THU-70250043, Pattern Recognition (Spring 2016) **Homework: 4**

Linear Classifiers and SVM

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Student: Qingfu Wen wqf15@mails.tsinghua.edu.cn

Problem 1

Suppose we have N points x_i in $\mathcal{R}^p$ in general position, with class labels $y_i \in \{-1, 1\}$. Prove that the perceptron learning algorithm converges to a separating hyperplane in a finite number of steps:

- Denoting a hyperplane by $f(x) = \beta_1^T(x) + \beta_0 = 0$, or in more compact notation $\beta^T x^* = 0$, where $x^* = (x, 1)$ and $\beta = (\beta_1, \beta_0)$. Let $z_i = \frac{x_i^*}{||x_i^*||}$. Show that separability implies the existence of a β_{sep} such that $y_i \beta_{sep}^T z_i \geq 1, \forall i$
- Given a current β_{old}, the perceptron algorithm identifies a point z_i that is misclassified, and produces the update $\beta_{new} = \beta_{old} + y_i z_i$. Show that $||\beta_{new} - \beta_{sep}||^2 \leq ||\beta_{old} - \beta_{sep}||^2 - 1$, and hence the algorithm converges to a separating hyperplane in no more than $||\beta_{start} - \beta_{sep}||^2$ steps. (Ripley, 1996)

SOLUTION:

1. when $\beta^T x^* > 0$, $y_i = 1$, $\beta^T x_i^* < 0$, $y_i = -1$. Thus, $\forall i, y_i \beta^T x_i^* > 0$.
   $\exists \epsilon > 0, \forall i, y_i \beta^T x_i^* \geq \epsilon$

$$\begin{aligned} y_i \beta^T x_i^* &\geq \epsilon \\ y_i ||x_i^*|| \beta^T z_i &\geq \epsilon \\ y_i \frac{||x_i^*||}{\epsilon} \beta^T z_i &\geq 1 \\ y_i \beta_{sep}^T z_i &\geq 1 \end{aligned}$$

2.

$$\begin{aligned} ||\beta_{new} - \beta_{sep}||^2 &= ||\beta_{new}||^2 + ||\beta_{sep}||^2 - 2\beta_{sep}^T \beta_{new} \\ &= ||\beta_{old} + y_i z_i||^2 + ||\beta_{sep}||^2 - 2\beta_{sep}^T (\beta_{old} + y_i z_i) \\ &= ||\beta_{old}||^2 + ||y_i z_i||^2 + 2y_i \beta_{old}^T z_i + ||\beta_{sep}||^2 - 2\beta_{sep}^T \beta_{old} - 2y_i \beta_{sep}^T z_i \\ &\leq ||\beta_{old}||^2 + ||\beta_{sep}||^2 - 2\beta_{sep}^T \beta_{old} + 1 - 2 \\ &= ||\beta_{old} - \beta_{sep}||^2 - 1 \end{aligned}$$

Suppose that the algorithm converges in k steps. Thus, we have $||\beta_{new} - \beta_{sep}||^2 = 0$

$$0 = ||\beta_{new} - \beta_{sep}||^2 \leq ||\beta_{new-1} - \beta_{sep}||^2 - 1 \leq ||\beta_{start} - \beta_{sep}||^2 - k$$

Thus

$$k \leq ||\beta_{start} - \beta_{sep}||^2$$

the algorithm converges in at most $||\beta_{start} - \beta_{sep}||^2$ steps.

Problem 2

In the multicategory case (c classes):

- A set of samples is said to be **linearly separable** if there exists a **linear machine (线性机器)** that can classify them correctly. If any samples labeled ω_i can be separated from all others by a single hyperplane, we shall say the samples are **totally linearly separable**. Give examples to show that totally linear separable must be linearly separable, but that the converse need not to be true.
- A set of samples is said to be **pairwise linearly separable** if there exist $c(c-1)/2$ hyperplanes H_{ij} such that H_{ij} separates samples labeled ω_i from samples labeled ω_j. Give examples to show that a pairwise linearly separable set of patterns may not be linearly separable.

