Online Nonlinear Support Vector Machine for Large-Scale Classification

Yuh-Jye Lee
Joint work with Y.-C. Tseng and I.-F. Chen

Lab of Data Science and Machine Intelligence
Dept. of CSIE@TaiwanTech.

Dept. of Applied Mathematics, NCTU
October 6, 2015

Outline

Outline

Supervised Learning Problems

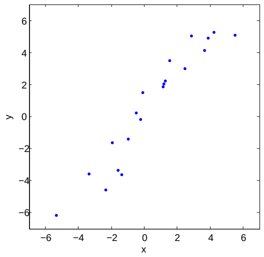
- Given a training set $S = \{(\mathbf{x}^1, y_1), (\mathbf{x}^2, y_2), \ldots, (\mathbf{x}^m, y_m)\}$. We would like to construct a *hypothesis (or classifier)*, $h(\mathbf{x})$ that can correctly predict the *unseen* label y given a new instance $\mathbf{x}$
  - If $h(\mathbf{x}) \neq y$ then we get some loss or penalty
  - For example: $\ell(h(\mathbf{x}), y) = \frac{1}{2}|h(\mathbf{x}) - y|$
- Key Assumption: training instances are drawn from an unknown but fixed probability distribution $P(\mathbf{x}, y)$ independently.
- Two supervised learning examples:
  - If y is drawn from a *finite set* it will be a *classification problem*.
  - If y is a *real number* it becomes a *regression problem*

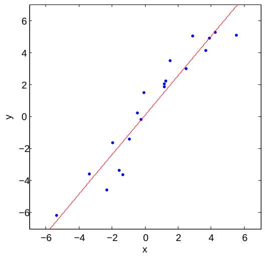
A Supervised Learning Example: Data Fitting

Suppose we want to fit the data

$$(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$$

with a straight line $y = w_0 + w_1 x$.





Least Squares Problem
Regression in Supervised Learning

- Given a linear system, $Aw = y,\ A \in \mathbb{R}^{m\times n}$ with $m > n$:
- If linear system has no solution, an approximated solution can be obtained by solving the following minimization problem.

$$\min_{w\in\mathbb{R}^n} r^\top r = \min_{w\in\mathbb{R}^n} \|r\|_2^2 = \min_{w\in\mathbb{R}^n} \sum_{i=1}^{m} (y_i - A_i w)^2, \qquad (1)$$

where $r = y - Aw \in \mathbb{R}^m$ is the *residual*.

- You can fit them with ℓ_1 loss function

$$\min_{w\in\mathbb{R}^n} \|r\|_1 = \min_{w\in\mathbb{R}^n} \sum_{i=1}^{m} |y_i - A_i w| \qquad (2)$$

Binary Classification Problem

Given a training set

$$S = \{(\mathbf{x}^j, y_j) | \mathbf{x}^j \in \mathbb{R}^d, y_j \in \{-1, 1\}, j = 1, \ldots, m\}$$

$$\mathbf{x}^j \in P \Leftrightarrow y_j = 1 \text{ \& } \mathbf{x}^j \in N \Leftrightarrow y_j = -1$$

Main Goal:

$$\boxed{\text{Predict the unseen class label for new data}}$$

Find a function $f : \mathbb{R}^d \to \mathbb{R}$ by learning from data

$$f(\mathbf{x}) \geq 0 \Rightarrow \mathbf{x} \in P \text{ and } f(\mathbf{x}) < 0 \Rightarrow \mathbf{x} \in N$$

$$h(\mathbf{x}) = sgn(f(\mathbf{x}))$$

The simplest function is linear:

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \sum_{i=1}^{d} w_i x_i + b$$

Expected Risk vs. Empirical Risk

- Assumption: training instances are drawn from an unknown but fixed probability distribution $P(\mathbf{x}, y)$ independently.
- Ideally, we would like to have the *optimal rule* h^* that minimizes the *Expected Risk*: $E(h) = \int \ell(h(\mathbf{x}), y) dP(\mathbf{x}, y)$ among all functions
- Unfortunately, we can not do it. $P(\mathbf{x}, y)$ is unknown and we have to restrict ourselves in a certain *hypothesis space*, $\mathcal{H}$
- How about compute $h^*_m \in \mathcal{H}$ that minimizes the *Empirical Risk*:

$$E_m(h) = \frac{1}{m} \sum_j \ell(h(\mathbf{x}^j), y_j)$$

- Only minimizing the empirical risk will be in danger of *overfitting*

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem
- The objective function consists of two parts: $E_m(h)$(bias)+ controls on VC-error bound (variance)
- Controlling the VC-error bound will avoid the *overfitting* risk
- It can be achieved via adding the *regularization* term into the objective function
- Note that: We have made lots of approximations when formulate a learning task as an optimization problem
    - Why bother to find the optimal solution for the problem?
    - One could stop the optimization iteration before its convergence

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem
- The objective function consists of two parts: $E_m(h)$(bias)+ controls on VC-error bound (variance)
- Controlling the VC-error bound will avoid the *overfitting* risk
- It can be achieved via adding the *regularization* term into the objective function
- Note that: We have made lots of approximations when formulate a learning task as an optimization problem
  - Why bother to find the optimal solution for the problem?
  - One could stop the optimization iteration before its convergence

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem
- The objective function consists of two parts: $E_m(h)$(bias)+ controls on VC-error bound (variance)
- Controlling the VC-error bound will avoid the *overfitting* risk
- It can be achieved via adding the *regularization* term into the objective function
- Note that: We have made lots of approximations when formulate a learning task as an optimization problem
  - Why bother to find the optimal solution for the problem?
  - One could stop the optimization iteration before its convergence

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem
- The objective function consists of two parts: $E_m(h)$(bias)+ controls on VC-error bound (variance)
- Controlling the VC-error bound will avoid the *overfitting* risk
- It can be achieved via adding the *regularization* term into the objective function
- Note that: We have made lots of approximations when formulate a learning task as an optimization problem
    - Why bother to find the optimal solution for the problem?
    - One could stop the optimization iteration before its convergence

