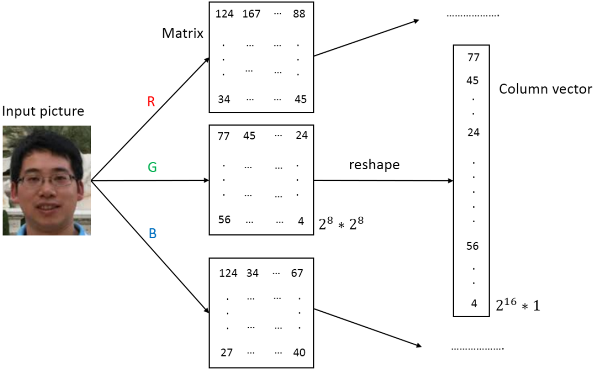


Column Vector

- Data:

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

- $\mathbf{x}$ is input,and we usually present it as column vector
- For example,$\mathbf{x}$ may be a picture stored as a matrix:



Basic Concepts about Machine Learning

- y is output (for example: name of a person)
- n is number
- 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:

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

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(loss function):

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

Regression

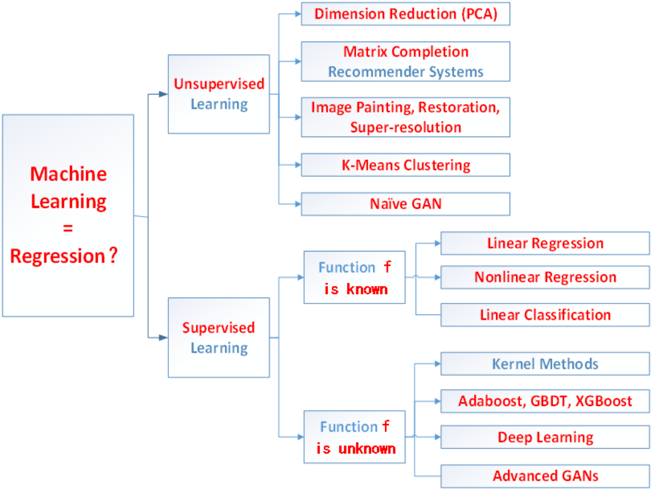
- The smaller value of $\mathcal{L}_D$ the better, and loss function($\mathcal{L}_D$) plays a major role in machine learning

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

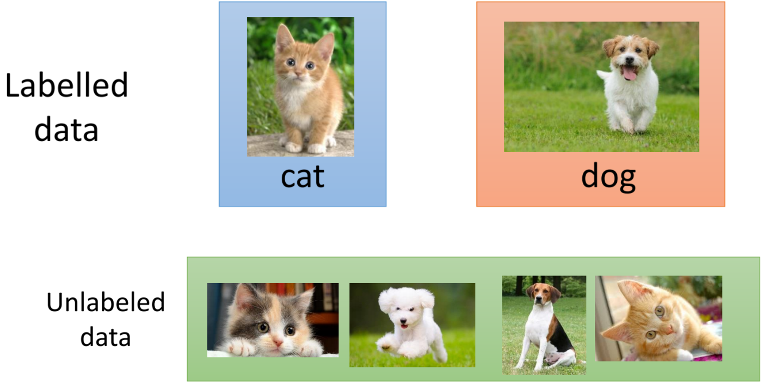


Figure: Images of cats and dogs

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 continuely 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

Linear Regression

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

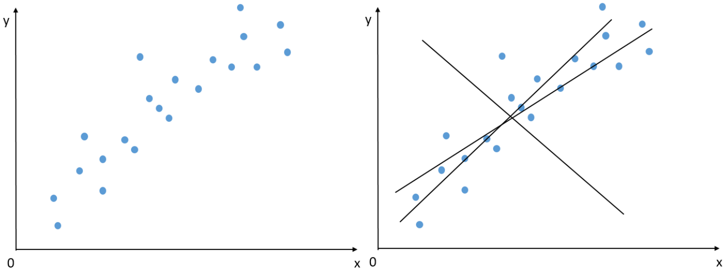


Figure: Simple linear 1D regression

Machine Learning Setup

- Inputs
  Input space $\mathbf{X} = \mathbb{R}^m$
  feature,covariants,predictors,etc.
- Outputs
  Output space:$\mathbf{Y}$
  many different types of predictions.
- Goal:Learn a hypothesis/model $f : \mathbf{X} \mapsto \mathbf{Y}$