SOLUTION:

1. we can easily see that totally linear separable must be linearly separable. for each label ω_i, there is a $f_i(x)$ which can separate it from others. Thus, all these $f_i(x)$ are formed to be a linear machine to classify them. On the contrary, linearly separable samples may not be totally linearly separable. For example,

$$\omega_1 = x < 0$$
$$\omega_2 = 0 < x < 1$$
$$\omega_3 = x > 1$$

   There is no linear function to separate ω_2 from others, but it can be linear separable by $x = 0$ and $x = 1$.

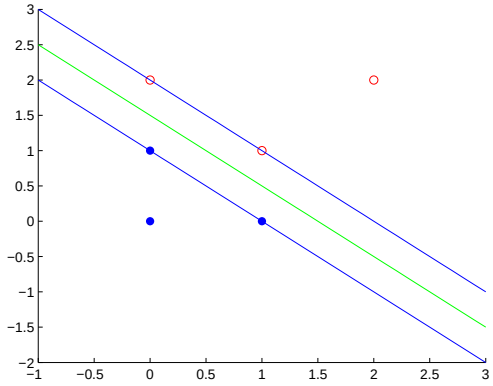
2. Pairwise linearly separable samples must be linearly separable by a linear machine.

Problem 3

Consider an SVM and the following training data from two categories:

Category	x_1	x_2
ω_1	0	0
ω_1	0	1
ω_1	1	0
ω_2	1	1
ω_2	2	2
ω_2	0	2

- Plot these six training points and construct by inspection the weight vector for the optimal hyperplane and the optimal margin.
- What are the support vectors?
- Construct the solution in the dual space by finding the Lagrange undetermined multipliers α_i. Compare your result to that in problem 3. 1.



1. From the above picture, we can see that the hyperplane is $x_1 + x_2 - 1.5 = 0$, so the weight vector can be $(-1.5, 1, 1)^T$.
2. The support vectors are $(0,1), (1,0), (1,1), (0,2)$.
3. we can try to solve the dual problem

$$\max_{\alpha} L(\alpha) = \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j x_i^T x_j$$

$$s.t.$$

$$\sum_{i=1}^{n} \alpha_i d_i = 0$$

$$a_i \geq 0, i = 1 \cdots n$$

First, we try to maximize $L(\alpha)$. From $\sum_{i=1}^{n} a_i d_i = 0$, we get $\alpha_6 = \alpha_1 + \alpha_2 + \alpha_3 - \alpha_4 - \alpha_5$. We replace this into $L(\alpha)$ and compute the partial derivative $\frac{\partial L(\alpha)}{\partial \alpha_i}$:

$$\begin{pmatrix} -1 & -2 & -2 & 0 & 1 \\ -2 & -5 & 2 & -1 & 1 \\ -2 & 2 & 5 & 1 & 1 \\ 0 & -1 & 1 & -1 & -1 \\ 1 & 1 & 3 & -1 & -2 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{pmatrix} = \begin{pmatrix} -2 \\ -2 \\ -2 \\ 0 \\ 0 \end{pmatrix}$$

Solving this, we get $\alpha = (3.2\ 0\ 0.8\ 0\ 2.8\ 1.2)^T$. Now, we can minimize $L(w)$

$$L(w) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i (d_i (w^T x_i + b) - 1)$$

$$\frac{\partial L(w)}{w} = w - \sum_{i=1}^{n} \alpha_i d_i x_i = 0$$

$$w = 3.2x_1 + 0.8x_3 - 2.8x_5 - 1.2x_6 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

since $\alpha_1 > 0$, $d_1(w^T x_1 + b) - 1 = 0$. $b = 1 - w^T x_1 = -3$. Thus, the solution is:

$$2x_1 + 2x_2 - 3 = 0$$

It is the same as the result in 3.1.

Problem 4

Convert the soft-marginal SVM

$$\begin{aligned} \min_{\omega} \quad & \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & y_i\omega^T x_i \geq 1-\xi_i, \quad i=1,\ldots,n \\ & \xi_i \geq 0, \quad i=1,\ldots,n \end{aligned}$$

into an unconstrained form and give a geometry interpretation.