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem
- The objective function consists of two parts: $E_m(h)$(bias)+ controls on VC-error bound (variance)
- Controlling the VC-error bound will avoid the *overfitting* risk
- It can be achieved via adding the *regularization* term into the objective function
- Note that: We have made lots of approximations when formulate a learning task as an optimization problem
  - Why bother to find the optimal solution for the problem?
  - One could stop the optimization iteration before its convergence

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem
- The objective function consists of two parts: $E_m(h)$(bias)+ controls on VC-error bound (variance)
- Controlling the VC-error bound will avoid the *overfitting* risk
- It can be achieved via adding the *regularization* term into the objective function
- Note that: We have made lots of approximations when formulate a learning task as an optimization problem
    - Why bother to find the optimal solution for the problem?
    - One could stop the optimization iteration before its convergence

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem
- The objective function consists of two parts: $E_m(h)$(bias)+ controls on VC-error bound (variance)
- Controlling the VC-error bound will avoid the *overfitting* risk
- It can be achieved via adding the *regularization* term into the objective function
- Note that: We have made lots of approximations when formulate a learning task as an optimization problem
  - Why bother to find the optimal solution for the problem?
  - One could stop the optimization iteration before its convergence

Gradient Descent: Batch Learning

- For an optimization problem

$$\min f(\mathbf{w}) = \min r(\mathbf{w}) + \frac{1}{m}\sum_{j=1}^{m} \ell(\mathbf{w}; (\mathbf{x}^j, y_j))$$

- GD tries to find a direction and the learning rate decreasing the objective function value.

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \eta(-\nabla f(\mathbf{w}^t))$$

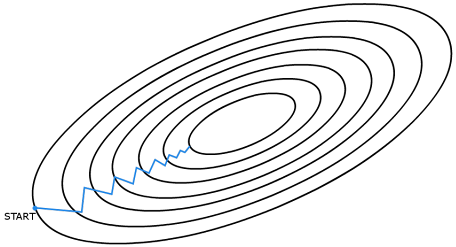
  where η is the learning rate, $-\nabla f(\mathbf{w}^t)$ is the steepest descent direction,

$$\nabla f(\mathbf{w}^t) = \nabla r(\mathbf{w}^t) + \frac{1}{m}\sum_{i=1}^{m} \nabla \ell(\mathbf{w}^t; (\mathbf{x}^i, y_i))$$

- When m is large, computing $\sum_{j=1}^{m} \nabla \ell(\mathbf{w}^t; (\mathbf{x}^j, y_j))$ may cost much time.

Gradient Descent is Bad if Started with a Bad Initial

- Only utilizes the First Order Information
- Only has a linear convergent rate for a simple quadratic function



Stochastic Gradient Descent: Online Learning

- In GD, we compute the gradient using the entire training set.
- In *stochastic gradient descent*(SGD), we use

$$\nabla\ell(\mathbf{w}^t;(\mathbf{x}^t,y_t)) \quad \text{instead of} \quad \frac{1}{m}\sum_{j=1}^{m}\nabla\ell(\mathbf{w}^t;(\mathbf{x}^j,y_j))$$

- So the descent direction of $f(\mathbf{w}^t)$ and $\mathbf{w}^{t+1}$

$$\mathbf{d}^t = -\nabla r(\mathbf{w}^t) - \nabla\ell(\mathbf{w}^t;(\mathbf{x}^t,y_t)), \quad \mathbf{w}^{t+1} = \mathbf{w}^t + \eta\mathbf{d}^t$$

- SGD computes the *descent direction* using only one instance.
- In experiment, SGD is significantly faster than GD when m is large.

People of ACM: David Blei (Sept. 9, 2014)

The recipient of the 2013 ACM- Infosys Foundation Award in the Computing Sciences, he is joining Columbia University this fall as a Professor of Statistics and Computer Science, and will become a member of Columbia's Institute for Data Sciences and Engineering.

[Q]: What is the most important recent innovation in machine learning?

[A]: One of the main recent innovations in ML research has been that we (the ML community) can now scale up our algorithms to massive data, and I think that this has fueled the modern renaissance of ML ideas in industry. The main idea is called *stochastic optimization*, which is an adaptation of an *old algorithm invented by statisticians in the 1950s*.

People of ACM: David Blei (Sept. 9, 2014)

The recipient of the 2013 ACM- Infosys Foundation Award in the Computing Sciences, he is joining Columbia University this fall as a Professor of Statistics and Computer Science, and will become a member of Columbia's Institute for Data Sciences and Engineering.

[Q]: What is the most important recent innovation in machine learning?

[A]: One of the main recent innovations in ML research has been that we (the ML community) can now scale up our algorithms to massive data, and I think that this has fueled the modern renaissance of ML ideas in industry. The main idea is called *stochastic optimization*, which is an adaptation of an *old algorithm invented by statisticians in the 1950s*.

People of ACM: David Blei (Sept. 9, 2014)

[Q]: What is the most important recent innovation in machine learning?

[Continuous]: *In short, many machine learning problems can be boiled down to trying to find parameters that maximize (or minimize) a function*. A common way to do this is "gradient ascent," iteratively following the steepest direction to climb a function to its top. This technique requires repeatedly calculating the steepest direction, and the problem is that this calculation can be expensive. *Stochastic optimization* lets us use *cheaper approximate calculations*. It has transformed *modern* machine learning.

Large-Scale (Big Data) Problems

- Two definitions of large-scale problems,
  - It consists of problems where the main computational constraint is the amount of time available, rather than the number of instances [Bottou, 2008].
  - Training set may not be stored in modern computer's memory [Langford, 2008].
- We are in a need of learning algorithms that scale linearly with the size of datasets
- The performance of the algorithms should be better than processing a random subset of the data via conventional learning algorithms

Large-Scale (Big Data) Problems

- Two definitions of large-scale problems,
    - It consists of problems where the main computational constraint is the amount of time available, rather than the number of instances [Bottou, 2008].
    - Training set may not be stored in modern computer's memory [Langford, 2008].
- We are in a need of learning algorithms that scale linearly with the size of datasets
- The performance of the algorithms should be better than processing a random subset of the data via conventional learning algorithms

Outline

Online Learning

Definition of online learning

Given a set of new training data,

- Online learner can update its model without reading old data while improving its performance.
- In contrast, off-line learner must combine old and new data and start the learning all over again, otherwise the performance will suffer.

