Convex Optimization

Lecture 6 - Applications in Machine Learning

Instructor: Yuanzhang Xiao

University of Hawaii at Manoa

Fall 2017

Today's Lecture

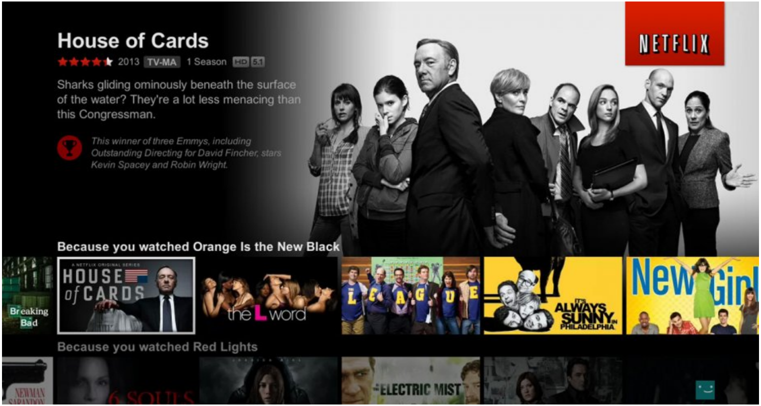
1. Regression / Prediction
2. Classification
3. Unsupervised Learning
4. Reinforcement Learning

Outline

1. Regression / Prediction
2. Classification
3. Unsupervised Learning
4. Reinforcement Learning

Supervised Learning - Examples

Netflix recommendation systems:



Supervised Learning

basic elements:

- $a^{(i)} \in \mathbb{R}^n$: features
  - gender, occupation, income, zip code
- $b^{(i)} \in \mathbb{R}$: target
  - ratings of the movie, watched a movie or not
- $\{(a^{(i)}, b^{(i)}) \,|\, i = 1, \ldots, m\}$: training set

find hypothesis $h : a \mapsto b$

- b continuous (e.g., rating): regression
- b discrete (e.g., watched or not): classification

Linear Regression

linear hypothesis:

$$h_x(a) = x_1 a_1 + \cdots + x_n a_n$$

where x are weights

choose x to minimize cost function:

$$f_0(x) : \mathbb{R}^n \to \mathbb{R}$$

many choices of different cost functions $\to$ different methods

Ordinary Least Squares Regression

ordinary least squares regression:

$$\text{minimize} \quad f_0(x) = \|Ax - b\|_2$$

where

$$A = \begin{bmatrix} a^{(1)T} \\ \vdots \\ a^{(m)T} \end{bmatrix}, \quad b = \begin{bmatrix} b^{(1)} \\ \vdots \\ b^{(m)} \end{bmatrix}$$

- convex optimization problem
- normal equation

$$A^T A x = A^T b$$

- if $\text{rank} A = n$, we have

$$x^\star = \left(A^T A\right)^{-1} A^T b$$

Chebyshev Regression

Chebyshev regression:

$$\text{minimize} \quad f_0(x) = \|Ax - b\|_\infty$$

where $\|y\|_\infty = \max\{|y_1|, \ldots, |y_m|\}$

- convex optimization problem (may be hard to solve)
- equivalent formulation: a linear program

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & -t\mathbf{1} \leq Ax - b \leq t\mathbf{1} \end{array}$$

with optimization variables x and $t \in \mathbb{R}$

Sum of Absolute Residuals Regression

sum of absolute residuals regression:

$$\text{minimize} \quad f_0(x) = \|Ax - b\|_1$$

where $\|y\|_1 = \sum_{i=1}^{m} |y_i|$

- convex optimization problem (may be hard to solve)
- equivalent formulation: a linear program

$$\begin{array}{ll} \text{minimize} & 1^T t \\ \text{subject to} & -t \leq Ax - b \leq t \end{array}$$

with optimization variables x and $t \in \mathbb{R}^m$

Regularized Regression

regularized regression:

$$\text{minimize} \quad f_0(x) = \|Ax - b\| + \gamma \|x\|$$

- forces x to be small
- less sensitive to errors in features A
- select few important features

- convex optimization problem (may be hard to solve)
- $\|Ax - b\|_1 + \gamma \|x\|_1$ can be reformulated as LP
- $\|Ax - b\|_2 + \gamma \|x\|_1$ can be reformulated as QP

Regression Regularized by Cardinality

regression regularized by cardinality:

$$\text{minimize} \quad f_0(x) = \|Ax - b\| + \operatorname{card}(x)$$

where $\operatorname{card}(x)$ is the number of nonzero elements of x

properties of $\operatorname{card}(\cdot)$:

- quasiconcave on $\mathbb{R}^n_+$:

$$\operatorname{card}(x + y) \geq \min\{\operatorname{card}(x), \operatorname{card}(y)\}$$

- non-convex

General Convex-Cardinality Problems

convex-cardinality problem: one that would be convex except for the appearance of $\mathrm{card}(\cdot)$ in objective or constraints

examples:

- regression regularized by cardinality
- minimum cardinality:

$$\begin{array}{ll} \text{minimize} & \mathrm{card}(x) \\ \text{subject to} & \|Ax - b\|_2 \leq \epsilon \end{array}$$

- sparse modeling / regressor selection:

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|_2 \\ \text{subject to} & \mathrm{card}(x) \leq k \end{array}$$

NP-hard problems

Solutions to General Convex-Cardinality Problems

exact solutions:

- fix sparsity pattern (which elements are nonzero), then solve the resulting convex problem
- 2^n convex problems to solve

convex heuristics:

- replace $\mathrm{card}(x)$ with $\|x\|_1$

example: minimize $\|Ax - b\|_2$ subject to $\mathrm{card}(x) \leq k$

- solve minimize $\|Ax - b\|_2 + \gamma \mathrm{card}(x)$
- adjust γ so that $\mathrm{card}(x) \leq k$
- fix sparsity pattern, and solve the resulting convex problem

theoretical guarantee that heuristic is exact for some problems

Outline

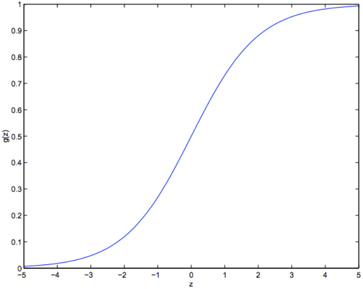
Logistic Regression

classification: $b \in \{0, 1\}$

logistic / sigmoid function:

$$h_x(a) = \frac{1}{1 + e^{-x^T a}}$$

illustration of $g(z) = \frac{1}{1+e^{-z}}$:



normalize $x^T a$ to $[0, 1]$

Logistic Regression as Convex Optimization Problem

assume that

$$\begin{aligned} \text{prob}\left(b = 1 | a; x\right) &= h_x(a) \\ \text{prob}\left(b = 0 | a; x\right) &= 1 - h_x(a) \end{aligned}$$

which can be written compactly as

$$\text{prob}\left(b | a; x\right) = \left[h_x(a)\right]^b \left[1 - h_x(a)\right]^{1-b}$$

assume training examples are independent, then likelihood of x is

$$L(x) = \text{prob}(b | A; x) = \prod_{i=1}^{m} \left[h_x(a^{(i)})\right]^{b^{(i)}} \left[1 - h_x(a^{(i)})\right]^{1-b^{(i)}}$$

easier to maximize the log-likelihood:

$$\ell(x) = \sum_{i=1}^{m} b^{(i)} \log \left[h_x(a^{(i)})\right] + \left(1 - b^{(i)}\right) \log \left[1 - h_x(a^{(i)})\right]$$

Logistic Regression as Convex Optimization Problem

rearrange the expressions:

$$
\begin{aligned}
b^{(i)} \log\left[h_x(a^{(i)})\right] &= -b^{(i)} \log\left(1 + e^{-a^{(i)T}x}\right) \\
\left(1 - b^{(i)}\right) \log\left[1 - h_x(a^{(i)})\right] &= \left(1 - b^{(i)}\right) \cdot \\
&\quad \left[\log\left(e^{-a^{(i)T}x}\right) - \log\left(1 + e^{-a^{(i)T}x}\right)\right]
\end{aligned}
$$

log-likelihood:

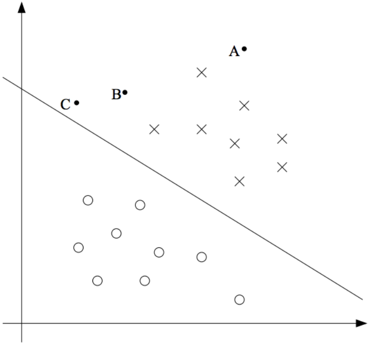
$$
\ell(x) = \sum_{i=1}^{m} -\log\left(1 + e^{-a^{(i)T}x}\right) - \left(1 - b^{(i)}\right)\left(a^{(i)T}x\right)
$$

the resulting convex optimization problem:

$$
\text{maximize} \quad \ell(x)
$$

Support Vector Machine (SVM)

a geometric view of classification problem:



- separating hyperplane: $a^T x + y = 0$
- point A: high confidence about the output
- point C: low confidence about the output

Support Vector Machine (SVM)

assume $b \in \{-1, 1\}$ (note the difference from logistic regression)
hypothesis function:

$$h_{x,y}(a) = g\left(a^T x + y\right)$$

where $g(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$

distance between separating hyperplane $a^T x + y = 0$ and the ith sample point $a^{(i)}$:

$$\gamma_i = b^{(i)} \left(a^{(i)T} x + y\right) / \|x\|_2$$

maximize the distance from the closest sample point: (nonconvex)

$$\begin{array}{ll} \text{maximize} & \gamma \\ \text{subject to} & \dfrac{b^{(i)} \left(a^{(i)T} x + y\right)}{\|x\|_2} \geq \gamma, \;\; i = 1, \ldots, m \end{array}$$

SVM as Convex Optimization

equivalent formulation: (still nonconvex)

$$\begin{array}{ll} \text{maximize} & \gamma \\ \text{subject to} & b^{(i)}\left(a^{(i)T}x + y\right) \geq \gamma, \;\; i = 1, \ldots, m \\ & \|x\|_2 = 1 \end{array}$$

convex reformulation:

$$\begin{array}{ll} \text{maximize} & \gamma \\ \text{subject to} & b^{(i)}\left(a^{(i)T}x + y\right) \geq \gamma, \;\; i = 1, \ldots, m \\ & \|x\|_2 \leq 1 \end{array}$$

more commonly-used convex reformulation:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|x\|_2^2 \\ \text{subject to} & b^{(i)}\left(a^{(i)T}x + y\right) \geq 1, \;\; i = 1, \ldots, m \end{array}$$

Dual Problem of SVM

Lagrangian:

$$L(x, y, \lambda) = \frac{1}{2}\|x\|_2^2 - \sum_{i=1}^{m} \lambda_i \left[b^{(i)} \left(a^{(i)T} x + y \right) - 1 \right]$$

dual function:

$$g(\lambda) = \inf_{x,y} \quad \frac{1}{2} x^T x - \left(\sum_{i=1}^{m} \lambda_i b^{(i)} a^{(i)} \right)^T x - \left(\sum_{i=1}^{m} \lambda_i b^{(i)} \right) y + \sum_{i=1}^{m} \lambda_i$$

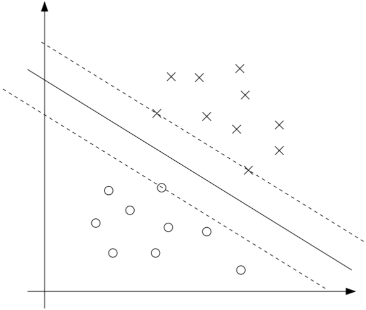
$$\Rightarrow x^*(\lambda) = \textstyle\sum_{i=1}^{m} \lambda_i b^{(i)} a^{(i)} = A^T \mathrm{diag}(b)\lambda, \quad \textstyle\sum_{i=1}^{m} \lambda_i b^{(i)} = 0$$

dual problem:

$$\begin{array}{ll} \text{maximize} & -\dfrac{1}{2}\lambda^T \mathrm{diag}(b)^T A A^T \mathrm{diag}(b)\lambda + 1^T \lambda \\ \text{subject to} & \lambda \geq 0 \\ & \displaystyle\sum_{i=1}^{m} \lambda_i b^{(i)} = 0 \end{array}$$

Support Vectors

support vectors: (the data where the inequality is binding)



a lot fewer support vectors than training data points

Solving SVM Efficiently

new training data $\left(a^{(m+1)}, b^{(m+1)}\right)$ coming in:

- if $b^{(m+1)}\left(a^{(m+1)T}x^\star + y^\star\right) > 1$, no need to update $x^\star$ and $y^\star$

if data is separable $\rightarrow$ strong duality $\rightarrow$ complementary slackness:

$$\lambda_i^\star\left[b^{(i)}\left(a^{(i)T}x^\star + y^\star\right) - 1\right] = 0$$

$\Rightarrow \lambda_i^\star > 0$ only if $a^{(i)}$ is a support vector

- sparsity in the solution to the dual problem
- $x^\star = \sum_{i:\ a^{(i)}\ \text{is a support vector}} \lambda_i^\star b^{(i)} a^{(i)}$
- classification:

$$a^T x^\star + y^\star = \sum_{i:\ a^{(i)}\ \text{is a support vector}} \lambda_i^\star b^{(i)}\left(a^T a^{(i)}\right) + y^\star$$