SOLUTION:

Directly, the dual problem is:

$$L(w,b,\xi,\alpha,\beta) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(d_i(w^T x_i + b) - 1 + \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i$$

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{n}\alpha_i d_i x_i$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n}\alpha_i d_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i - \xi_i = 0, i = 1, \cdots, n$$

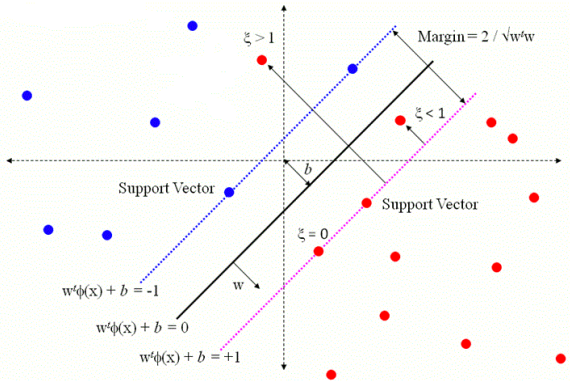
Thus, $L(w,b,\xi,\alpha,\beta)$ can be simplified as follow:

$$\max_{\alpha} L(\alpha) = \max_{\alpha}\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j d_i d_j x_i^T x_j$$

$$s.t.$$

$$\sum_{i=1}^{n}\alpha_i d_i = 0$$

$$0 \leq a_i \leq C, i = 1 \cdots n$$



From the above picture, we can see that $\xi = 0$ means the point is on the support vector, $0 < \xi < 1$ means the point is in the margin, $\xi \geq 1$ means it is misclassified.

Problem 5

Let there be a binary classification problem with $y \in \{0, 1\}$. The perceptron uses hypotheses of the form $h_\theta(x) = g(\theta^T x)$, where $g(z) = 1\{z \geq 0\}$. In this problem we will consider a **single-sample-correction-like** implementation of the perceptron algorithm where each update to the parameters θ is made using only one training example. The perceptron algorithm will only make one pass through the entire training set. The update rule for this version of the perceptron algorithm is given by

$$\theta^{(i+1)} := \theta^{(i)} + \alpha[y^{(i+1)} - h_{\theta^{(i)}}(x^{(i+1)})]x^{(i+1)},$$

where $\theta^{(i)}$ is the value of the parameters after the algorithm has seen the first i training examples. Prior to seeing any training examples, $\theta^{(0)}$ is initialized to $\vec{0}$ (This means that θ will always be a linear combination of the $\phi(x^{(i)})$, i.e., $\theta^{(i)} = \sum_{l=1}^{i} \beta_l \phi(x^{(l)})$ after incorporated i training points)

Let K be a Mercer kernel corresponding to some very high-dimensional feature mapping ϕ. Suppose ϕ is so high-dimensional (say, ∞-dimension) that it's infeasible to ever represent $\phi(x)$ explicitly. Describe how you would apply the "kernel trick" we have seen in SVM to the perceptron to make it work in the high-dimensional feature space ϕ, but **without ever explicitly computing** $\phi(x)$. Your description should specify:

- How you will (implicitly) represent the high-dimensional parameter vector $\theta^{(i)}$;
- How you will efficiently make a prediction on a new input $x^{(i+1)}$, i.e., how you will compute $h_{\theta^{(i)}}(x^{(i+1)}) = g({\theta^{(i)}}^T \phi(x^{(i+1)}))$, using your representation of $\theta^{(i)}$;
- How you will modify the update rule given above to perform an update to θ on a new training example $(x^{(i+1)}, y^{(i+1)})$; i.e., using the update rule corresponding to the feature mapping ϕ:

  $$\theta^{(i+1)} := \theta^{(i)} + \alpha[y^{(i+1)} - h_{\theta^{(i)}}(\phi(x^{(i+1)}))]\phi(x^{(i+1)})$$

- 在高维空间中，

  $$\theta^{(i+1)} = \theta^{(i)} + \alpha[y^{(i+1)} - h_{\theta^{(i)}}(\phi(x^{(i+1)}))]\phi(x^{(i+1)})$$

