Chapter 5: Advanced Classification Methods

Richard Liu

October 26, 2019

Contents

KNN

Very straightforward!

Section 1

SVM

Subsection 1

Introduction

- Solve classification problems.
  - Given a data point, making a decision on which group it belongs to.
- Basic ideas: Given a set of data(training data), construct a so-called 'boundary' and use it to 'split' different groups of data and make decisions.

Graphic representation

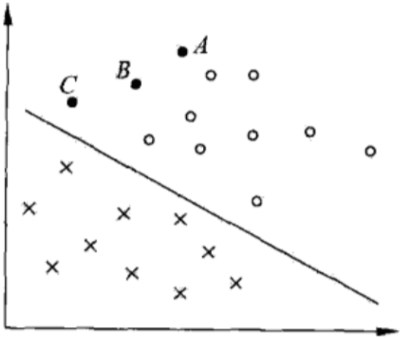


Figure: Bi-object classification problem

Subsection 2

Linear SVM- hardmax

Basic ideas

- Consider a bi-object classification problem, we use a linear boundary

$$wx(\text{inner}\quad\text{product}) + b = 0$$

to separate two groups. And use

$$sign(wx_i + b)$$

to denote the group to which x_i belongs.

- Different signs mean different groups and the concrete group identity of one data point depends on the training results.

Functional Margin

- If one points lies too close to the boundary, it is reasonable to suspect that this point belongs to the other category.
  - New training data points may shift the boundary and change the classified category of this kind of points.
- So we use Functional Margin

$$\hat{\gamma}_i = y_i(wx_i + b)$$

to denote the margin of one point. And

$$\hat{\gamma} = \min_i \gamma_i$$

to denote the margin of the boundary.

Geometrical Margin

- Drawbacks: Proportional change of w, b will make functional margin change, though the boundary remains the same.
- Solution: Use Geometrical Margin

$$\gamma_i = y_i \left(\frac{w}{\|w\|} x_i + \frac{b}{\|w\|} \right)$$

And

$$\gamma = \min_i \gamma_i$$

to denote the functional margin, respectively.

Optimization Form

Obviously, people want to make all points as far as possible from the boundary to make the results more stable. So the optimization problem is

$$\max_{w,b} \quad \frac{\hat{\gamma}}{||w||}$$
$$s.t. \quad y_i(wx_i + b) \geq \hat{\gamma}$$

WLOG, let $\hat{\gamma} = 1$, we can change the formula as

$$\min_{w,b} \frac{1}{2}||w||^2$$
$$s.t. \quad y_i(wx_i + b) - 1 \geq 0$$

Why we call the model SVM?

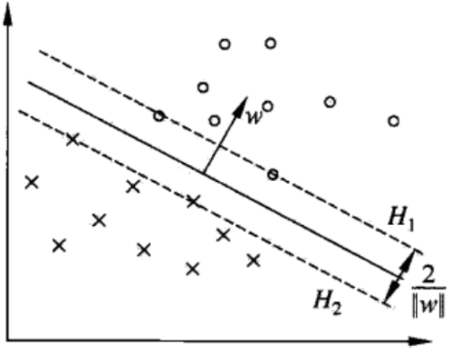


Figure: Support Vectors and Margin

Subsection 3

Linear SVM- softmax

Linear SVM- softmax

In some cases, it is hard to satisfy the conditions

$$y_i(w \cdot x_i + b) \geq 1$$

for all points. So we add a penalized parameter C. Then the model updates as

$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{N} \xi_i$$

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

Where ξ_i is the adjustment parameters.

