UNIVERSITY
OF OSLO



IN4310 Linear Models

Ali Ramezani-Kebrya

ali@uio.no



Learning goals

Basic components of machine learning problems

Loss functions: quadratic loss, 0-1 loss, hinge loss

Empirical risk minimization

An overview on first-order optimization methods

Outline

What is Machine (Deep) Learning? [1, 2]

Discriminative Machine Learning Problem and ERM

Linear Regression

Classification with Linear Model and SVM

First-order Optimization Methods

Machine Learning (ML) and Its Components

Learning: converting experience into expertise or knowledge

ML: automated learning of meaningful patterns in data

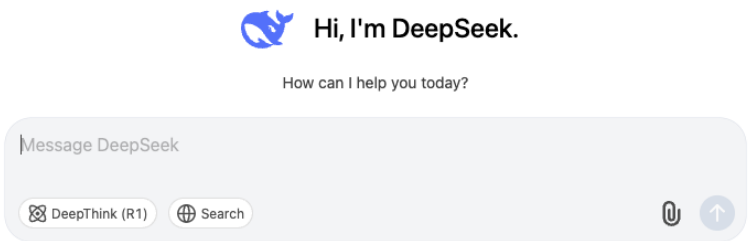
In ML, our goal is to label unseen data (generalization matters)

Main components of ML systems:

- Dataset
- Model (hypothesis class)
- Learning algorithm

[Shalev-Shwartz & Ben-David, 2014]

Deep Learning Revolution





DALL-E

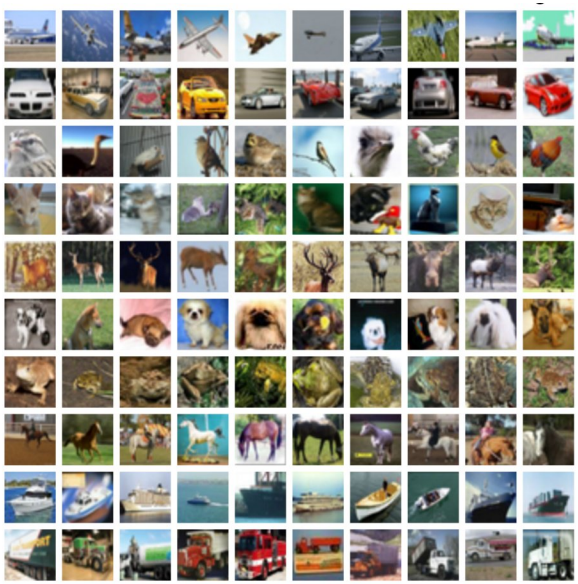


Shutterstock by Scharfsinn

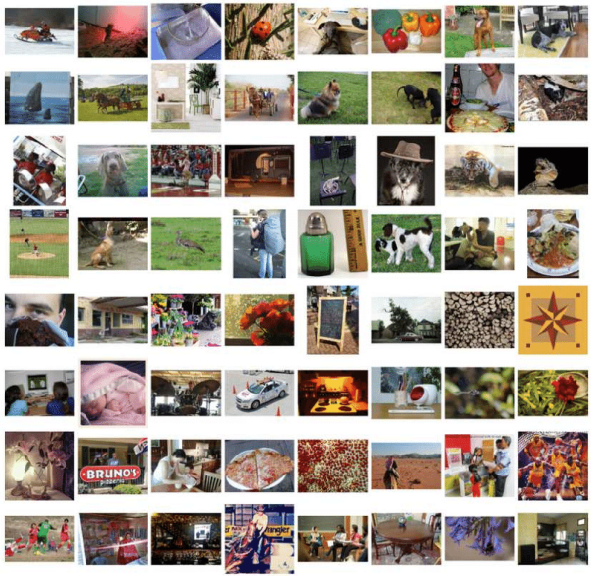


SIMA, DeepMind

Supervised Learning: Multi-class Image Classification



CIFAR-10: 60000 32x32 colored images (3 channels) from 10 classes [Krizhevsky, 2009]



ImageNet: 14 million colored images from 21K classes [Deng et al., 201]