  假设θ 初值为0，θ 可以用$\phi(x^{(i+1)})$ 的线性组合来表示，因此我们只需要存储线性组合的系数β_l就可以表示θ了。

- 预测过程就是要计算$h_{\theta^{(i)}}(x^{(i+1)})$，也就是$g({\theta^{(i)}}^T \phi(x^{(i+1)}))$。对于计算$g({\theta^{(i)}}^T \phi(x^{(i+1)}))$，

  $$g({\theta^{(i)}}^T \phi(x^{(i+1)})) = g(\sum_{l=1}^{i} \beta_l \phi(x^{(l)})^T \phi(x^{(i+1)})) = g(\sum_{l=1}^{i} \beta_l K(x^{(l)}, x^{(i+1)}))$$

  将向量乘积转化为kernel函数就可以快速计算出结果。

- 对于新的$(x^{(i+1)}, y^{(i+1)})$，需要迭代更新θ，要算出每次$\theta^{(i+1)}$所对应的β_{i+1}.

  $$\beta_{i+1} = \alpha(y^{(i+1)} - h_{\theta^{(i)}}(\phi(x^{(i+1)})))$$

  当分类正确的时候，$y^{(i+1)} - h_{\theta^{(i)}}(\phi(x^{(i+1)})) = 0$，否则等于$2y^{(i+1)}$。因此

  $$\theta^{(i)} = \sum_{i \in Misclassified} 2y^{(i)}\phi(x^{(i)})$$

Problem 6

经典感知器（Algorithm 1）的训练过程可以看成在解区内寻找一个解，并没有对这个解的性能有所限定。这个解只需满足 $\alpha^T y_n > 0$，其中 α是感知器的权向量，y_n是规范化增广样本向量。而 margin 感知器（Algorithm 2）则要求算法收敛的超平面有一个大于 γ 的 margin，其中 γ 是预先设定的一个正数。即， margin 感知器的解需要满足 $\alpha^T y_n > \gamma$。

Algorithm 1 Fixed-Increment Single Sample Correction Algorithm

initialize α,$k \leftarrow 0$
repeat
  $k \leftarrow (k+1)\ mod\ n$
  if y_k is misclassified by α **then**
    $\alpha \leftarrow \alpha + y_k$
  end if
until all patterns are properly classified
return α

因此，在 margin 感知器的训练中，“错误”包括两种情况：1）标签预测错误（prediction mistake）；2）标签预测正确但 margin不够大（margin mistake）。

Algorithm 2 Single Sample Correction Algorithm With Margin

initialize $\alpha, k \leftarrow 0,\ \gamma > 0$
repeat
  $k \leftarrow (k+1)\ mod\ n$
  if $\alpha^T y_k \leq \gamma$ **then**
    $\alpha \leftarrow \alpha + y_k$
  end if
until all patterns are properly classified with a large enough margin γ
return α

- 随机生成 200 个二维平面上的点，其中 100 个标记为 1，剩下的 100 个标记为 -1，保证它们是线性可分的。画出这 200 个点的分布。
- 编程实现经典感知器算法，并在生成的数据集上运行。在一张图上画出分界线和数据点。
- 编程实现 margin 感知器算法，并在生成的数据集上运行。在一张图上画出分界线和数据点，分析γ 取值对算法收敛性及分界面位置的影响。

γ	time(s)
0.1	0.130
1	0.302
10	1.874
100	15.206

Table 1: problem 6-3

From the above table and figure, we can see that when γ gets bigger, training time is longer but the performance of algorithm are better(larger γ with larger margin).

