Computational, sample, hypothesis-space complexity

Huyen Do

June 16, 2010

1 Computational Complexity

1.1 Popular classes of computational complexity

1. Constant time $O(1)$
2. Linear time: $O(n)$
3. Quadratic time: $O(n^2)$
4. Cubic time: $O(n^3)$
5. Polynomial time P: $2^{O(logn)}$
6. Exponential time E: $2^{O(n)}$ or $2^{poly(n)}$
7. Logarithmic time: $O(logn)$

1.2 SVC-algorithms

The optimization problem of SVM- algorithms is a quadratic programming with linear equality and inequality constraints and a positive definite matrix. In general the ellipsoid method solves such quadratic optimization problem in polynomial time.

By exploiting some special features of SVC algorithms, the complexity can be reduced enormously. For example, Platt SMO algorithms have complexity between $O(N)$ and $O(N^2)$ where N is number of instances.

According to svmtutorial of Burges () the worst case computational complexity is $O(N_{sv}^3)$ (inversion of the Hessian) where N_{sv} is number of support vectors. Let denote: N_sv number of support vectors, d - number of features and N - number of instances, the complexity of SVM is $O(N_{sv}^3 + (N_{sv}^2 n + N_{sv}Nd)$.

1.3 Decision Trees

$O(Nd^2)$ where N -number of instances, d - number of features.

1.4 KNN Algorithm

K-nn search for k nearest neighbors of the new instances. The simplest (naive) solution: $O(Nd)$ where N is number of instances and d is number of features. For a better approach, the complexity can be reduced to $O(logN)$ (branch and bound algorithm.

1.5 Logistic Regression Algorithm

1.6 FisherLinearDiscriminant

The optimization problem is a regular eigenvalue problem for a symmetric positive definite matrix which has the computational complexity of $O(N^3)$ where N is size of the matrix or number of instances.

1.7 Naive Bayes Algorithm

$O(Nd)$

2 Complexity of Hypothesis space

2.1 Complexity measurement

One simple measure of the complexity of hypothesis space is its cardinality. However this measure can be used for only finite hypothesis space. For infinite hypothesis space we can use the VC dimension which is defined as following: $VC(\mathcal{H})$ of hypothesis space $\mathcal{H}$ defined over instance space X is the size of the largest finite subset of X shattered by $\mathcal{H}$. If arbitrarily large finite sets of X can be shattered by $\mathcal{H}$, the $VC(\mathcal{H})$ will be ∞.

VC dimension of class of linear learning machine over R^d is $d + 1$.

For any finite $\mathcal{H}$, $VC(\mathcal{H}) \leq log_2(\mathcal{H})$.

Rademacher complexity: measures richness of a class of real-valued functions with respect to a probability distribution.

2.2 SVC-algorithms

Hypothesis space: linear hyperplanes.

There is a connection between the margin and the VC-dimension (Vapnik). The lemma states that the VC-dimension is lower the larger the margin. That's why support vector machines implement structural risk minimization.

According to svmtutorial of Burges (Burges,): Let K be a positive kernel which corresponds to a minimal embedding space $\mathcal{H}$. Then the VC dimension of the corresponding support vector machine (where the parameter C is allowed to take all values) is $dim(\mathcal{H}) + \infty$.

For polynomial kernel $(x_1.x_2)^p$: $D_{VC} = C_p^{d+p-1} + 1$.

From Vapnik: VC dimension of $\mathcal{H}_\gamma$ - set of linear classifier in $\mathbb{R}^D$ with margin γ on samples X , is bounded by

$$VC(\mathcal{H}_\gamma) \leq min\{D, [\frac{4R^2}{\gamma^2}]\}$$

where $\|x_i\| \leq R, \forall i$

2.3 Decision Trees

Hypothesis space: orthogonal space partitioning.

For a 'standard' Decision trees, the cardinality of the hypothesis space is $c\Pi_{i=1}^d(f_i+1)$ where d-number of features, and each feature has f_i discrete values, $i = 1..d$, c- number of classes.

2.4 KNN Algorithm

Hypothesis space: any nonlinear functions.

Given a fixed training set, the Voronoi diagram which determines the k nearest neighbor is also fixed. The hypothesis space of 1-nn is $O(N)$ where N is number of instances. The VC dimension of 1-nn is also N.

2.5 FisherLinearDiscriminant

Hypothesis space: linear hyperplanes.

VC dimension: at last $d + 1$ where N is number of features.

2.6 Logistic Regression Algorithm

2.7 Naive Bayes Algorithm

3 Sample complexity

3.1 Definition (Mitchell, 1997)

In the context of PAC framework, the sample complexity of the learning problem is the number of training instances required to achieve a level of accuracy.

The number of training instances required to assure that any consistent hypothesis will be probably (with probability $(1 - \delta)$ approximately (within error ϵ) correct when learning any target concepts in $\mathcal{H}$ is:

$$n \geq \frac{1}{\epsilon}(ln|\mathcal{H}| + ln(1/\delta))$$

.

Consider any concept class C such that $VC(C) \geq 2$, any learner L and any $0 < \epsilon < \frac{1}{8}$, and $0 < \delta < \frac{1}{100}$. Then there exists a distribution $\mathcal{D}$ and target concept in C such that if L observes fewer instances than

$$max[\frac{1}{\epsilon}log(1/\delta), \frac{VC(C)-1}{32\epsilon}]$$

then with probability at least δ, L outputs a hypothesis h having $error_{\mathcal{D}} > \epsilon$

3.2 SVC-algorithms

$$n \geq max[\frac{1}{\epsilon}log(1/\delta), \frac{VC(SVC)-1}{32\epsilon}]$$

3.3 Decision Trees

$$n \geq \frac{1}{\epsilon}(ln|\mathcal{H}| + ln(1/delta))$$

where $|\mathcal{H}| = c\Pi_{i=1}^{d}(f_i + 1)$ and d-number of features, and each feature has f_i discrete values, $i = 1..d$, c- number of classes.

3.4 KNN Algorithm

3.5 Logistic Regression Algorithm

3.6 FisherLinearDiscriminant

3.7 Naive Bayes Algorithm

References

Burges. Svm tutorial.

Mitchell, T. M. (1997). *Machine learning.* Boston, Massachusetts: WCB/McGraw-Hill.