Subsection 4

Solve the optimization problem

Solve the optimization problem

Theorem 1

The dual problem of the optimization formula before is

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N} \alpha_i$$

$$s.t. \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \cdots, N$$

Solve the optimization problem

Proof(Section 1)

Construct a Lagrange function

$$L(w, b, \xi, \alpha, \mu) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i$$

$$-\sum_{i=1}^{N}\alpha_i(y_i(w \cdot x_i + b) - 1 + \xi_i) - \sum_{i=1}^{N}\mu_i\xi_i$$

where $\alpha_i, \mu_i \geq 0$. It is a max-min problem, so we first solve the minimal w.r.t. w, b, ξ

Solve the optimization problem

Proof(Section 2)

$$\begin{cases} \nabla_w L = w - \sum_{i=1}^N \alpha_i y_i x_i = 0 \\ \nabla_b L = -\sum_{i=1}^N \alpha_i y_i = 0 \\ \nabla_{\xi_i} L = C - \alpha_i - \mu_i = 0 \end{cases}$$

Solve the system we can obtain

$$\begin{cases} w = \sum_{i=1}^N \alpha_i y_i x_i \\ \sum_{i=1}^N \alpha_i y_i = 0 \\ C - \alpha_i - \mu_i = 0 \end{cases} \tag{1}$$

Solve the optimization problem

Proof(Section 3)

Substitute (1) into $L(w, b, \xi, \alpha, \mu)$ we can obtain

$$\min_{w,b,\xi} L = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^{N}\alpha_i$$

where

$$\begin{cases} \sum_{i=1}^{N}\alpha_i y_i = 0 \\ C - \alpha_i - \mu_i = 0 \\ \alpha_i \geq 0 \\ \mu_i \geq 0 \end{cases}$$

Solve the optimization problem

Proof(Section 4)

Just do a little transformation, remove μ_i. The proof is done.

Remark

After getting solutions $\alpha^* = (\alpha_1^*, \alpha_2^*, \cdots, \alpha_N^*)^T$, the boundary function will be

$$\begin{cases} w^* = \sum_1^N \alpha_i^* y_i x_i \\ b^* = y_j - \sum_{i=1}^N y_i \alpha_i^* (x_i \cdot x_j) \end{cases}$$

More rigorous details are linked to convex optimization theory which I do not want to show.

Support Vectors

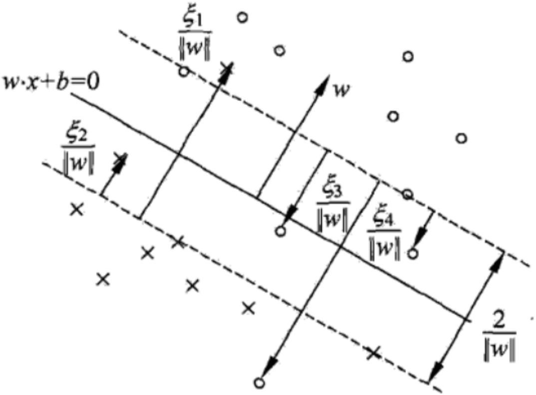


Figure: Support Vectors and Margin for soft-max

Explanation

- ξ_1: Wrong, $\xi_1 > 1, \alpha_1^* = C$
- ξ_2: Right but not good, $0 < \xi_2 < 1, \alpha_2^* = C$
- Other circles: Right, $\alpha_2^* < C$

Subsection 5

*Hinge Loss

Hinge Loss

Theorem 2

The original optimization problem of LSVM

$$\begin{cases} \min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i \\ s.t. \quad y_i\left(w \cdot x_i + b\right) \geqslant 1 - \xi_i \\ \xi_i \geqslant 0 \end{cases}$$

tantamounts to

$$\min_{w,b} \sum_{i=1}^{N} \left[1 - y_i\left(w \cdot x_i + b\right)\right]_+ + \lambda\|w\|^2$$

Section 2

SVM: Kernel Trick

Subsection 1

An Example

An Example

Suppose the boundary of two groups is an ellipse-like shape, how to use SVM to classify these two?

- Answer: Kernel Trick
- Use a new map function to change the data points into new linear space. Then the non-linear boundary can become linear in new space.