- Online is considered as a solution of large learning tasks
- Usually require several passes (or *epochs*) through the training instances
- Need to keep all instances unless we only run the algorithm one single pass

Outline

Perceptron Algorithm [Rosenblatt, 1956]

- An online learning algorithm and a mistake-driven procedure
- The current classifier is updated whenever the new arriving instance is misclassified

```
Initiation: $k = 0$ , $R = \max_{1 \le j \le m} \|\mathbf{x}^j\|_2$
repeat
        for $t = 1 : m$
            if $y_t(\langle \mathbf{w}^k, \mathbf{x}^t \rangle + b_k) \le 0$
                $\mathbf{w}^{k+1} = \mathbf{w}^k + \eta y_t \mathbf{x}^t$
                $b_{k+1} = b_k + \eta y_t R^2$
                $k = k + 1$
            end
        end
until no mistake made within the for-loop
```

- k is number of mistakes. $\eta > 0$ is the learning rate.

Perceptron Algorithm [Rosenblatt, 1956]

- The Perceptron is considered as a SGD method. The underlying optimization problem of the algorithm

$$\min_{(\mathbf{w},b)\in\mathbb{R}^{d+1}} \quad \sum_{j=1}^{m}(-y_j(\langle\mathbf{w},\mathbf{x}_j\rangle + b))_+$$

- In the linearly separable case, the Perceptron alg. will be terminated in finite steps no matter what learning rate is chosen
- In the nonseparable case, how to decide the appropriate learning rate that will make the least mistake is very difficult
- Learning rate, η, can be a nonnegative number. More general case, it can be a positive definite matrix

Outline

Key Idea of PA Algorithm, K. Crammer, et al.,2005

- The PA algorithm suggests that the new classifier should not only classify the new arriving data correctly but also *as close to the current classifier as possible*
- It can be formulated the problem as follows:

$$\begin{aligned} \mathbf{w}^{t+1} \in \quad & \underset{\mathbf{w}\in\mathbb{R}^d}{\arg\min} \quad \frac{1}{2}\|\mathbf{w}-\mathbf{w}^t\|_2^2 \\ & \text{s.t.} \qquad \ell(\mathbf{w};(\mathbf{x}^t, y_t)) = 0 \end{aligned} \tag{3}$$

where $\ell(\mathbf{w};(\mathbf{x}^t, y_t))$ is a hinge loss function

Simplify the PA Algorithm

- We can simplify Eq.(3) as follows:

$$\begin{aligned} \mathbf{w}^{t+1} \in \quad & \arg\min_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 - \langle \mathbf{w}, \mathbf{w}^t \rangle \\ & \text{s.t.} \qquad \ell(\mathbf{w}; (\mathbf{x}^t, y_t)) = 0 \end{aligned} \tag{4}$$

- In the Eq.(4), the PA algorithm implicitly minimizes the regularization term, $\frac{1}{2}\|\mathbf{w}\|_2^2$
- Minimizing $-\langle \mathbf{w}, \mathbf{w}^t \rangle$ is also expressed that the new updated classifier must be similar to the current classifier

PA-1 and PA-2

- Based on the concept of the *soft-margin* classifier, the non-negative slack variable ξ was introduced into the PA algorithm in two different ways
  - Linear scale with ξ (called *PA-1*)

$$\begin{aligned} \mathbf{w}^{t+1} \in \quad & \arg\min_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{2}\|\mathbf{w}-\mathbf{w}^t\|_2^2 + C\xi \\ & \text{s.t.} \qquad \ell(\mathbf{w};(\mathbf{x}^t, y_t)) \leq \xi \;\text{ and }\; \xi \geq 0 \end{aligned}$$

  - Square scale with ξ (called *PA-2*)

$$\begin{aligned} \mathbf{w}^{t+1} \in \quad & \arg\min_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{2}\|\mathbf{w}-\mathbf{w}^t\|_2^2 + C\xi^2 \\ & \text{s.t.} \qquad \ell(\mathbf{w};(\mathbf{x}^t, y_t)) \leq \xi \end{aligned}$$

PA Algorithm Closed Form Updating

- It seems that we have to solve an optimization problem for each instance. Fortunately, PA, PA-1 and PA-2 come with the closed form of updating schemes
- They share the same closed form $\mathbf{w}^{t+1} = \mathbf{w}^{t} + \tau_t y_t \mathbf{x}^{t}$ where $\tau_t > 0$ is defined as

$$\tau_t = \begin{cases} \frac{\ell(\mathbf{w}^t;\ (\mathbf{x}^t, y_t))}{\|\mathbf{x}^t\|_2^2} & \text{(PA)} \\ \min\{C, \frac{\ell(\mathbf{w}^t;\ (\mathbf{x}^t, y_t))}{\|\mathbf{x}^t\|_2^2}\} & \text{(PA-1)} \\ \frac{\ell(\mathbf{w}^t;\ (\mathbf{x}^t, y_t))}{\|\mathbf{x}^t\|_2^2 + \frac{1}{2C}} & \text{(PA-2)} \end{cases}$$

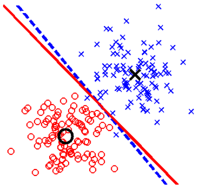
- It replaced the fixed learning rate such as Perceptron algorithm with the dynamic learning rate depending on the current instance

