Using Algorithms Designed for Adversaries on Transfer Learning Problems

Chris Mesterharm [1] Michael J. Pazzani [2]

[1]Applied Communication Sciences
150 Mt Airy Road
Basking Ridge, NJ 07920
[2]University of California, Riverside
900 University Ave
Riverside, CA 92521

Oct 11, 2012

Problem Definition

Transfer Learning

Two or more learning problems that are related to one another such that learning about one gives information about the others.

Example 1

Use data from pancreatic cancer to help predict liver cancer.

Example 2

Use drug trial information on a particular group (male) to learn to predict a drugs effect on a different group (female).

Example 3

Transfer knowledge on locating battleships in satellite images to locating aircraft carriers.

Outline

1. Learning Model.
2. Transfer (Multitask) Learning
3. Shifting Concepts.
4. Relevant Subset.

Inductive Learning Model

Notation

Assume I have a fixed but unknown distribution $P(X, Y)$ where X is the set of examples and Y is the set of labels. Assume that H is a set of hypothesis that map X to Y.

Problem

Input A set of m independent samples from $P(X, Y)$.

Output A function $\hat{h} \in H$ that has low error on $P(X, Y)$.

Bound $\forall h \in H \;\; \text{err}(h) \leq \overline{\text{err}}(h) + \epsilon(m, \delta, H)$ where δ is the probability of failure. For example, $\epsilon = O\left(VC(H)/m + \ln\left(1/\delta\right)/m\right)$

[Vapnick and Chervonenkis 1968]

Structural Risk Minimization

Notation

Partition H into $H = H_1 \cup H_2 \cup \ldots$ and give a non-negative weight μ_i to each element of the partition such that $\sum_{i=1}^{\infty} \mu_i = 1$.

Problem

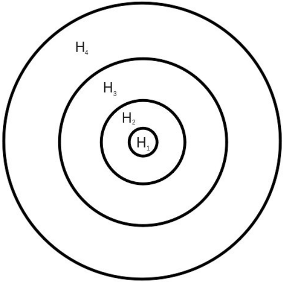
Input: A set of m independent samples from $P(X, Y)$.

Output: A function $\hat{h} \in H$ that has low error on $P(X, Y)$.

Bound: $\forall i \in N \ \forall h \in H_i \ \ \text{err}(h) \leq \overline{\text{err}}(h) + \epsilon(m, \delta, H_i, \mu_i)$ where δ is the probability of failure.

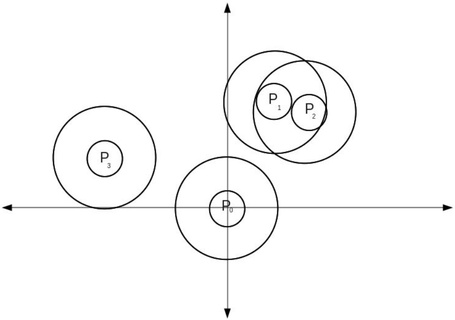
[Vapnick and Chervonenkis 1974]

One possible model of transfer learning is a customizable form a structural risk minimization.



The old hypotheses is the center of H_1, and the complexity of learning grows the further a hypothesis is from the center.

Assume P_1 to P_k are k different learning problems where each problem has m_i training instances.



How do we use instances to do well on all k problems?

Tracking Changing Concepts

On-line learning problem

- Hyperplane concept that is allowed to change.
- Let $C = u^1,\ u^2,\ \ldots,\ u^k$ be a sequence of k hyperplane weight vectors.
- Assume that m_1 labeled instances are generated according to the classifier predict 1 iff $\mathbf{u^1} \cdot \mathbf{x} \geq 1$, that m_2 instances are generated according to $\mathbf{u^2} \cdot \mathbf{x} \geq 1,\ \ldots$.
- We can bound the number of mistakes as a function of the distances between the weight vectors.

Shifting hyperplanes with Winnow

Let $\lambda \geq \max_{t \in \{1,\ldots,T\}} \|X_t\|_1$.
Let $\zeta = \min_{t \in \{1,\ldots,T\},\ i \in \{1,\cdots,n\}} \{x_{i,t} \mid x_{i,t} > 0\}$.
Let $H(C) = \sum_{i=1}^{n} (u_i^k + \sum_{j=1}^{k-1} \max(0, u_i^j - u_i^{j+1}))$.
$H(C) \leq \sum_{j=0}^{k-1} ||\mathbf{u}^j - \mathbf{u}^{j+1}||_1$.
Let $\nu_t = \max[0, \Delta - y_t(\mathbf{u_{C(t)}} \cdot \mathbf{x_t} - 1)]$.
Let $N = \sum_{t=1}^{T} \nu_t$

Theorem

For instances generated by a concept sequence C, if $\alpha = 1 + \Delta$ and $\epsilon = \sigma = \frac{\Delta}{50\lambda}$ then the number of mistakes is less than

$$(2.05 + \Delta)\left(\frac{\zeta H(C)}{\Delta(1+\Delta)} + \frac{\ln\left(\frac{50\lambda}{\Delta\zeta}\right) H(C)}{\Delta^2} + \frac{N}{\Delta(1+\Delta)}\right) .$$

Sequences of Concepts

We can relate changing concepts to multitask learning by considering all possible sequences of concepts.

sequences

- Let T be the number of possible sequences of concepts.
- $T = \frac{k!}{0!} + \frac{k!}{1!} + \cdots \frac{k!}{(k-1)!} \leq ek!$
- Increases the number of mistakes by at most $O(k \ln k)$.
- As long as k is small $ek!$ is computationally tractable.
- Easy to parallelize.
- Can consider only subset of sequences.

Multitask Bounds

sequences

- We need to convert algorithm from on-line to batch.
  - Can use voting, averaging, etc.
- Bound is for average accuracy over all concepts.
- Can use any p-norm changing concept bound.
- Bound depends on the existence of a good sequence.
- Error bound will depend total number of mistakes divided by the total number of instances.

Winnow example where C contains all k problems

$$O\left(\frac{\ln(n)\,H(C)}{\Delta^2\sum_{i=1}^{k} m_i} + \frac{N}{\Delta(1+\Delta)\sum_{i=1}^{k} m_i} + \frac{k\ln(k)}{\sum_{i=1}^{k} m_i} + \sqrt{\frac{\ln(1/\delta)}{\sum_{i=1}^{k} m_i}}\right).$$

Algorithms such as Winnow depend on the KL-divergence. This can give improved performance over the the 1-norm for Transfer Learning problems.

Subset bound

- Let A and B correspond to a partition of the attributes into two pieces.
- Assume $|A| = r$ and $|B| = n - r$ where $r << n$.
- Assume instances have label 1 iff $\sum_{i \in A} u_i x_i + \sum_{i \in B} u_i x_i \geq 1$.
- The number of mistakes made by Subspace Winnow is roughly at most

$$2\left(\frac{\sum_{i \in A} u_i \ln(2r) + \sum_{i \in B} u_i \ln(2n)}{\Delta^2}\right)$$

Conclusion

On-line adversarial learning algorithms are an effective tool for building batch learning algorithms.

Advantages

- Adversary corresponds to a worst case sequence.
- Extremely cheap.
- Strong theoretical results.
- Standard techniques exist to convert to batch as final stage.

Learning Problems

- Transfer learning.
- Active learning.
- Multiclass extensions.