Lecture 6

He Li 2014012684

2017/10/31

1. Supported Vector Machine Continued

Soft-margin SVM. *How to find a linear classifier when the data set is not seperable? The soft-margin SVM can be defined as:*

$$\min_{w,b,\xi_i} \frac{1}{2}\|\omega\|^2 + C\sum_i \xi_i$$
$$s.t. \quad \begin{cases} y_i(\omega^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases}$$

The soft-margin SVM can be rewritten as

$$\max_{\lambda_i} \sum_i \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j (x_i^T x_j)$$
$$s.t. \quad \begin{cases} 0 leq \lambda_i \leq C \\ \sum_i \lambda_i y_i = 0 \end{cases}$$

Hinge Loss. *The above sotf-margin SVM can also be rewritten as a optimization problem without constraint. Using hinge loss, the above problem is as same as:*

$$\min_{\omega,b} \frac{1}{2}\|\omega\|^2 + C\sum_i \left[1 - y_i(\omega^T x_i + b)\right]_+$$

where

$$x_+ = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

Here we use hinge loss as a surrogate loss of 0-1 loss, which has the following two good properties:

- hinge loss is the upper bound of 0-1 loss.
- hinge loss is computationally efficient.
- although hinge loss is not differentiable everywhere, it is convex.

Sometimes we want to do some mapping on the original space.

$$\max_{\lambda_i} \sum_i \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j (\phi(x_i)^T \phi(x_j))$$
$$s.t. \quad \begin{cases} 0 leq \lambda_i \leq C \\ \sum_i \lambda_i y_i = 0 \end{cases}$$

where $x = (x^{(1),...,x^{(d)}})$, and for example

$$x :\mapsto \phi(x) = (x^{(1)}, ..., x^{(d)}, [x^{(1)}]^2, [x^{(1)}x^{(2)}], ..., [x^{(d)}]^2)$$

However, sometimes we cannot have a explicit form of $\phi(\cdot)$. We have **kernel trick**.

Kernel Trick. *We define a binary function* $K(\cdot, \cdot)$*,*

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

For example, Gaussian Kernel is

$$K(x, x') = \exp\left\{-\frac{\|x - x'\|^2}{2\sigma^2}\right\}$$

Reproducing kernel Hilbert space**???**

2. Boosting(Meta Learning)

Idea Combine base classifier

1. generate
2. combine

Algorithm 1 AdaBoost

AdaBoost. Require: Input $S = \{(x_1, y_1), ..., (x_n, y_n)\}$, $y_i \in \{\pm 1\}$
Require: $\mathcal{A}$ a base learning algorithm
  Initialize $D_1(i) = \frac{1}{n},\ i \in \{1, ..., n\}$
  for $t = 1, 2, ..., T$ **do**
    Learn a base classifier $h_t(\cdot)$ using $\mathcal{A}$ with $D_t(\cdot)$ on S
    $\epsilon_t := \sum_{i=1}^n D_t(i) I[y_i \neq h_t(x_i)]$
    $\gamma_t := 1 - 2\epsilon_t$
    $\alpha_t := \frac{1}{2} \ln \frac{1-\gamma_t}{1+\gamma_t}$
    $z_t := \sum_i D_t(i) \exp\{-y_i \alpha_t h_t(x_i)\}$
    $D_{t+1}(i) = \frac{D_t(i) \exp\{-y_i \alpha_t h_t(x_i)\}}{z_t}$
  end for
  return $F(x) = \text{sgn}\left[\sum_{t=1}^T \alpha_t h_t(x)\right]$

Proposition(Homework). *AdaBoost is a greedy exponential loss with the following two properties:*

$$\alpha_t = \arg\min_{\alpha} \sum_{i=1}^n D_t(i) \exp\{-y_i \alpha_t(x_i)\} \tag{1}$$

$$\prod_{t=1}^T z_t = \frac{1}{n} \sum_{i=1}^n \exp\left\{-y_i \sum_{t=1}^T \alpha_t h_t(x_i)\right\} = \frac{1}{n} \sum_i \exp\{-y_i f(x_i)\} \tag{2}$$

Note that $\exp\{-y_i f(x_i)\}$ is also a surrogate loss of 0-1 loss, differentiable as well as convex.

Proposition(Homework). *Suppose* $\gamma_t \geq \gamma \geq 0$ *for* $t \in [1, ..., T]$*. Then*

$$\begin{aligned} P_s(yf(x) \leq 0) &= \frac{1}{n} I[y_i f(x_i) \leq 0] \\ &\leq \frac{1}{n} \sum_{i=1}^n \exp\{-y_i f(x_i) \leq 0\} \\ &\leq (1 - \gamma^2)^{\frac{T}{2}} \end{aligned}$$

Proposition(Homework). *Calculate the following function*

$$\frac{1}{n} D_{t+1}(i) I[y_i \neq h_t(x_i)]$$

Note that

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$
$$\tilde{f}(x) = \frac{\sum_{t=1}^{T} \alpha_t h_t(x)}{\sum_{t=1}^{T} \alpha_t}$$

which is a convex combination of $h_t(x)$ and $y\tilde{f}(x) \in [-1, 1]$. We can see this as a margin. In SVM margin represents Euclidean distance yet here margin denotes confidence. **???**

3. Bagging(Bootstrap aggregating)

Bootstrap. *Given dataset* $D = \{x_1, ..., x_n\}$*, draw with replcement, we can get many dadaset with the same sample size.* $\{x_1^1, ..., x_n^1\}, \{x_1^2, ..., x_n^2\}, ..., \{x_1^k, ..., x_n^k\}$.

Algorithm 2 Bagging

Bagging. Require: Input $S = \{(x_1, y_1), ..., (x_n, y_n)\}$
Require: $\mathcal{A}$ a base learning algorithm
  Bootstrap on the original dataset S and get $S_1, ..., S_k$
  for $t = 1, 2, ..., k$ **do**
    Learn a base classifier $h_t(\cdot)$ using $\mathcal{A}$ on S_t
  end for
  return $F(x) = \frac{1}{k} h_t(x)$