Goal: For an image-label pair $(\mathbf{x}, y)$ drawn from an unknown distribution, find a classifier $h \in \mathcal{H}$ with minimum misclassification probability

$$\min_{h \in \mathcal{H}} \Pr(h(\mathbf{x}) \neq y)$$

Outline

What is Machine (Deep) Learning? [1, 2]

Discriminative Machine Learning Problem and ERM

Linear Regression

Classification with Linear Model and SVM

First-order Optimization Methods

General Discriminative ML

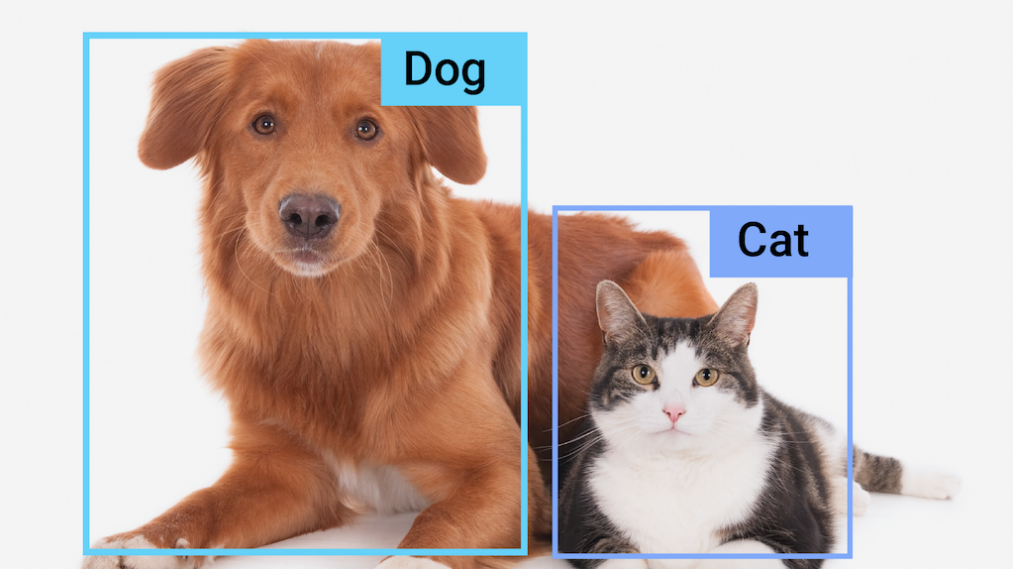
Predictor/classifier $h_{\mathbf{w}}$: Input space $\mathcal{X} \rightarrow$ Output space $\mathcal{Y}$

Model has trainable parameters $\mathbf{w}$ (weights and biases of a neural network)

Use (feature,label) training data and a loss function to optimize $\mathbf{w}$

Prediction is done with a certain misclassification probability

Detection in Images: Class Label and Position



Google AI for developers

Input: image

Output: a number of bounding boxes

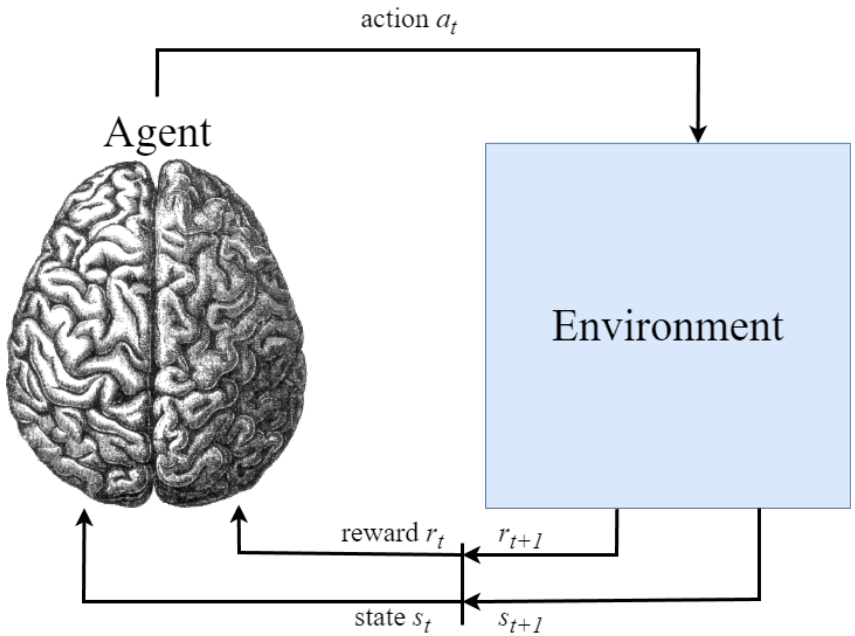
Text Summarization, Key Word Assignment

```
SPOCK: He was here. That's the great thing to me.
It's a most the course of course. I have been the
computer of the death. That many time between the
area of here
YeR I'm Computer to the activation to treat of the
logic. That's a dead. Captain Kirk.
SPOCK: There is a man and the truth.
```