Mathematical Explanation

Suppose $x = (x_1, x_2)'$ for each data point, let

$$z = (z_1, z_2)' = \phi(x) = (x_1^2, x_2^2)'$$

Then

$$w_1 x_1^2 + x_2 x_2^2 + b = w \cdot \phi(x) + b = w \cdot z + b = w_1 z_1 + w_2 z_2 + b$$

where $w = (w_1, w_2)'$.This means ϕ can change the ellipse to a hyperplane.

Graphical Representation

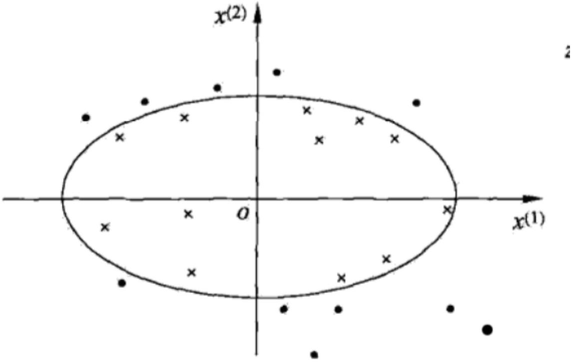


Figure: Ellipse-shape boundary

Graphical Representation

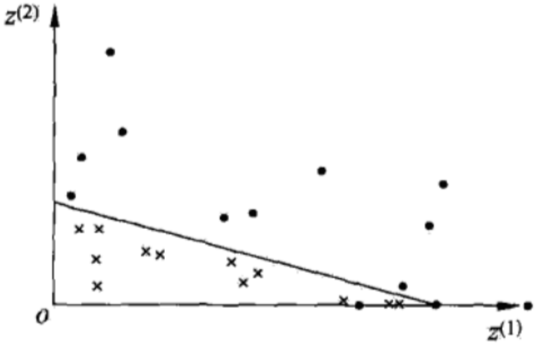


Figure: Hyperplane boundary

Subsection 2

Mathematical Meaning

Map Function and Kernel

Consider the optimization function

$$Q(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N}\alpha_i$$

After the mapping ϕ, the function will be

$$Q(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (\phi(x_i) \cdot \phi(x_j)) - \sum_{i=1}^{N}\alpha_i$$

Map Function and Kernel

Let $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, we can get the ultimate form

$$Q(\alpha) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{N} \alpha_i$$

- It is widely accepted that $K(\cdot, \cdot)$ is much easier to compute than $\phi(\cdot)$.
- One application is to make non-separable data separable in a higher-dimension linear space without figuring out ϕ

Graphical Representation

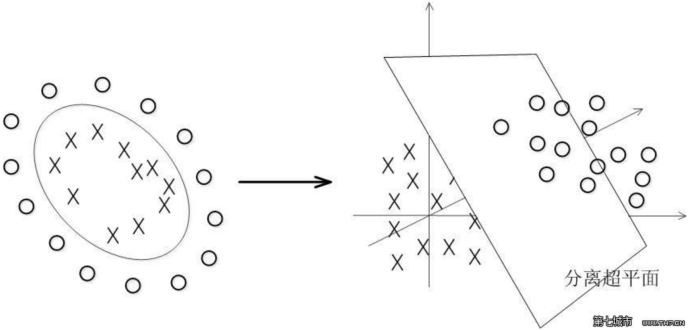


Figure: Low to High Dimension Transformation

Remark

After the change, the decision function will be

$$f(x) = sign(\sum_{i=1}^{N} \alpha_i^* y_i K(x_i, x) + b^*)$$

(See page 17)

Section 3

Ensemble Learning: Bagging

Why ensemble?

Example

Consider an ensemble of 25 independent binary classifiers, each of which has an error rate of $\epsilon = 0.35$, then the total error rate of the ensemble classifier is

$$e_{ensemble} = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.06$$

Although the assumption is hard to reach out, the improvements are usually significant when models possess slight correlations.

General Algorithm

Algorithm 5.5 General procedure for ensemble method.

```
Let D denote the original training data, k denote the number of base classifiers,
    and T be the test data.
for i = 1 to k do
    Create training set, D_i from D.
    Build a base classifier C_i from D_i.
end for
for each test record x ∈ T do
    C*(x) = Vote(C_1(x), C_2(x), ..., C_k(x))
end for
```

Bagging

Algorithm 5.6 Bagging algorithm.