Outline

1. Regression / Prediction
2. Classification
3. Unsupervised Learning
4. Reinforcement Learning

Unsupervised Learning

basic elements:

- $a^{(i)} \in \mathbb{R}^n$: features
- no label
- $\left\{ \left(a^{(i)}\right) | i = 1, \ldots, m \right\}$: training set

divide the data into k clusters

common approach:

- initially, "guess" k labels (i.e., centers of clusters)
- iterate between the following two steps:
  1. assign data to different clusters
  2. update the centers of clusters

k-Means Clustering

k-means clustering:

- randomly select k cluster centroids $\mu_1, \dots, \mu_k \in \mathbb{R}^n$
- iterate between the following two steps:
  1. assign data to different clusters: for each $j = 1, \dots, k$,

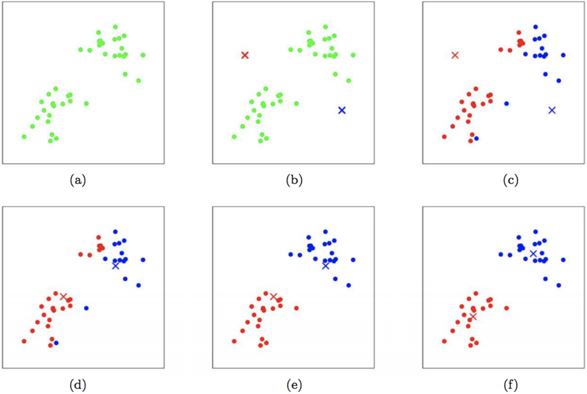
$$\mathcal{C}_j = \left\{ i : \|a^{(i)} - \mu_j\| \le \|a^{(i)} - \mu_{j'}\|, \ \forall j' \right\}$$

  2. update the centers of clusters: for each $j = 1, \dots, k$,

$$\text{minimize} \quad \sum_{i \in \mathcal{C}_j} \|a^{(i)} - \mu_j\|$$

  with optimization variable μ_j

Illustration of *k*-Means Clustering



- (a), (b): data and initial guess
- (c), (d): first iteration
- (e), (f): second iteration

Outline

1. Regression / Prediction
2. Classification
3. Unsupervised Learning
4. Reinforcement Learning

Reinforcement Learning

autonomous driving

gaming (e.g., AlphaGo)

robotics

Markov Decision Process

basic elements:

- $s \in S$: states
  - traffic, GPS position, lanes, distances from other cars, etc.
- $a \in A$: actions
  - route, speed, lane switching, etc.
- $P(s'; s, a)$: state transition probabilities
- $r(s, a)$: rewards
  - time spent, "safety", etc.
- $\delta \in (0, 1)$: discount factor

find a policy $\pi : S \to A$ to maximize the total reward:

$$\mathbb{E}_\pi \left\{ r(s_0, a_0) + \delta r(s_1, a_1) + \delta^2 r(s_2, a_2) + \cdots \right\}$$

sometimes state transition probabilities and rewards are unknown

Bellman Equation

value function:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi \left\{ r\left(s_0 = s, \pi(s)\right) + \delta r\left(s_1, \pi(s_1)\right) + \delta^2 r\left(s_2, \pi(s_2)\right) + \cdots \right\} \\ &= \underbrace{r\left(s, \pi(s)\right)}_{\text{current reward}} + \underbrace{\delta \sum_{s' \in S} P\left(s'; s, \pi(s)\right) V^\pi(s')}_{\text{expected future reward}} \end{aligned}$$

optimal value function:

$$V^\star(s) = \max_\pi V^\pi(s)$$

Bell equation: optimal value functions satisfy

$$V^\star(s) = \max_{a \in A} r\left(s, a\right) + \delta \sum_{s' \in S} P\left(s'; s, a\right) V^\star(s')$$

given optimal value functions, easy to find optimal policy

Solving Bellman Equation as Linear Program

from Bellman equation, the optimal values satisfy

$$V^\star(s) \geq r(s,a) + \delta \sum_{s' \in S} P\left(s'; s, a\right) V^\star(s'), \ \forall a \in A$$

LP formulation of Bellman equation:

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{s \in S} V(s) \\ \text{subject to} & V(s) \geq r(s,a) + \delta \displaystyle\sum_{s' \in S} P\left(s'; s, a\right) V(s'), \ \forall a \in A, \ \forall s \in S \end{array}$$

with optimization variables $V(s), \ \forall s \in S$