Input: sequence of words $(w_1, \ldots, w_K)$
Output: sequence of words $(w_1, \ldots, w_L)$
"The text is about Star Trek. It mentions Spock and James T. Kirk."

Input: sequence of words $(w_1, \ldots, w_K)$
Output: set of words $\{w_1, \ldots, w_L\}$
Multi-label classification, possibly sequential outputs from an RNN
"Star Trek"

Reinforcement Learning



Input: sequence of states

Output: next action (move left, move right, ...)

Visual Question Answering and Image Captioning



Input: Image (+ sequence of words for VQA)

Output: set of words, sequential outputs from an RNN

How to define a discriminative machine learning problem?

Components of an ML Problem

Basic components of an ML problem

I/O: specify input space and output space

Prediction model: define a class of prediction models

Loss: measure discrepancy of model prediction versus ground truth

Optimizer: define an algorithm for updating model parameters using (labelled) training data / for finding a good predictor

What input and output space can be used for the examples above?

Predictor and Loss

Predictor/classifier $h_{\mathbf{w}}$

$h_{\mathbf{w}}(\mathbf{x})$ is a good approximation of the ground truth label $\mathbf{y} \in \mathcal{Y}$

$\ell(h_{\mathbf{w}}(\mathbf{x}_i), \mathbf{y}_i)$ is the loss on sample i, i.e., $(\mathbf{x}_i, \mathbf{y}_i)$

Empirical Risk Minimization (ERM)

Let $h_{\mathbf{w}} : \mathcal{X} \to \mathcal{Y}$ denote a predictor parameterized by $\mathbf{w} \in \mathbb{R}^d$. ERM:

$$\min_{\mathbf{w}} \left\{ R_n(\mathbf{w}) := \frac{1}{n} \sum_{i=1}^{n} \ell(h_{\mathbf{w}}(\mathbf{x}_i), \mathbf{y}_i) \right\}$$

where $\ell(h_{\mathbf{w}}(\mathbf{x}_i), \mathbf{y}_i)$ is the loss on sample i, i.e., $(\mathbf{x}_i, \mathbf{y}_i)$.

Some frequently used loss functions ℓ:

Logistic loss: $\ell(h_{\mathbf{w}}(\mathbf{x}), \mathbf{y}) = \log(1 + \exp(-\mathbf{y} \cdot h_{\mathbf{w}}(\mathbf{x})))$

Quadratic loss: $\ell(h_{\mathbf{w}}(\mathbf{x}), \mathbf{y}) = \|\mathbf{y} - h_{\mathbf{w}}(\mathbf{x})\|_2^2$

Hinge loss: $\ell(h_{\mathbf{w}}(\mathbf{x}), \mathbf{y}) = \max(0, 1 - \mathbf{y} \cdot h_{\mathbf{w}}(\mathbf{x}))$

Outline

What is Machine (Deep) Learning? [1, 2]

