Comp6211e: Optimization for Machine Learning

Tong Zhang

Lecture 1: Machine Learning and Optimization

Background of Machine Learning

- More efficient data gathering and generation process.
  - Digitalization of offline data (EB)
  - PC and Mobile Internet data (ZB)
  - Sensor networks, Internet of Things, Clouds (YB)
- High storage capability
- Increased computational power
  - multi-core CPU
  - GPU
  - distributed computing

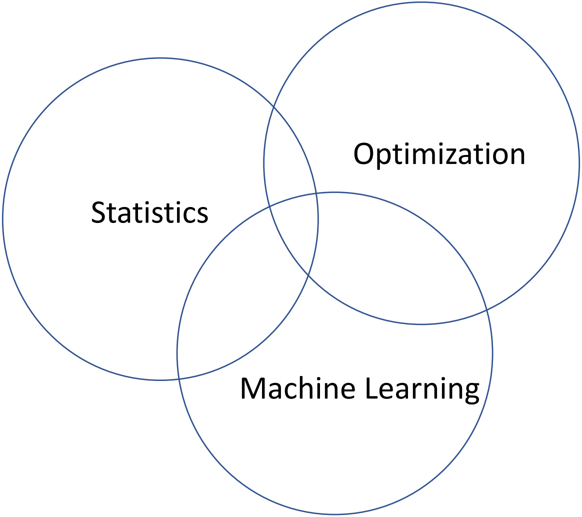
Machine Learning

- Learn from Data (Experience)
  - Statistics + Computing
- Example:
  - how to recognize objects in an image (supervised learning: classification)
  - how to generate a human face (unsupervised: generative model)
  - how to play poker (reinforcement learning: games)
- Active research field
  - Intersection of statistics, computer science, optimization, control, engineering (pattern recognition) and so on.

Steps in Machine Learning

- Define the problem (what to learn?)
- Data preparation (put data in electronic form)
  - data cleaning (remove abnormal data)
  - representation (put in a form data mining algorithm can understand)
- Learning
  - feature selection/filtering
  - statistical model
  - training: learn model parameter from data
- Interpretation/validation

Machine Learning, Statistics, Optimization



Supervised learning

- Prediction problem
  - Input X: known information.
  - Output Y: unknown information.
  - Goal: to predict Y based on X
- Observe historical data $(X_1, Y_1), \ldots, (X_n, Y_n)$
- Learning:
  - find prediction function f that maps X to the corresponding Y.
  - model: $f(w, x)$ with paramter w: relate X to Y
  - training: learn w that fits the training data by optimizing a creterion.

Machine Learning Pipeline

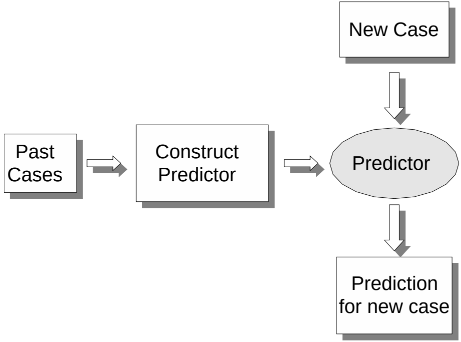


Figure: Predicting the Future Based on the Past

- past cases: training data (known input and known output)
- new cases: test data (known input and unknown output)
- training/learning (construct predictor): optimization

Example: stock price prediction

- Known information: the current stock market status
- Goal: predict Coca-Cola's stock price one week from now.
- Training data: historic stock price of Coca-Cola and market status one week before.
- Encode market status into input feature vector
  - find predictive patterns:
    - Coca-Cola's current stock price and trend is relevant
    - Pepsi's current stock price and trend is relevant
    - Microsoft's current stock price and trend is less relevant
  - validation of patterns: expert knowledge and regression test.

Optimization Problem: Least Squares Regression

Optimization Problem (training):

$$\min_{\beta} \sum_{i=1}^{n} (X_i^\top \beta - Y_i)^2$$

Linear Least Squares Regression

- $i = 1, \ldots, n$: historic data
- X_i: encoding historic information (features)
- β model parameters
- Y_i: target (stock price to be predicted)

Banknote Example

Four measurements on 100 genuine Swiss banknotes and 100 counterfeit ones:

$x[1]$ length of the bill (in mm),

$x[2]$ width of the left edge (in mm),

$x[3]$ width of the right edge (in mm),

$x[4]$ bottom margin width (in mm).

...