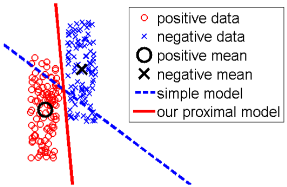
Outline

Motivation for a Proximal Classifier

- In online learning framework, we do not keep the previous instances in the memory
- A lack of memory for previous instances might hurt the learning efficiency
- The previous instances which have been classified correctly may be misclassified again
- We keep some simple statistical information (mean and variance) to summarize the previous instances
- Have to take the cost into account, fixed and small size memory and linear CPU time

Illustrations of Simple Proximal Models





- If the dataset is easy to separate, the means difference suggests a good proximal classifier (left figure)
- For the more complicated dataset, the performance of the LDA direction is better (right figure)
- The proximal classifier would provide a good suggestion for our classifier updating

Proximal Classifier: Quasi-LDA

- However, the cost of computing the LDA is expensive when the input space is in the high dimensional space
- We proposed the quasi-LDA direction as our proximal classifier

$$(\mathbf{w}_p^t)_i = \frac{(\mathbf{m}_+^t)_i - (\mathbf{m}_-^t)_i}{(\mathbf{s}_+^t)_i + (\mathbf{s}_-^t)_i}\ , \quad i = 1, 2, \ldots, d$$

where

$\mathbf{w}_p^t$	: the proximal classifier on round t
$\mathbf{m}_{+/-}^t$	: mean vector of Pos./Neg. class on round t
$\mathbf{s}_{+/-}^t$	: variance vector of Pos./Neg. class on round t

- It is very cheap both in CPU time and in memory usage

The Details of Proximal Classifier

Small Trick: $Var(X) = E((X - E(X))^2) = E(X^2) - (E(X))^2$

- The details about the statistical information we maintained

$$\begin{aligned} P^t &= \{j \mid y_j \in positive\ ,\ j = 1, 2, \dots, t\} \\ N^t &= \{j \mid y_j \in negative\ ,\ j = 1, 2, \dots, t\} \end{aligned}$$

$$\begin{aligned} (\mathbf{m}_+^t)_i &= \frac{1}{|P^t|}\sum_{j\in P^t}(\mathbf{x}^j)_i & i &= 1, 2, \dots, d \\ (\mathbf{m}_-^t)_i &= \frac{1}{|N^t|}\sum_{j\in N^t}(\mathbf{x}^j)_i & i &= 1, 2, \dots, d \\ (\mathbf{s}_+^t)_i &= \frac{1}{|P^t|-1}\sum_{j\in P^t}(\mathbf{x}^j)_i^2 - \frac{|P^t|}{|P^t|-1}(\mathbf{m}_+^t)_i^2 & i &= 1, 2, \dots, d \\ (\mathbf{s}_-^t)_i &= \frac{1}{|N^t|-1}\sum_{j\in N^t}(\mathbf{x}^j)_i^2 - \frac{|N^t|}{|N^t|-1}(\mathbf{m}_-^t)_i^2 & i &= 1, 2, \dots, d \end{aligned}$$

- All we need are $(\mathbf{m}_+^t)_i$, $(\mathbf{m}_-^t)_i$, $\sum\limits_{j\in P^t}(\mathbf{x}^j)_i^2$ and $\sum\limits_{j\in N^t}(\mathbf{x}^j)_i^2$ for each attribute in online fashion

PA Algorithm with a Proximal Model

How to combine the PA algorithm with the proximal classifier?

- Remember, in the PA algorithm, $\langle \mathbf{w}, \mathbf{w}^t \rangle$ is expressed as the similarity between $\mathbf{w}$ and $\mathbf{w}^t$
- Therefore, we can formulate our idea by adding $-\gamma\langle \mathbf{w}, \mathbf{w}_p^t \rangle$ into the Eq.(4), called PAm:

$$\begin{aligned} \mathbf{w}^{t+1} \in \quad & \arg\min_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 - \langle \mathbf{w}, \mathbf{w}^t \rangle - \gamma\langle \mathbf{w}, \mathbf{w}_p^t \rangle \\ & \text{s.t.} \qquad \ell(\mathbf{w}; (\mathbf{x}^t, y_t)) = 0 \end{aligned}$$

- Minimizing $-\langle \mathbf{w}, \mathbf{w}^t \rangle - \gamma\langle \mathbf{w}, \mathbf{w}_p^t \rangle$ is the way to keep the new classifier $\mathbf{w}$ to be close to $\mathbf{w}^t$ and $\mathbf{w}_p^t$
- $-\gamma\langle \mathbf{w}, \mathbf{w}_p^t \rangle$ also can be introduced into the PA-1 and PA-2, called PAm-1 and PAm-2, respectively

Closed Form Updating Rule of PAm Algorithm

We derive the closed form updating rules for PAm, PAm-1 and PAm-2 as follows, they also shared the same closed form

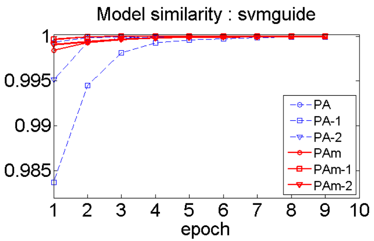
$$\mathbf{w}^{t+1} = \mathbf{w}^t + \gamma \mathbf{w}_p^t + \alpha_t y_t \mathbf{x}^t$$

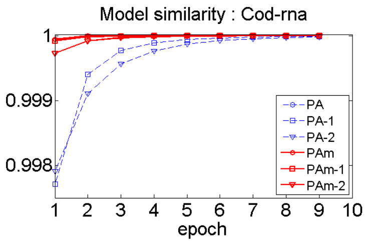
where $\alpha_t > 0$ is defined as

$$\alpha_t = \begin{cases} \frac{\ell(\mathbf{w}^t;(\mathbf{x}^t,y_t)) - \gamma y_t \langle \mathbf{w}_p^t, \mathbf{x}^t \rangle}{\|\mathbf{x}^t\|_2^2} & \text{(PAm)} \\ \min\{C, \frac{\ell(\mathbf{w}^t;(\mathbf{x}^t,y_t)) - \gamma y_t \langle \mathbf{w}_p^t, \mathbf{x}^t \rangle}{\|\mathbf{x}^t\|_2^2}\} & \text{(PAm-1)} \\ \frac{\ell(\mathbf{w}^t;(\mathbf{x}^t,y_t)) - \gamma y_t \langle \mathbf{w}_p^t, \mathbf{x}^t \rangle}{\|\mathbf{x}^t\|_2^2 + \frac{1}{2C}} & \text{(PAm-2)} \end{cases}$$