Discriminative Machine Learning Problem and ERM

Linear Regression

Classification with Linear Model and SVM

First-order Optimization Methods

I/O for Linear Regression

A. `http://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set`

X_1 transaction date
X_2 house age (unit: year)
X_3 distance to the nearest MRT station (unit: meter)
X_4 # convenience stores in the living circle on foot (integer)
X_5 geographic coordinate, latitude. (unit: degree)
X_6 geographic coordinate, longitude. (unit: degree)

Output Y: house price per unit of area

Input and Output Space in Regression Problems

`http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime`

`http://archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test`

`https://en.wikipedia.org/wiki/Concrete_slump_test`

`http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset`

Typically $\mathcal{X} \subseteq \mathbb{R}^d$

Single regression target: $\mathcal{Y} \subseteq \mathbb{R}$

Multiple regression targets: $\mathcal{Y} \subseteq \mathbb{R}^k$

Predictor for Linear Regression

Linear function without/with a bias

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} = \sum_{i=1}^{d} x_i w_i, \mathbf{w} \in \mathbb{R}^d$$

$$h_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b = \sum_{i=1}^{d} x_i w_i + b, \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

A weighted sum of features x_i

Loss Function for Regression

Loss function to measure quality of prediction for a data sample which is a pair $(\mathbf{x}, \mathbf{y})$: quadratic loss, squared ℓ_2 loss, squared Euclidean distance

$$\ell(h_{\mathbf{w}}(\mathbf{x}), \mathbf{y}) = \|h_{\mathbf{w}}(\mathbf{x}) - \mathbf{y}\|_2^2$$

Properties:

$\ell(h_{\mathbf{w}}(\mathbf{x}), \mathbf{y}) = 0 \Leftrightarrow h_{\mathbf{w}}(\mathbf{x}) = \mathbf{y}$

Symmetry: capture deviations on both sides of ground truth $\mathbf{y}$

Simple derivative

Optimizer for Regression

Skip this step for now. First consider:

What does the linear mapping represent?

What is the goal of optimization?

What is a good linear mapping given data?

How to use linear mapping for classification?

Binary Classification with Linear Model

Linear function without/with a bias

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} = \sum_{i=1}^{d} x_i w_i, \mathbf{w} \in \mathbb{R}^d$$

$$h_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b = \sum_{i=1}^{d} x_i w_i + b, \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

$$f_{\mathbf{w},b}(\mathbf{x}) = \mathrm{sgn}(h_{\mathbf{w},b}(\mathbf{x})) \in \{-1, +1\}$$

0-1 Loss Function for Classification

0-1 loss to measure classifier's performance for a pair $(\mathbf{x}, y)$:

$$\ell(f_{\mathbf{w}}(\mathbf{x}), y) = \mathbb{1}[f_{\mathbf{w}}(\mathbf{x}) \neq y]$$

Counts mispredictions

Non-differentiable with uninformative derivative

Outline

What is Machine (Deep) Learning? [1, 2]

Discriminative Machine Learning Problem and ERM

Linear Regression

Classification with Linear Model and SVM

First-order Optimization Methods

Setup for Classification

Goal: predict a class for an input $\mathbf{x} \in \mathcal{X}$



Pixabay user Loggawiggler

Which sort ? Fly agaric ? Death cap ? Chestnut bolete?

Output space $\mathcal{Y}$ is discrete and finite

Binary classification $\mathcal{Y} = \{0, 1\}$ or $\{-1, 1\}$

C-class classification $\mathcal{Y} = \{0, 1, \ldots, C-1\}$

Loss Function for Classification

Binary prediction by $f_{\mathbf{w}}(\mathbf{x}) > 0\,?\, -1 : +1$

0-1 loss

$$\ell(h_{\mathbf{w}}(\mathbf{x}), y) = \mathbb{1}[\mathrm{sgn}(h_{\mathbf{w}}(\mathbf{x})) \neq y]$$