1: Let k be the number of bootstrap samples.
2: **for** $i = 1$ to k **do**
3: Create a bootstrap sample of size N, D_i.
4: Train a base classifier C_i on the bootstrap sample D_i.
5: **end for**
6: $C^*(x) = \underset{y}{\operatorname{argmax}} \sum_i \delta\big(C_i(x) = y\big)$.
$\{\delta(\cdot) = 1$ if its argument is true and 0 otherwise$\}$.

Approximately 63% of the data will be sampled into the dataset for training by bootstrap.

Explanation

$$\frac{1}{n} + \frac{n-1}{n^2} + \frac{(n-1)^2}{n^3} + \cdots = ?$$

One Practical Example

Table 5.4. Example of data set used to construct an ensemble of bagging classifiers.

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	−1	−1	−1	−1	1	1	1

Except for bagging, we will also use it to illustrate the process of Adaboost.

One Practical example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

x <= 0.65 ==> y = 1
x > 0.65 ==> y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ==> y = 1
x > 0.3 ==> y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Section 4

Ensemble Learning: Boosting

Boosting

Definition

An iterative procedure used to adaptively change the distribution of training examples so that the basic classifiers will focus on examples that are hard to classify.

- Note that boosting classifier tries to focus on the training examples wrongly classified, it is susceptible to overfitting.

Adaboost

Algorithm 5.7 AdaBoost algorithm.

```
1: w = {w_j = 1/N | j = 1, 2, ..., N}.   {Initialize the weights for all N examples.}
2: Let k be the number of boosting rounds.
3: for i = 1 to k do
4:     Create training set D_i by sampling (with replacement) from D according to w.
5:     Train a base classifier C_i on D_i.
6:     Apply C_i to all examples in the original training set, D.
7:     ε_i = (1/N) [Σ_j w_j δ(C_i(x_j) ≠ y_j)]   {Calculate the weighted error.}
8:     if ε_i > 0.5 then
9:         w = {w_j = 1/N | j = 1, 2, ..., N}.   {Reset the weights for all N examples.}
10:        Go back to Step 4.
11:    end if
12:    α_i = (1/2) ln ((1 − ε_i)/ε_i).
13:    Update the weight of each example according to Equation 5.69.
14: end for
15: C*(x) = argmax_y Σ_{j=1}^{T} α_j δ(C_j(x) = y)).
```

Line 7: $\epsilon_i = \frac{1}{N}\left[\sum_j w_j\, \delta\big(C_i(x_j) \neq y_j\big)\right]$

Line 12: $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$

Line 15: $C^*(\mathbf{x}) = \underset{y}{\operatorname{argmax}} \sum_{j=1}^{T} \alpha_j \delta(C_j(\mathbf{x}) = y))$

Adaboost

Equation 5.69

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \times \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor.

Adaboost: A Practical Example

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

(a) Training records chosen during boosting

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Some Analytical Results

Remark

$$e_{\text{ensemble}} \leq \prod_i [\sqrt{\epsilon_i(1-\epsilon_i)}]$$

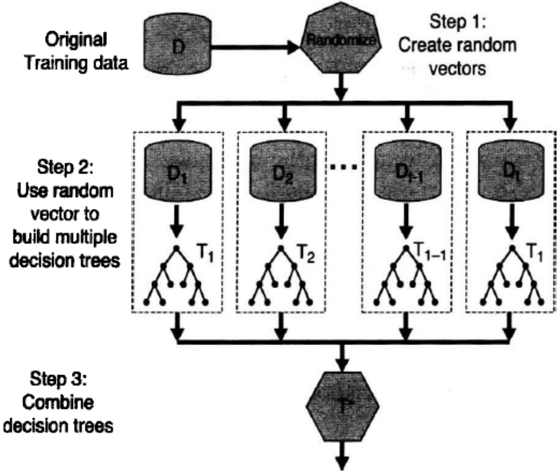
where ϵ_i is the training error.