Response variable y: status of the banknote [-1 for genuine and 1 for counterfeit]

Probabilistic model that predicts counterfeiting based on the four measurements

Statistical Model for Binary Classification

Binary response: $Y_i \in \{\pm 1\}$:

$$Y_i | X_i \sim \mathit{Bernoulli}(\mu_i), \ell(\mu_i) = \beta^\top X_i,$$

where

$$P(Y_i = 1) = \mu_i \qquad P(Y_i = -1) = 1 - \mu_i.$$

Logistic regression model.
Link function is *logit transform* on probability of success

$$\ell(\mu_i) = \log(\mu_i/(1 - \mu_i)) = \beta^T X_i \,,$$

with

$$\mu_i = \frac{1}{1 + \exp(-\beta^T X_i)}$$

Binary Classification: Linear Logistic Regression

Likelihood

$$\prod_{i=1}^{n} P(Y_i|X_i) = \prod_{i=1}^{n} \left(\frac{1}{1 + \exp(-Y_i X_i^T \beta)} \right)$$

- Maximum Likelihood Estimate (MLE) is logistic regression:

$$\min_{\beta} \sum_{i=1}^{n} \ln(1 + \exp(-Y_i X_i^T \beta))$$

- Convex optimization problem in terms of β (easy to solve)

Multiclass Classification: Linear Logistic Regression

- Class $1, \ldots, K$, $\beta = [\beta_1, \ldots, \beta_k]$ $(\beta_k \in R^p)$

?

:

dog

cat

horse

monkey

- Multi-class conditional probability model:

$$P(Y = c|X) = \frac{\exp(\beta_c^T X)}{\sum_{\ell=1}^K \exp(\beta_\ell^T X)}.$$

- Maximum Likelihood Estimate:

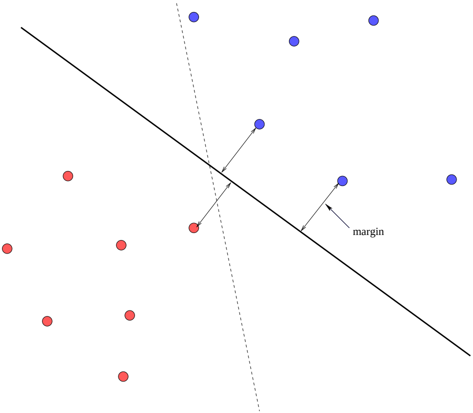
$$\hat{\beta} = \arg\max_{\beta} \sum_{i=1}^n \ln \frac{\exp(\beta_{Y_i}^T X_i)}{\sum_{\ell=1}^k \exp(\beta_\ell^T X_i)}$$

Multi-class Classification: Neural Networks

- Class $1, \ldots, K$
- Input data $X \in R^d$
- Label $Y \in \{1, \ldots, K\}$
- w: neural network parameters, with function $f(w, \cdot) : R^d \to R^K$
- Maximum Likelihood estimate:

$$\max_w \sum_{i=1}^{n} \ln \frac{\exp(f_{Y_i}(w, X_i))}{\sum_{\ell=1}^{k} \exp(f_{Y_\ell}(w, X_i))}$$

Support Vector Machine: Linear Separator



Linear separator is defined by the separating hyperplane with weight w

$$\{x : w^\top x = 0\}$$

Optimal Separating Hyperplane

Given training data $(X_i, Y_i) \in R^d \times \{\pm 1\}$ for $(i = 1, \ldots, n)$. We want to find $w \in R^d$ to optimize margin

$$\max_w \gamma(w),$$

where margin is defined as

$$\gamma(w) = \frac{\min_i w^T X_i Y_i}{\|w\|_2 \sup_i \|X_i\|_2}$$

Equivalent formulation: constrained convex optimization problem:

$$\min_w \|w\|_2^2$$
$$\text{subject to } w^\top X_i Y_i \geq 1 \quad (i = 1, \ldots, n)$$

Regularization

Regularized Least Squares (ridge regression)

$$\min_{\beta} \left[\sum_{i=1}^{n} (X_i^\top \beta - Y_i)^2 + \frac{\lambda}{2} \|\beta\|_2^2 \right]$$

Regularized logistic regression:

$$\min_{\beta} \left[\sum_{i=1}^{n} \ln(1 + \exp(-Y_i X_i^T \beta)) + \frac{\lambda}{2} \|\beta\|_2^2 \right]$$

Soft-margin SVM:

$$\min_{\beta} \left[\sum_{i=1}^{n} (1 - Y_i X_i^T \beta)_+ + \frac{\lambda}{2} \|\beta\|_2^2 \right]$$

Lasso: non-smooth regularization

$$\min_{\beta} \left[\sum_{i=1}^{n} (X_i^\top \beta - Y_i)^2 + \lambda \|\beta\|_1 \right]$$

The solution β is sparse

- only a small number of β_j are non-zeros.