Problem: sgn is unsuitable for gradient optimization

For $y \in \{-1, +1\}$, rewrite $\mathbb{1}[\mathrm{sgn}(h_{\mathbf{w}}(\mathbf{x})) \neq y]$ as $\mathbb{1}[y h_{\mathbf{w}}(\mathbf{x}) < 0]$ without sgn

$$y h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} > 0 & \text{if } h_{\mathbf{w}}(\mathbf{x}) > 0, y > 0 \text{ or } h_{\mathbf{w}}(\mathbf{x}) < 0, y < 0 \text{ no error} \\ < 0 & \text{if } h_{\mathbf{w}}(\mathbf{x}) > 0, y < 0 \text{ or } h_{\mathbf{w}}(\mathbf{x}) < 0, y > 0 \text{ error} \end{cases}$$

Loss Function for Classification

Hinge loss

$$\ell(h_{\mathbf{w}}(\mathbf{x}), y) = \max\left(0, 1 - y h_{\mathbf{w}}(\mathbf{x})\right)$$

An upper bound on 0-1 loss (insight: fix $y = +1$. Then plot $\mathbb{1}[zy < 0]$ and $\max(0, 1 - zy)$)

If the hinge loss is small, 0-1 loss must be small

Minimize empirical risk using hinge loss $\mathbf{w}^\star = \arg\min_{\mathbf{w}} R_n(\mathbf{w})$ where

$$R_n(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \max\left(0, 1 - y_i h_{\mathbf{w}}(\mathbf{x}_i)\right)$$

Then predict using $f_{\mathbf{w}^\star}$

Support Vector Machines (SVM)

Add a quadratic regularization term on the weights to the overall risk

$$\arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \max\left(0, 1 - y_i h_{\mathbf{w}}(\mathbf{x}_i)\right) + \lambda \|\mathbf{w}\|_2^2$$

SVM:

Averaged hinge loss $\max\left(0, 1 - y_i h_{\mathbf{w}}(\mathbf{x}_i)\right)$

A linear/affine model $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i$

Quadratic regularization on the weights $\lambda \|\mathbf{w}\|^2$

A different perspective: $\frac{1}{\|\mathbf{w}\|_2}$ is the margin to be maximized

What makes the difference whether we have a classification or a regression problem?

Outline

What is Machine (Deep) Learning? [1, 2]

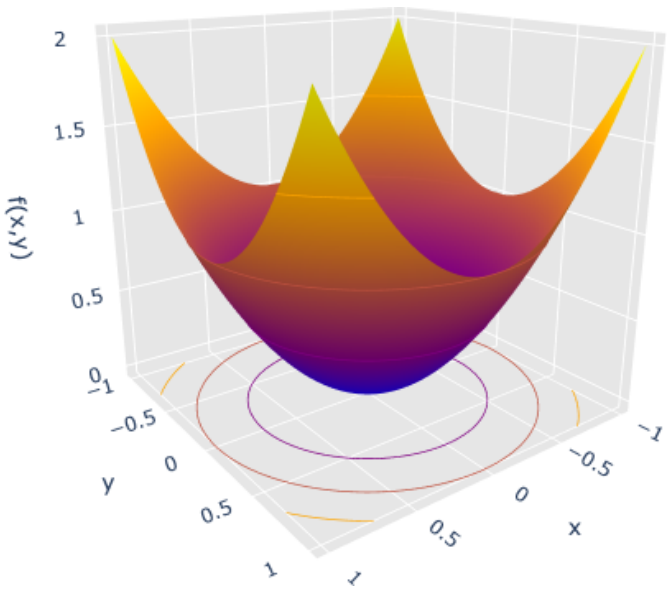
Discriminative Machine Learning Problem and ERM