Remark

$$e_{\text{ensemble}} \leq \prod_i [\sqrt{1-4\gamma_i^2}] \leq \exp(-2\sum_i \gamma_i^2)$$

where $\gamma_i = 0.5 - \epsilon_i$

Random Forests



- A class of ensemble methods designed for decision trees.
- Each tree is generated based on values of an independent set of random vectors sampled from a fixed probability distribution.
- Usually, we randomly select F input features to split at each node of the decision tree. Empirically $F = \log_2 d + 1$, where d is the number of features.
- When d is relatively low, we usually expand the features by linear combination.

Section 5

*Class Imbalance Problem

We will not cover this section, but some concepts, such as **ROC curve**, **Alternative Metrics**, are in fact very important in data engineering.

Section 6

Supplementary

Subsection 1

The Generalization Error of Random Forest

The Generalization Error of Random Forest

Theorem 2

$$E \leq \bar{\rho}(1 - s^2)/s^2$$

where E is the error.

Before we introduce the meanings of unknown variables, we introduce some concepts, then leading to the proof.

Definition: Margin Function

$$mg(\mathbf{X}, Y) = av_k I(h_k(\mathbf{X}) = Y) - \max_{j \neq Y} av_k I(h_k(\mathbf{X}) = j)$$

where $h_k(\mathbf{X}) = h(\mathbf{X}, \Theta_k)$, where Θ_k is the parameter space.

Explanation

This function measures the extent to which the average number of votes at $\mathbf{X}$, Y for the right class exceeds the average vote for any other one. Here $\mathbf{X}$, Y is the training set.

Definition: Generalization Error

$$PE^* = P_{\mathbf{X},Y}(mg(\mathbf{X}, Y) < 0)$$

Explanation

Here, this means the values are sampled from the space $(\mathbf{X}, Y)$, not only the training set.

Lemma

As the number of tree increases, we have

$$PE^* \to P_{\mathbf{X},Y}(P_\Theta(h(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j) < 0)$$

almost surely.

Definition: Margin Function 2

$$mr(\mathbf{X}, Y) = P_\Theta(h(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j)$$

$$= E_\Theta[I(h(\mathbf{X}, \Theta) = Y) - I(h(\mathbf{X}, \Theta) = \hat{j}(\mathbf{X}, Y))]$$

where

$$\hat{j}(\mathbf{X}, Y) = \arg\max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j)$$

Definition: Strength

$$s = E_{\mathbf{X}, Y} mr(\mathbf{X}, Y)$$

By Chebyshev Inequality we have

$$PE^* \leq \text{var}(mr)/s^2$$

Explanation

$$PE^* = P_{\mathbf{X},Y}(mr(\mathbf{X},Y) - s < -s), then?$$

Lemma: Chebyshev Inequality

$$P(|X - E(X)| > \epsilon) \leq \frac{D(X)}{\epsilon^2}$$

Definition: Raw Margin Function

$$rmg(\Theta, \mathbf{X}, Y) = I(h(\mathbf{X}, \Theta) = Y) - I(h(\mathbf{X}, \Theta) = \hat{j}(\mathbf{X}, Y))$$

From this definition we could get

$$mr(\mathbf{X}, Y) = E_{\Theta}[rmg(\Theta, \mathbf{X}, Y)]$$

That is to say

$$mr(\mathbf{X}, Y)^2 = E_{\Theta,\Theta'}[rmg(\Theta, \mathbf{X}, Y)rmg(\Theta', \mathbf{X}, Y)]$$

Explanation

$$[E_{\Theta}f(\Theta)]^2 = E_{\Theta,\Theta'}f(\Theta)f(\Theta')$$

when Θ, Θ' are independent.