Supervised Learning

- Given set of input,output pairs

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

- Learn the "best" model based on D
- Predict $\hat{y}$ for unseen x based on f(**x**)

Linear Regression: Common

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

- Parameters:$\mathbf{w} \in \mathbb{R}^m, w_0 \in \mathbb{R}$
- Input:$\mathbf{x}$ where $x_j \in \mathbb{R} \quad for \quad j \in 1,...m$ features
- Model Function:

$$\begin{aligned} f(\mathbf{x}; w_0, \mathbf{w}) &= w_0 + w_1 x_1 + ... + w_m x_m \\ &= \sum_{j=1}^{m} w_j x_j + w_0 \\ &= \mathbf{w}^\top \mathbf{x} + w_0 \end{aligned}$$

Linear Regression

- What makes a good model?

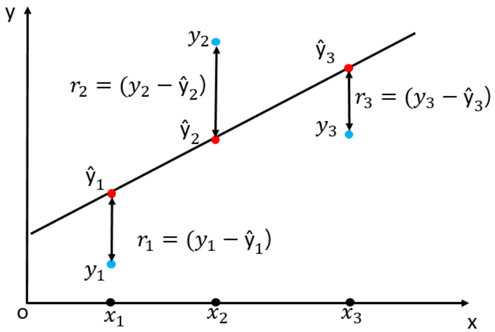


Figure: Contributing loss terms for 1D regression

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

Matrix Presentation

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

Matrix Presentation

- Proof:

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

Analytical Solution

- Closed-form of 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

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

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

Issues 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
- The inverse takes $O(m^3)$ to compute

Any solutions?

Gradient Descent!!

Contents

Regularized Least Square (RLS) Regression

Impose regularization on $\mathbf{w}$:

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

Here, $\frac{1}{2}||\mathbf{w}||_2^2$ is called **Regularizer**, λ is called **trade-off parameter** or **regularization parameter**

Training: find minimizer of least squared loss

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

Closed-form Solution for Regularized Least Square (RLS)

First-order condition of the optimal solution:

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$$

For the Least Regression problem, we have

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

We obtain the optimal $\mathbf{w}^*$ by

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$$

Issues of the Closed-form Solution

Closed-form solution: $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$:

- The inverse of a large matrix needs huge memory
- The inverse takes $O(m^3)$ complexity to compute

Contents

Machine Learning

Training Procedure

- Identify a set of hypotheses $f(\mathbf{x}; \mathbf{w})$
- Define a loss criterion $\mathcal{L}_D$
- Pick the best $\mathbf{w}^*$ by minimizing a loss function $\mathcal{L}_D(\mathbf{w})$

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

Learning is done through optimization

Main Tool: Gradients

Typical case (with possibly parameterized g)

$$\boldsymbol{\mathcal{L}}_D(\mathbf{w}) : \mathbb{R}^n \mapsto \mathbb{R}$$

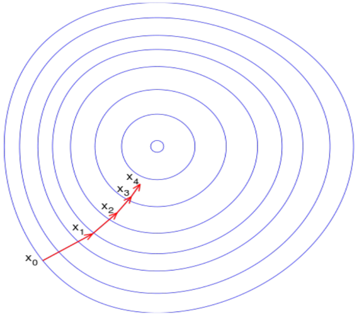
Gradient (vector of partial derivatives)

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

(We will always write as column vectors)

Descent Direction

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



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

Descent Direction

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

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

We have:

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

Note that $\eta > 0$ and

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

Gradient Descent

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

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

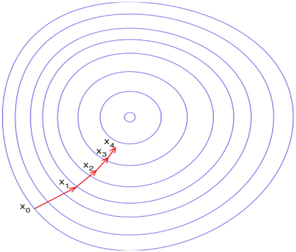


Figure: Gradient steps on a simple m = 2 loss function.

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!