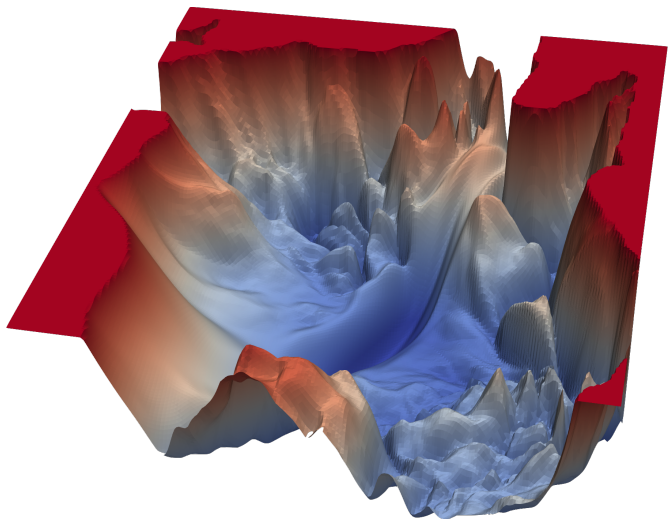
Linear Regression

Classification with Linear Model and SVM

First-order Optimization Methods

Landscape of ERM for Deep Neural Networks





Convex (left) vs. non-convex (right) optimization landscape

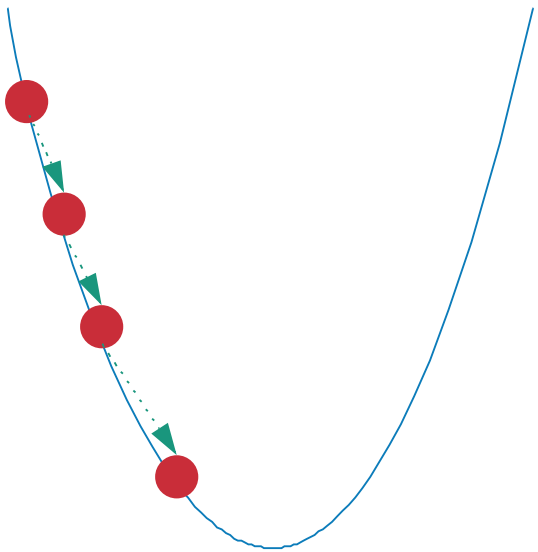
Gradient Descent for ERM

$\min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w})$ where $f : \mathbb{R}^d \to \mathbb{R}$

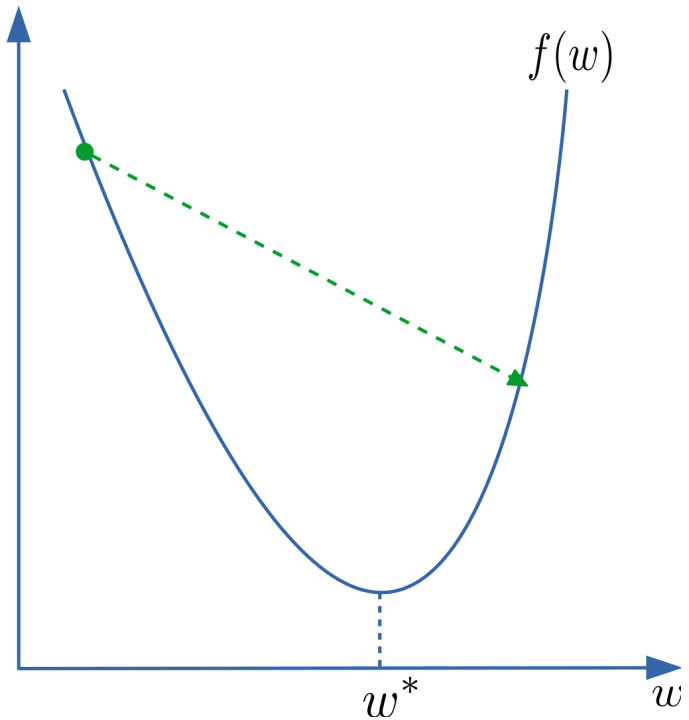
Sometimes no analytical solution: numerical solution

Assume $f(\mathbf{w})$ is differentiable

Consider a one-dimensional convex function $f(w)$



Gradient Descent: Intuition



Let w^* achieve the minimum of $f(w)$

If $w = w^* \rightarrow f'(w) = 0$

If $w > w^* \rightarrow f'(w) > 0$, i.e., $f(w)$ is increasing

If $w < w^* \rightarrow f'(w) < 0$, i.e., $f(w)$ is decreasing

Gradient Descent for ERM

Initialize $\mathbf{w}_0$

For $t = 0, 1, 2, \cdots$

Compute $\nabla_{\mathbf{w}} R_n(\mathbf{w}_t)$

Update $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t \nabla_{\mathbf{w}} R_n(\mathbf{w}_t)$