Convergence Behavior: PA vs. PAm

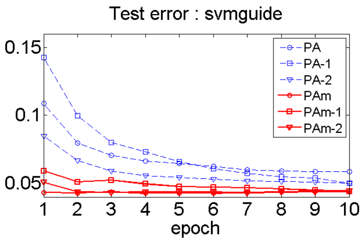
- Run 10 epochs with the same input order for each methods and record the classifier when an epoch is completed
- The proximal classifier will not be changed once the first epoch is finished
- We compute the cosine similarity between two consecutive epoch classifiers

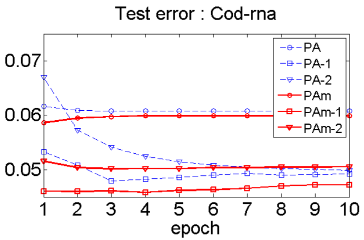




Test Error Rate of each Epoch

- It becomes a constant after 2 to 3 epochs

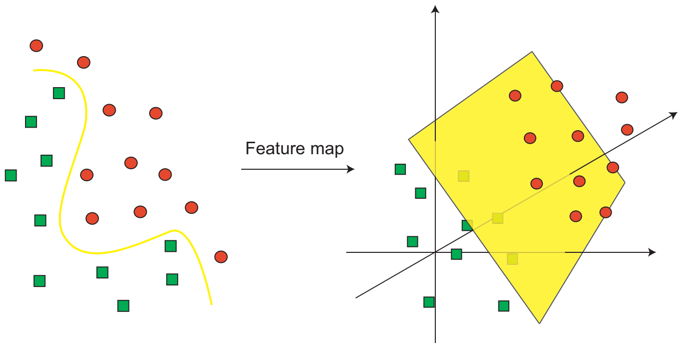




Outline

The Illustration of Nonlinear SVM

$$X \overset{\Phi}{\dashrightarrow} F$$



nonlinear pattern in data space

approximate linear pattern in feature space

Kernel Trick

- We can use the value of kernel function to represent the inner product of two training points in feature space as follows:
$$K(\mathbf{x},\mathbf{z}) = <\phi(\mathbf{x}),\phi(\mathbf{z})> .$$
- The most popular kernel function is the Gaussian kernel
$$K(\mathbf{x},\mathbf{z}) = e^{-\gamma\|\mathbf{x}-\mathbf{z}\|_2^2}.$$
- The kernel matrix $K(A,A^\top)_{m\times m}$ represents the inner product of all points in the feature space where $K(A,A^\top)_{ij} = K(\mathbf{x}_i,\mathbf{x}_j)$.
- Replace $AA^\top$ by a nonlinear kernel $K(A,A^\top)$ without defining an explicit feature map ϕ

Nonlinear SVM: A Full Model

- Nonlinear SVM classifier: $f(x) = \sum_{i=1}^{m} \alpha_i k(x, A_i) + b$
  - As many parameters α_i as the data points
- Nonlinear SVM is a linear combination of basis functions,

$$\mathcal{B} = \{1\} \cup \left\{ k(\cdot, x^i) \right\}_{i=1}^{m}$$

  $\mathcal{B}$ is an overcomplete dictionary of functions when m is large
- Fitting data to an overcomplete full model may
  - Increase computational difficulties model complexity
  - Need more CPU time and memory space
  - Be in danger of overfitting

Reduced SVM: A Compressed Model

It's desirable to cut down the model complexity

- Reduced SVM randomly selects a small subset $\widetilde{S}$ to generate the basis functions $\widetilde{\mathcal{B}}$:
  $\widetilde{S} = \{(\widetilde{x}^i, \widetilde{y}_i) \big| i = 1, \dots, \widetilde{m}\} \subseteq S, \overline{\mathcal{B}} = \{1\} \cup \left\{k(\cdot, \widetilde{x}^i)\right\}_{i=1}^{\widetilde{m}}$
- RSVM classifier is in the form $f(x) = \sum_{i=1}^{\widetilde{m}} \widetilde{u}_i k(x, \widetilde{x}^i) + b$
- The parameters are determined by fitting entire data

$$\min_{\widetilde{u}, b, \xi \geqslant 0} \quad C \sum_{j=1}^{m} \xi_j + \frac{1}{2}(\|\widetilde{u}\|_2^2 + b^2)$$

$$\text{s.t.} \qquad D(K(A, \widetilde{A}^\top)\widetilde{u} + \mathbf{1}b) + \xi \geqslant \mathbf{1}$$

Nonlinear SVM vs. RSVM

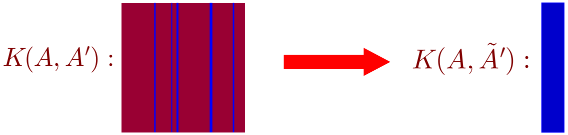
$K(A,A^\top)\in\mathbb{R}^{m\times m}$ vs. $K(A,\widetilde{A}^\top)\in\mathbb{R}^{m\times\widetilde{m}},\ m>>\widetilde{m}$

Nonlinear SVM

$$\min_{u,b,\xi\geqslant 0}\ C\sum_{j=1}^{m}\xi_j+\frac{1}{2}(\|u\|_2^2+b^2)$$
$$D(K(A,A^\top)u+\mathbf{1}b)+\xi\geqslant\mathbf{1}$$
where $K(A,A^\top)_{ij}=k(x^i,x^j)$