Note that

$$var(mr) = E_{\Theta,\Theta'}(Cov_{\mathbf{X},Y} rmg(\Theta,\mathbf{X},Y) rmg(\Theta',\mathbf{X},Y))$$

$$= E_{\Theta,\Theta'}(\rho(\Theta,\Theta') sd(\Theta) sd(\Theta')) = \bar{\rho}(E_{\Theta} sd(\Theta))^2 \leq \bar{\rho} E_{\Theta} var(\Theta)$$

Explanation

$$[E(X)]^2 \leq E(X^2) \quad Cov_{\mathbf{X},Y}(rmg(\Theta,\mathbf{X},Y) rmg(\Theta',\mathbf{X},Y))$$

$$= E_{\mathbf{X},Y}(rmg(\Theta,\mathbf{X},Y) rmg(\Theta',\mathbf{X},Y)) -$$

$$E_{\mathbf{X},Y}(rmg(\Theta,\mathbf{X},Y)) E_{\mathbf{X},Y}(rmg(\Theta',\mathbf{X},Y))$$

Here we define

$$\bar{\rho} = \frac{E_{\Theta,\Theta'}(\rho(\Theta, \Theta')sd(\Theta)sd(\Theta'))}{E_{\Theta,\Theta'}(sd(\Theta)sd(\Theta'))}$$

Now we have

$$E_{\Theta}var(\Theta) \leq E_{\Theta}(E_{\mathbf{X},Y}rmg(\Theta, \mathbf{X}, Y))^2 - s^2 \leq 1 - s^2$$

Combine with all together, we could obtain our results.

Subsection 2

Xgboost: An Introduction

Learning Objective

For a given dataset with n examples and m features $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}(|\mathcal{D}| = n, \mathbf{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R})$, we have

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), f_k \in \mathcal{F}$$

Here $\mathcal{F} = \{f(\mathbf{x}) = w_{q(\mathbf{x})}\}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ is the CART space.

- T is the number of leaves in the tree, q is the tree structure, e.g. $q_1(\mathbf{x}) = 1$ means for tree 1, $\mathbf{x}$ belongs to leaf 1. Each f_k corresponds to unique q, w.

Learning Objective

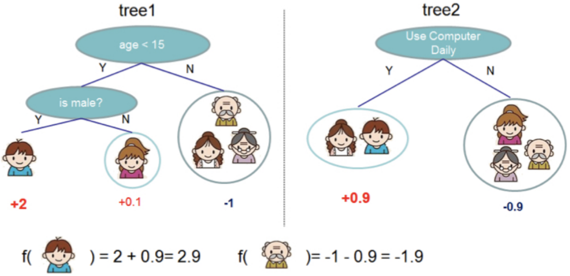


Figure 1: Tree Ensemble Model. The final prediction for a given example is the sum of predictions from each tree.

Learning Objective

We would like to minimize the following regularized objective.

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

where l is the difference function. $\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$

- All differentiable!

Gradient Tree Boosting

For functions are used as parameters, traditional optimization techniques could not be used. Instead, we exploit greedy algorithm, miminizing

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

By Taylor Development we have

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

where g_i, h_i are corresponding partial derivatives.

Remove the term independent of time step t, we have

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}[g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

$$= \sum_{i=1}^{n}[g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T$$

where $I_j = \{i|q(\mathbf{x}_i) = j\}$, the instance set of leaf j.(Remember the definition of $\mathcal{F}$)

Optimization: We could get our final results.

Remark: Optimized weights

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}$$

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

Python Code: Click Here

Algorithm

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
    $G_L \leftarrow 0, \ H_L \leftarrow 0$
    for j *in sorted*$(I$, *by* $\mathbf{x}_{jk})$ **do**
        $G_L \leftarrow G_L + g_j, \ H_L \leftarrow H_L + h_j$
        $G_R \leftarrow G - G_L, \ H_R \leftarrow H - H_L$
        $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
    end
end
Output: Split with max score

References

1. Pang-Ning Tan, et al. Introduction to Data Mining
2. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting
3. Leo Breiman, Random Forests
4. Tianqi Chen, et al. XGBoost: A Scalable Tree Boosting System

THANK YOU FOR YOUR LISTENING!