Continue until $\|\nabla_{\mathbf{w}} R_n(\mathbf{w}_t)\| \approx 0$

α: step size "learning rate"

Large step size: output error could be large

Small step size: slow convergence

Stochastic Gradient Descent for ERM

GD Update rule: $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \frac{\alpha_k}{n} \sum_{j=1}^{n} \nabla_{\mathbf{w}} \ell_j(\mathbf{w}_t)$

Computational complexity per update: $\mathcal{O}(nd)$

n can be very large

Stochastic Gradient Descent:

For $t = 0, 1, 2, \cdots$

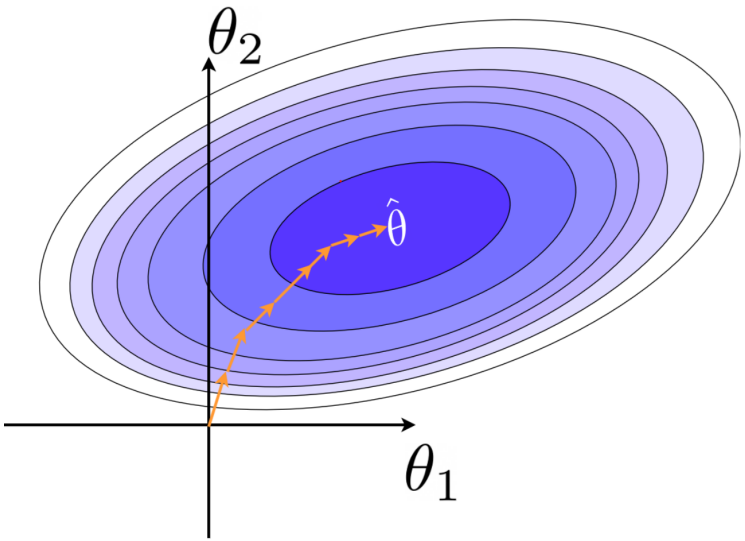
Select $j \in \{1, \cdots, n\}$ uniformly at random

$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla_{\mathbf{w}} \ell_j(\mathbf{w}_k)$

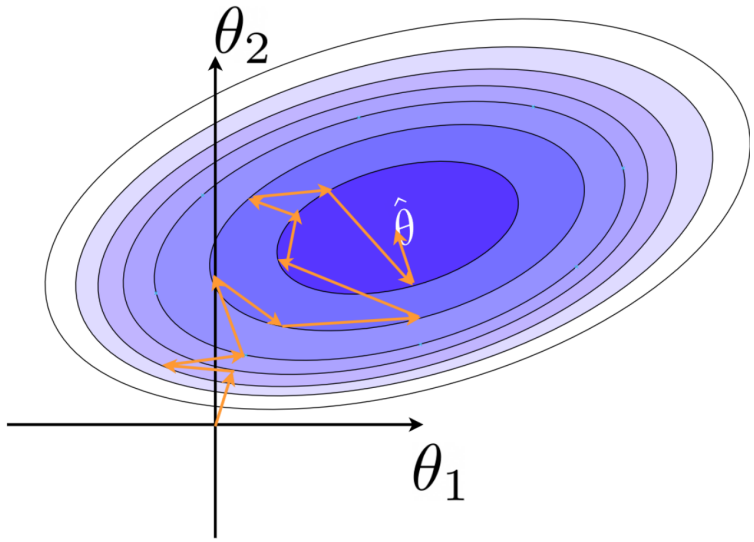
Computational complexity per update: $\mathcal{O}(d)$

Requires more iterations

GD vs. SGD



Gradient Descent



Stochastic Gradient Descent

Closed-form Solution to Solve Linear Regression

Next: Recap on one method to find parameters in linear regression

Setting: no bias. Parameters are $\mathbf{w}$.