RSVM

$$\min_{\widetilde{u},b,\xi\geqslant 0}\ C\sum_{j=1}^{m}\xi_j+\frac{1}{2}(\|\widetilde{u}\|_2^2+b^2)$$
$$D(K(A,\widetilde{A}^\top)\widetilde{u}+\mathbf{1}b)+\xi\geqslant\mathbf{1}$$
where $K(A,\widetilde{A}^\top)_{ij}=k(x^i,\widetilde{x}^j)$



Represent **w** in the Dual Form

- From the dual form of SVM, the normal vector **w** can be expressed in terms of the data points, i.e.,

$$\mathbf{w} = \sum_{i=1}^{m} (\mathbf{u})_i \mathbf{x}^i = A^\top \mathbf{u},$$

  where $A = [\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^m]^\top$ is the training data matrix.
- Now, we only need to find the solution of **u** and b.
- We let $\mathbf{w}^t = A^\top \mathbf{u}^t$ and b_t be the current classifier

The Passive and Aggressive Algorithm in the Dual Form

- We substitute $\mathbf{w}$ and $\mathbf{w}^t$ in the minimization problem of PA,

$$\min_{(\mathbf{u},b)\in\mathbb{R}^{m+1}} \quad \frac{1}{2}(\mathbf{u}^\top AA^\top\mathbf{u} + b^2) - ({\mathbf{u}^t}^\top AA^\top\mathbf{u} + bb_t) \qquad (5)$$

$$\text{s.t.} \qquad 1 - y_t(\mathbf{u}^\top A\mathbf{x}^t + b) \leq 0,$$

- The PA can be reformulated in terms of inner products between the data points.
- We can extend to the nonlinear version by utilizing the “kernel trick”.

Kernel PA Closed Form Updating

We derive the closed form updating rules for KPA as follows:

$$\begin{aligned} \mathbf{u}^{t+1} &= \mathbf{u}^t + \alpha_t y_t K(A, A^\top)^{-1} K(A, \mathbf{x}^t) \\ b_{t+1} &= b_t + \alpha_t y_t, \end{aligned} \tag{6}$$

where ℓ_t is the loss suffered on round t and α_t is defined as

$$\alpha_t = \frac{\ell_t}{K(A, \mathbf{x}^t)^\top K(A, A^\top)^{-1} K(A, \mathbf{x}^t) + 1}$$

Can't be used in *online manner*

Reduced Kernel Trick can help here!

Kernel PA Closed Form Updating

We derive the closed form updating rules for KPA as follows:

$$\begin{aligned} \mathbf{u}^{t+1} &= \mathbf{u}^{t} + \alpha_t y_t K(A, A^\top)^{-1} K(A, \mathbf{x}^t) \\ b_{t+1} &= b_t + \alpha_t y_t, \end{aligned} \tag{6}$$

where ℓ_t is the loss suffered on round t and α_t is defined as

$$\alpha_t = \frac{\ell_t}{K(A, \mathbf{x}^t)^\top K(A, A^\top)^{-1} K(A, \mathbf{x}^t) + 1}$$

Can't be used in *online manner*

Reduced Kernel Trick can help here!

Sketch of Reduced Kernel PA

- Preselect a small subset $\widetilde{S} = \{(\widetilde{x}^i, \widetilde{y}_i) \big| i = 1, \ldots, \widetilde{m}\}$
- Generate $K(\widetilde{A}, \widetilde{A}^\top)^{-1}$ and substitute $K(A, A^\top)^{-1}$ in KPA
- For numerical robustness, we can add a small regularization term $\epsilon\mathbf{I}$,

$$K(\widetilde{A}, \widetilde{A}^\top)_\epsilon = K(\widetilde{A}, \widetilde{A}^\top) + \epsilon\mathbf{I},$$

- Diagonalize $K(\widetilde{A}, \widetilde{A}^\top)_\epsilon^{-1} = PVP^\top = (PV^{\frac{1}{2}})(PV^{\frac{1}{2}})^\top$
- Change variables, letting

$$\mathbf{z} = (PV^{\frac{1}{2}})^\top \widetilde{\mathbf{u}}, \quad \mathbf{z}^t = (PV^{\frac{1}{2}})^\top \widetilde{\mathbf{u}}^t,$$

and

$$\widehat{K}(\widetilde{A}, \mathbf{x}^t) = (PV^{\frac{1}{2}})^\top K(\widetilde{A}, \mathbf{x}^t).$$

Final Formulation of RKPA

- Then the Reduced Kernel PA can be rewritten as follows:

$$\min_{(\mathbf{z},b)\in\mathbb{R}^{\widetilde{m}+1}} \quad \frac{1}{2}[\mathbf{z}^\top \mathbf{z} + b^2] - [{\mathbf{z}^t}^\top \mathbf{z} + bb_t] \tag{7}$$

$$\text{s.t.} \qquad 1 - y_t[\mathbf{z}^\top V^{-1}\widehat{K}(\widetilde{A},\mathbf{x}^t) + b] \leq 0,$$

and the decision function becomes

$$f(\mathbf{x}) = \mathbf{z}^\top V^{-1}\widehat{K}(\widetilde{A},\mathbf{x}^t) + b.$$

- It is straightforward to obtain the minimization problems defining Reduced Kernel PA-1 (RKPA-1) and Reduced Kernel PA-2 (RKPA-2).

RKPA Updating Rules:

- $\mathbf{z}^{t+1} = \mathbf{z}^t + \alpha_t y_t V^{-1}\widehat{K}(\widetilde{A}, \mathbf{x}^t),\ b_{t+1} = b_t + \alpha_t y_t$

$$\alpha_t = \begin{cases} \dfrac{\ell_t}{\left\|V^{-1}\widehat{K}(\widetilde{A}, \mathbf{x}^t)\right\|^2 + 1} & \text{(RKPA)} \\ \min\{C, \dfrac{\ell_t}{\left\|V^{-1}\widehat{K}(\widetilde{A}, \mathbf{x}^t)\right\|^2 + 1}\} & \text{(RKPA-1)} \\ \dfrac{\ell_t}{\left\|V^{-1}\widehat{K}(\widetilde{A}, \mathbf{x}^t)\right\|^2 + 1 + \frac{1}{2C}} & \text{(RKPA-2)} \end{cases}, \qquad (8)$$

  where ℓ_t is the loss suffered on round t.