The L_1 regularizer is non-smooth.

We need **proximal gradient algorithms** to solve such problems.

Matrix Completion

Assume we observe part of a matrix X as Y_{ij}

						…
Alice	1	?	?	4	?	
Bob	?	2	5	?	?	
Carol	?	?	4	5	?	
Dave	5	?	?	?	4	
⋮						

Want to reconstruct the full matrix X:

$$\sum_{\text{observed } (i,j)} (X_{i,j} - Y_{i,j})^2 + \lambda \|X\|_{\text{trace}}.$$

Trace-norm of a matrix is the sum of its singular values.

Need **proximal gradient** methods to solve.

Optimization for Supervised Learning

In training, we try to optimize

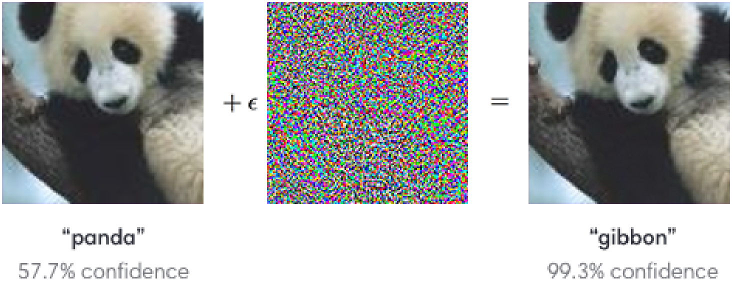
$$\min_w \sum_{i=1}^n L(f(w, X_i), Y_i) + R(w)$$

- Training data: (X_i, Y_i).
- Prediction function: $f(w, x)$, e.g., linear classifier or neural networks
- Model parameter w: to be learned from data
- $L(f, y)$: loss function such as least squares or logistic regression
- $R(w)$: regularizer

This is an optimization problem with finite sum structure

$$\min_w \phi(w) + R(w) \qquad \phi(w) = \frac{1}{n} \sum_{i=1}^n \phi_i(w)$$

Adversarial Example



Adversarial Defense Training

In normal training, we try to optimize

$$\min_w \sum_{i=1}^{n} L(f(w, X_i), Y_i)$$

In adversarial training, we allow a preturbation $\tilde{X}_i$ of each data point X_i:

$$\min_w \max_{\{\tilde{X}_i : \|\tilde{X}_i - X_i\| \leq \epsilon, i=1,\ldots,n\}} \sum_{i=1}^{n} L(f(w, \tilde{X}_i), Y_i)$$

This becomes a **saddle point (minimax) problem.**

Another example of saddle point problem: generative adversarial network (GAN)

Balck Box Adversarial Attack

Given input X, and an unknown classification model $f(X)$, with label Y.

Given a different Label $Y' \neq Y$, find perturbation δ such that

$$\min_{\delta:\|\delta\|\leq\epsilon} L(f(X+\delta)), Y'),$$

where $L(\cdot,\cdot)$ is a loss function.

For each x, we can evaluate $f(x)$ but not its derivative $\nabla f(x)$.

- **Derivative free optimization**
- For each δ, we may not compute the derivative with respect to δ

Constrained Formulation

The distributed optimization problem with m nodes, and training loss $\phi_j(w)$ on node j:

$$\min_w \phi(w) \qquad \phi(w) = \sum_{j=1}^m \phi_j(w)$$

is equivalent to

$$\min_w \sum_{j=1}^m \phi_j(w_j), \qquad w = w_1 = w_2 = \cdots = w_m$$

which becomes **constrained optimization.**

Each node has its own parameter, and we require them to be the same.
In addition to computation, we need communication to syncronize among nodes. This leads to computation/communication tradeoff.

Summary: Machine Learning and Optimization

- Finite sum problem: (gradient and stochastic gradient)

$$\min_w \sum_{i=1}^{n} \phi_i(w) + R(w),$$

  where regularizer $R(w)$ may be **non-smooth (proximal gradient).**
- **Minimax** Problem (Saddle Point):

$$\min_w \max_v \phi(w, v)$$

- **Constrained** Problem:

$$\min_w \sum_{j=1}^{m} \phi_j(w_j) \qquad w_j = w \text{ for } j = 1, \ldots, m.$$

- **Derivative Free** Optimization:

$$\min_w \phi(w),$$

  where we can evaluate $\phi(w)$ but not $\nabla\phi(w)$.