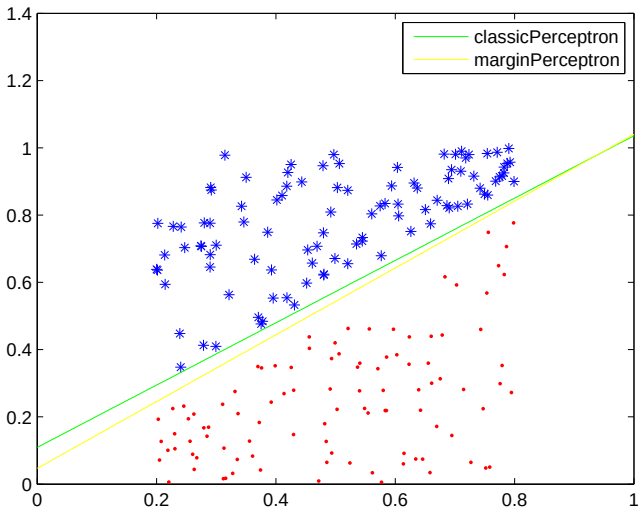


Figure 1: problem 6-2

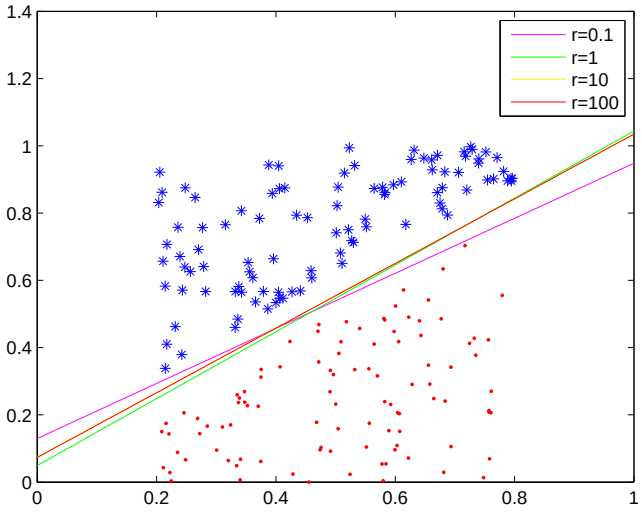


Figure 2: problem 6-3

Problem 7

真实数据往往是线性不可分的，此时经典的感知器算法无法收敛。针对这种情况，Gallant 在文献 [1] 中提出了口袋感知器算法（Pocket Perceptron）。其原理是依然采用经典感知器来训练，但在训练过程中维护一个最优权值的口袋，在迭代预设次数后算法结束，输出口袋中记录的最优权值。

- 参照文献 [1] 第 181 页的算法流程图，编程实现口袋感知器算法。
- 在 MNIST 数据集中针对 “0” 和 “1” 两类数字运行口袋感知器算法，给出结果和相应的分析。（鼓励探究不同的图像特征，这里给出一个简单的特征示例：将图片在 x 方向投影得到 28 维统计量，在 y 方向投影得到 28 维统计量，两个统计量合并成一个 56 维特征向量）
- 观察算法运行过程中错误率的变化情况，画出“错误率—迭代次数”曲线，并分析。

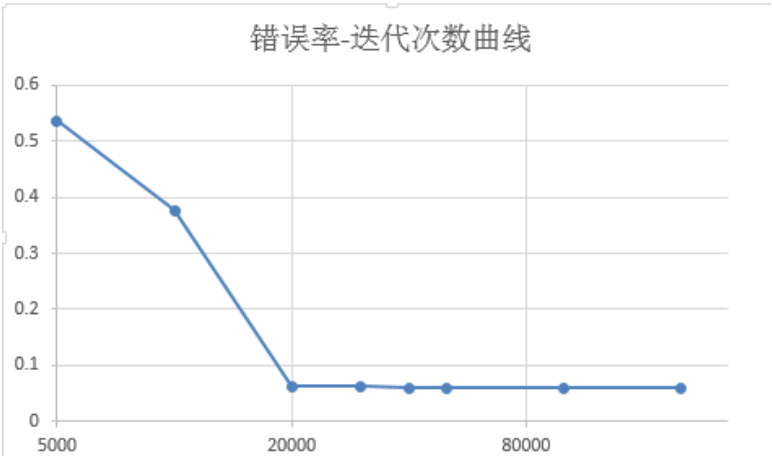


Figure 3: problem 7

在起始迭代中，迭代次数不断增加，可以使得错误率不断减小，但是当达到某一固定值后，错误率恒定，增加迭代次数没有变化。从算法的流程我们可以发现，每次迭代都会去找更新错误率最小值，当错误率全局最小后，增加迭代次数确实是徒劳的。

References

[1] Gallant S I. Perceptron-based learning algorithms[J]. Neural Networks, IEEE Transactions on, 1990, 1(2): 179-191.