- We no longer need to calculate the inverse of $K(\widetilde{A}, \widetilde{A}^\top)$, only that of V, the inverse of V is trivial to calculate.

Reduced Kernel Fisher Discriminant Analysis (RKFDA)

- KFDA is a kernelized version of LDA.
- To be able to handle large-scale datasets, we also introduce the reduced kernel trick, then the solution of RKFDA is of the form

$$\tilde{\mathbf{u}}_p = [\mathrm{COV}(K(A_p, \tilde{A}^\top)) + \mathrm{COV}(K(A_N, \tilde{A}^\top))]^{-1}(\widetilde{M_P} - \widetilde{M_N}),$$

where A_P and A_N are, respectively, the positive data and negative data;, $\widetilde{M_P}$ and $\widetilde{M_N}$ are their respective mean vectors of reduced kernel data.

RKPAm: RKPA+quasi-KFD

$$\begin{aligned} \mathbf{z}^{t+1} &= \mathbf{z}^t + \gamma \mathbf{z}_p^t + \alpha_t y_t V^{-1} \widehat{K}(\widetilde{A}, \mathbf{x}^t) \\ b_{t+1} &= b_t + \gamma b_p^t + \alpha_t y_t \end{aligned} \tag{9}$$

$$\alpha_t = \begin{cases} \dfrac{\ell_t - y_t \gamma \left({\mathbf{z}_p^t}^\top V^{-1} \widehat{K}(\widetilde{A}, \mathbf{x}^t) + b_p^t \right)}{\left\| V^{-1} \widehat{K}(\widetilde{A}, \mathbf{x}^t) \right\|^2 + 1} & \text{(RKPAm)} \\ \min\{C, \dfrac{\ell_t - y_t \gamma \left({\mathbf{z}_p^t}^\top V^{-1} \widehat{K}(\widetilde{A}, \mathbf{x}^t) + b_p^t \right)}{\left\| V^{-1} \widehat{K}(\widetilde{A}, \mathbf{x}^t) \right\|^2 + 1}\} & \text{(RKPAm-1)} \\ \dfrac{\ell_t - y_t \gamma \left({\mathbf{z}_p^t}^\top V^{-1} \widehat{K}(\widetilde{A}, \mathbf{x}^t) + b_p^t \right)}{\left\| V^{-1} \widehat{K}(\widetilde{A}, \mathbf{x}^t) \right\|^2 + 1 + \frac{1}{2C}} & \text{(RKPAm-2)} \end{cases}$$

Outline

Experimental Objective and Setting

- Objective:
  - Linear model (PA) vs. Nonlinear model (RKPA)
  - Sensitive to the input order.
- In our experiment, the results of the RKPA, RKPA-1 and RKPA-2 were compared with the PA, PA-1 and PA-2, respectively.
- We run a single pass of the PA algorithm and RKPA algorithm 10 times with different input orders each time.
- For nonlinear model, we use the Gaussian kernel.
- We compare the results on 8 datasets. The sizes of these datasets range from small-scale, medium-scale to large-scale. Table 1 summarizes the statistics of the datasets.

The Summary of Datasets

Table: The statistics of the datasets used in the experiment.

Dataset	Training	Testing	Features
svmguide1	3,089	4,000	4
w3a	4,912	44,837	300
a9a	32,561	16,281	123
ijcnn1	35,000	91,701	22
Cod-rna	59,535	271,617	8
usps01	266,079	75,383	676
covertype	522,910	58,102	54
Checkerboard	1,000,000	2,000	2

Comparison of Testing Error Rate over 10 Runs

Table: The comparison of average testing error rate and the standard deviation (%).

	Linear SSVM	PA		RKPA		PA-1		RKPA-1		PA-2		RKPA-2	
dataset	batch	avg	(std)	avg	(std)	avg	(std)	avg	(std)	avg	(std)	avg	(std)
svmguide1	4.33	14.21	(8.04)	**4.37**	(**0.70**)	10.00	(6.59)	**3.62**	(**0.15**)	10.05	(5.17)	**3.61**	(**0.27**)
w3a	1.67	10.49	(11.60)	**2.93**	(**0.50**)	**2.14**	(**0.08**)	2.83	(0.11)	**2.11**	(**0.08**)	2.81	(0.13)
a9a	14.88	21.01	(**3.56**)	**19.54**	(3.62)	15.25	(**0.14**)	**15.19**	(0.29)	15.22	(**0.15**)	**15.15**	(0.31)
ijcnn1	8.68	14.10	(4.86)	**6.58**	(**1.48**)	8.67	(**0.15**)	**5.00**	(0.78)	8.97	(**0.05**)	**5.21**	(0.80)
Cod-rna	4.83	10.33	(8.07)	**6.24**	(**2.01**)	5.57	(0.81)	**4.28**	(**0.48**)	6.14	(1.67)	**4.46**	(**0.61**)
usps01	3.35	4.88	(0.79)	**0.57**	(**0.07**)	3.62	(0.08)	**0.57**	(**0.07**)	3.61	(**0.07**)	**0.57**	(**0.07**)
covertype	24.52	34.56	(3.62)	**12.04**	(**2.10**)	24.11	(**0.20**)	**10.77**	(0.66)	24.81	(**0.19**)	**10.96**	(0.89)
Checkerboard	50.65	50.83	(2.33)	**1.07**	(**0.23**)	48.63	(2.81)	**0.90**	(**0.11**)	48.22	(2.58)	**0.83**	(**0.14**)

Comparison of Testing Error Rate over 10 Runs: RKPA vs. RKPAm

Table: The comparison of average testing error rate and the standard deviation (%).

