機器學習 作業一 RE6111024葉嘉浤

1. Implementation of Linear Classifier with the update rule.

Input:training data,label,initialize weights,learning rate

For i in training sample:

1.Random choose a sample

2.compute dot product of weights and features of observation

3.take the sign of product result as prediction

If prediction < 0:

repeat

Weight = weight + learning rate \* feature \* label

Constant = constant + label

Until predition > 0 or reach maximum iteration

Output:weights,constant

1. Find the solution by solving this equation using least-squared manner

Input:training data,label

weights = (x^t,x)^-1 x^t y

Output:weights

1. Implementation of the voted perceptron and average voted perceptron

* Voted perceptron

Input:training data,label,learning rate,weights,constant

For i in training sample:

1.Random choose a sample

2.compute dot product of weights and features of observation

3.take the sign of product result as prediction

If prediction > 0:

Correct ++

If prediction < 0:

Save the weights and the number of correction

Weight = weight + learning rate \* feature \* label

Constant = constant + label

Reset correction

Output:weights,constant

* Average voted perceptron

Input: training data,label,learning rate,weights,constant

For i in training sample:

1.Random choose a sample

2.compute dot product of weights and features of observation

3.take the sign of product result as prediction

If prediction > 0:

Correct ++

If prediction < 0:

Save the weights and the number of correction

Weight = weight + learning rate \* feature \* label

Constant = constant + label

Reset correction

Weights = weights \* correction / sum of correct

Output:weights,constant

1. Implementation of the linear classifier with the minimum ||w||^2 property and verify whether the margin of this version is larger than that of the conventional linear classifier

Input:training data,label,learning rate,weights,constant

For i in training sample:

1.Random choose a sample

2.compute dot product of weights and features of observation

3.take the sign of product result as prediction

if prediction < 0:

weights\_new = weight + learning rate \* feature \* label

constant\_new = constant + label

for j in sampled:

distance = distrance + dot(weights,features) + constant

distance\_new = distance\_new + dot(weights\_new,features) + constant\_new

if distance\_new < distance:

weights = weights\_new

constant = constant\_new

Output:weights,constant

1. Based on 4., add the slack variable term in the linear classifier and find the most effective weighting value C

Input:training data,label,learning rate,weights,constant

For i in training sample:

1.Random choose a sample

2.compute dot product of weights and features of observation

3.take the sign of product result as prediction

if prediction < 0:

weights\_new = weight + learning rate \* feature \* label

constant\_new = constant + label

for j in sampled:

distance = distrance + dot(weights,features) + constant

distance\_new = distance\_new + dot(weights\_new,features) + constant\_new

if dot(weights,features)+constant > slack:

distance = distrance + dot(weights,features) + constant

distance\_new = distance\_new + dot(weights\_new,features) + constant\_new

if distance\_new < distance:

weights = weights\_new

constant = constant\_new

Output:weights,constant

1. Use existing SVM package,and make a performance comparison between the built-in SVM and the implementations above.