A First Method to Solve Linear Regression

Goal: to find parameters which minimizes empirical risk:

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2$$

A given dataset $\mathcal{S}^{\text{tr}} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
Then $f_{\mathbf{w}^\star}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}^\star$ is the selected mapping
Rare case: there exists a closed-form solution

Write the empirical risk R_n as function of $\mathbf{w}$
Compute $\nabla R_n(\mathbf{w})$ - the gradient with respect to $\mathbf{w}$
Solve $\nabla R_n(\mathbf{w}) = 0$ for $\mathbf{w}$
Solution can be maximum, minimum, or saddle point
Verify that the Hessian in the solution point is positive definite
Function has positive curvature in every direction

A First Method to Solve Linear Regression

Write in matrix form:

$$X = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} = \begin{pmatrix} x_{1,1}, \ldots, x_{1,d} \\ \vdots \\ x_{n,1}, \ldots, x_{n,d} \end{pmatrix} \in \mathbb{R}^{n \times d}$$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Then:

$$R_n(\mathbf{w}) = \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2 = (X\mathbf{w} - \mathbf{y})^\top \cdot (X\mathbf{w} - \mathbf{y})$$

A First Method to Solve Linear Regression

Compute $\nabla R_n(\mathbf{w})$ and set it to zero

$$\begin{aligned}
\nabla_{\mathbf{w}}(X\mathbf{w}-\mathbf{y})^\top \cdot (X\mathbf{w}-\mathbf{y}) &= 2X^\top \cdot (X\mathbf{w}-\mathbf{y}) \\
2X^\top(X\mathbf{w}-\mathbf{y}) &= 0 \\
\Rightarrow (X^\top X)\mathbf{w} &= X^\top \mathbf{y} \\
\mathbf{w} &= (X^\top X)^{-1}X^\top \cdot \mathbf{y}
\end{aligned}$$

if the matrix inverse $(X^\top X)^{-1}$ exists

A First Method to Solve Linear Regression

Closed-form solution to linear regression without bias

Let X be the training data matrix

$$X = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} = \begin{pmatrix} x_{1,1}, \ldots, x_{1,d} \\ \vdots \\ x_{n,1}, \ldots, x_{n,d} \end{pmatrix} \in \mathbb{R}^{n \times d}, \ \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^{n \times 1}$$

The model is $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$. A solution is given as

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

if the matrix inverse $(X^\top X)^{-1}$ exists.

Prediction is done using: $f_{\mathbf{w}^\star}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}^\star$

A First Method to Solve Linear Regression

How to extend this solution to the case with a bias?

$$\mathbf{x}^\top = \Big(x_1, \ldots, x_d\Big) \rightarrow \hat{\mathbf{x}}^\top = \Big(x_1, \ldots, x_d, 1\Big)$$

Then the parameter $\mathbf{w}$ also gets an additional dimension

$$\hat{\mathbf{w}}^\top = \Big(w_1, \ldots, w_d, w_{d+1}\Big)$$

Then:

$$\hat{\mathbf{x}}^\top \hat{\mathbf{w}} = \mathbf{x}^\top \mathbf{w} + b$$

w_{d+1} acts as bias

From Linear to Ridge regression

From Linear to Ridge Regression

Overfitting: low training error and high test error

Reason: fit to the noise in the training data

One solution: add a regularization and avoid weights $\mathbf{w}$ getting too large:

$$\arg\min_{\mathbf{w}} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

From Linear to Ridge Regression

Ridge regression

Ridge regression is Linear regression with ℓ_2 regularization

$$\lambda \|\mathbf{w}\|^2$$

λ is a hyperparameter. The solution changes to

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

In practice, one needs to find a good value for the hyperparameter λ on a validation set, before measuring the performance on the test set. The effect of this regularization will be discussed later.

From Linear to Ridge Regression

Advantages of ridge regression over linear regression without regularization:

For any $\lambda > 0$ a solution always exists: $(X^\top X + \lambda \mathbf{I})$ is always invertible because it is positive definite

The Hessian is positive definite, thus one finds always a minimum

References

[1] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning from data*. AMLBook New York, 2012.

[2] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.