	RKPA		RKPAm		RKPA-1		RKPAm-1		RKPA-2		RKPAm-2	
dataset	avg	(std)	avg	(std)	avg	(std)	avg	(std)	avg	(std)	avg	(std)
svmguide1	4.37	(0.70)	**3.73**	(**0.54**)	3.62	(**0.15**)	**3.43**	(**0.15**)	3.61	(0.27)	**3.36**	(**0.22**)
w3a	2.93	(**0.50**)	**2.71**	(0.57)	2.83	(**0.11**)	**2.62**	(0.12)	2.81	(**0.13**)	**2.63**	(0.16)
a9a	19.54	(**3.62**)	**19.51**	(3.69)	15.19	(0.29)	**15.17**	(**0.22**)	15.15	(0.31)	**15.14**	(**0.30**)
ijcnn1	6.58	(1.48)	**3.63**	(**0.69**)	5.00	(0.78)	**3.64**	(**0.73**)	5.21	(0.80)	**3.64**	(**0.73**)
Cod-rna	6.24	(2.01)	**5.64**	(**1.15**)	4.28	(0.48)	**4.23**	(**0.47**)	4.46	(0.61)	**4.38**	(**0.59**)
usps01	0.57	(0.07)	**0.54**	(**0.06**)	0.57	(0.07)	**0.54**	(**0.06**)	0.57	(0.07)	**0.54**	(**0.06**)
covertype	**12.04**	(**2.10**)	15.10	(3.97)	10.77	(0.66)	**10.38**	(**0.55**)	10.96	(0.89)	**10.74**	(**0.83**)
Checkerboard	1.07	(0.23)	**0.82**	(**0.18**)	0.90	(**0.11**)	**0.83**	(0.17)	0.83	(**0.14**)	**0.78**	(**0.14**)

Comparison of Average Running Time over 10 Runs

Table: Comparison of average training time (sec.).

	RKPA	RKPAm	RKPA-1	RKPAm-1	RKPA-2	RKPAm-2
svmguide1	0.53	0.57	0.50	0.59	0.52	0.62
w3a	0.66	0.74	0.66	0.79	0.68	0.80
a9a	1.76	3.56	1.68	3.43	1.74	4.20
ijcnn1	4.46	5.54	4.42	5.52	4.53	5.84
Cod-rna	7.57	9.64	7.24	9.85	7.43	11.81
usps01	52.11	58.01	53.07	57.41	52.52	57.51
covertype	5127.77	5294.39	5128.12	5357.68	5131.65	5398.59
Checkerboard	12.81	16.38	13.21	16.80	13.92	18.75

Outline

Conclusions

- We used the online learning framework to solve large-scale binary classification problem
- We derived a set of update schemes (linear and nonlinear) in closed form for PAm, RKPAm
- Compared to the conventional PA algorithm, utilizing the proximal classifier will have
    - Less sensitive to the input order
    - Less number of updating made in a single pass
    - Higher similarity between two consecutive epochs resulting classifiers
- We can have a near-optimal classifier in a single pass

Conclusions

- We used the online learning framework to solve large-scale binary classification problem
- We derived a set of update schemes (linear and nonlinear) in closed form for PAm, RKPAm
- Compared to the conventional PA algorithm, utilizing the proximal classifier will have
  - Less sensitive to the input order
  - Less number of updating made in a single pass
  - Higher similarity between two consecutive epochs resulting classifiers
- We can have a near-optimal classifier in a single pass

Conclusions

- We used the online learning framework to solve large-scale binary classification problem
- We derived a set of update schemes (linear and nonlinear) in closed form for PAm, RKPAm
- Compared to the conventional PA algorithm, utilizing the proximal classifier will have
  - Less sensitive to the input order
  - Less number of updating made in a single pass
  - Higher similarity between two consecutive epochs resulting classifiers
- We can have a near-optimal classifier in a single pass

Conclusions

- We used the online learning framework to solve large-scale binary classification problem
- We derived a set of update schemes (linear and nonlinear) in closed form for PAm, RKPAm
- Compared to the conventional PA algorithm, utilizing the proximal classifier will have
  - Less sensitive to the input order
  - Less number of updating made in a single pass
  - Higher similarity between two consecutive epochs resulting classifiers
- We can have a near-optimal classifier in a single pass

Conclusions

- We used the online learning framework to solve large-scale binary classification problem
- We derived a set of update schemes (linear and nonlinear) in closed form for PAm, RKPAm
- Compared to the conventional PA algorithm, utilizing the proximal classifier will have
  - Less sensitive to the input order
  - Less number of updating made in a single pass
  - Higher similarity between two consecutive epochs resulting classifiers
- We can have a near-optimal classifier in a single pass

Conclusions

- We used the online learning framework to solve large-scale binary classification problem
- We derived a set of update schemes (linear and nonlinear) in closed form for PAm, RKPAm
- Compared to the conventional PA algorithm, utilizing the proximal classifier will have
  - Less sensitive to the input order
  - Less number of updating made in a single pass
  - Higher similarity between two consecutive epochs resulting classifiers
- We can have a near-optimal classifier in a single pass

Conclusions

- We used the online learning framework to solve large-scale binary classification problem
- We derived a set of update schemes (linear and nonlinear) in closed form for PAm, RKPAm
- Compared to the conventional PA algorithm, utilizing the proximal classifier will have
    - Less sensitive to the input order
    - Less number of updating made in a single pass
    - Higher similarity between two consecutive epochs resulting classifiers
- We can have a near-optimal classifier in a single pass

Reference

- Léon Bottou and Olivier Bousquet. The Tradeoffs of Large Scale Learning. *Advances in Neural Information Processing Systems*, 20:161–168, 2008
- John Langford. *Concerns about the Large Scale Learning Challenge*, 2008
- K. Crammer et al. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.
- K. Hiraoka et al. Convergence analysis of online linear discriminant analysis. *In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 3:387–391, Como, Italy, 2000.
- L. I. Kuncheva and C. O. Plumpton. Adaptive learning rate for online linear discriminant classifiers. *Structural, Syntactic, and Statistical Pattern Recognition*, 5342:510–519, 2008.

Q & A

Q & A

Thank you !