CS 229, Autumn 2016
Problem Set #2 Solutions: Naive Bayes, SVMs, and Theory

Qingxin6174

1. Denote that $K_1(x,z) = \langle \phi_1(x), \phi_1(z) \rangle$, $K_2(x,z) = \langle \phi_2(x), \phi_2(z) \rangle$, $K_3(x,z) = \langle \phi_3(x), \phi_3(z) \rangle$.
   (a) $K(x,z) = K_1(x,z) + K_2(x,z) = \left\langle \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \end{bmatrix}, \begin{bmatrix} \phi_1(z) \\ \phi_2(z) \end{bmatrix} \right\rangle$ is a kernel;
   (b) let $0 \leqslant K_1(x,z) < K_2(x,z)$, $K(x,z) = K_1(x,z) - K_2(x,z) < 0$ isn't a kernel;
   (c) $K(x,z) = aK_1(x,z) = \langle \sqrt{a}\,\phi_1(x), \sqrt{a}\,\phi_1(z) \rangle$ is a kernel;
   (d) $K(x,z) = -aK_1(x,z) \leqslant 0$ isn't a kernel;
   (e) $K(x,z) = K_1(x,z)\,K_2(x,z) = \langle \varphi(x), \varphi(z) \rangle$ is a kernel, where φ is straightening transform of matrix $\phi_1\phi_2^T$;
   (f) $K(x,z) = f(x)f(z) = \langle f(x), f(z) \rangle$ is a kernel;
   (g) $K(x,z) = K_3(\phi(x), \phi(z)) = \langle \phi_3 \circ \phi(x), \phi_3 \circ \phi(z) \rangle$ is a kernel;
   (h) $K(x,z) = p(K_1(x,z)) = \sum_{i=0}^{n} a_i K_1^i(x,z)$, $(a_i > 0)$ is a kernel.

2. (a) Let $K\left(\varphi^{(i)}, x\right) = {\theta^{(i)}}^T \phi(x)$, $K\left(\varphi^{(0)}, x\right) = {\theta^{(0)}}^T \phi(x) = \vec{0}$;
   (b) $h_{\theta^{(i)}}\left(x^{(i+1)}\right) = g\left({\theta^{(i)}}^T \phi\left(x^{(i+1)}\right)\right) = g \circ K\left(\varphi^{(i)}, x^{(i+1)}\right)$;
   (c) $K\left(\varphi^{(i+1)}, x^{(i+1)}\right) := K\left(\varphi^{(i)}, x^{(i+1)}\right) + \alpha \mathbf{1}\{K\left(\varphi^{(i)}, x^{(i+1)}\right) y^{(i+1)} < 0\} y^{(i+1)} K\left(x^{(i+1)}, x^{(i+1)}\right)$.

3. (a) Here is the MATLAB code filling in `nb_train.m`

```
phi_1 = trainCategory * trainMatrix + 1;
phi_0 = (1 - trainCategory) * trainMatrix + 1;
phi_1 = phi_1' / sum(phi_1);  phi_0 = phi_0' / sum(phi_0);
phi_y = sum(trainCategory) / numTrainDocs;
```

   and here is the MATLAB code filling in `nb_test.m`

```
output=((1./(1+(1-phi_y)/phi_y*exp(testMatrix*log(phi_0./phi_1))))>=1/2);
```

   Test error: 0.0163.
   (b) `[~,idx]=sort(phi_1./phi_0,1,'descend');`
   the 5 tokens are *httpaddr*, *spam*, *unsubscrib*, *ebai* and *valet*.
   (c) Figure 1 shows naive Bayes classifier's test set error with different training set size. Training set size 1400 gives the best test set error.

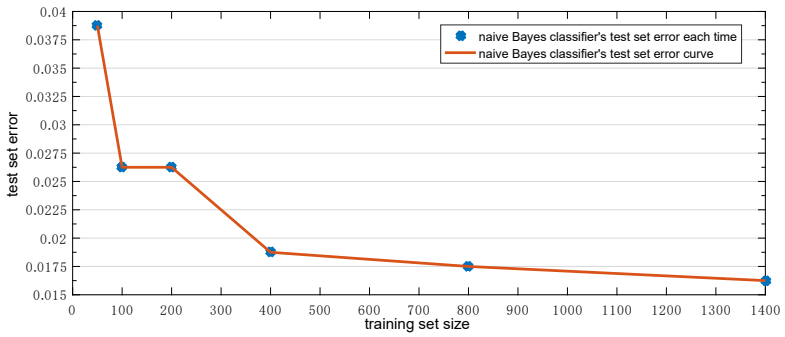

Fig. 1: Naive Bayes classifier's test set error with different training set size.

(d) Here is the MATLAB code filling in `svm_train.m`

```
steps = 40*numTrainDocs;
lambda = 1/(64*numTrainDocs);
tau = 8;
K = exp(-pdist2(Xtrain,Xtrain).^2/(2*tau^2));
alpha = zeros(numTrainDocs,steps+1);
for t = 1:steps
    idx = randperm(numTrainDocs,1);
    alpha(:,t+1) = alpha(:,t) - 1/sqrt(t)*(lambda*K*alpha(:,t) - ...
        (ytrain(idx)*K(idx,:)*alpha(:,t)<1) * ytrain(idx)*K(:,idx));
end
average_alpha = sum(alpha,2)/steps;
```

and here is the MATLAB code filling in `svm_test.m`

```
predictions = exp(-pdist2(Xtest,Xtrain).^2/(2*tau^2))*average_alpha>0;
predictions = 2 * predictions - 1;
```

Figure 2 shows the SVM's test set error with different training set size. (SGD makes the test set error different in the same training set size.)

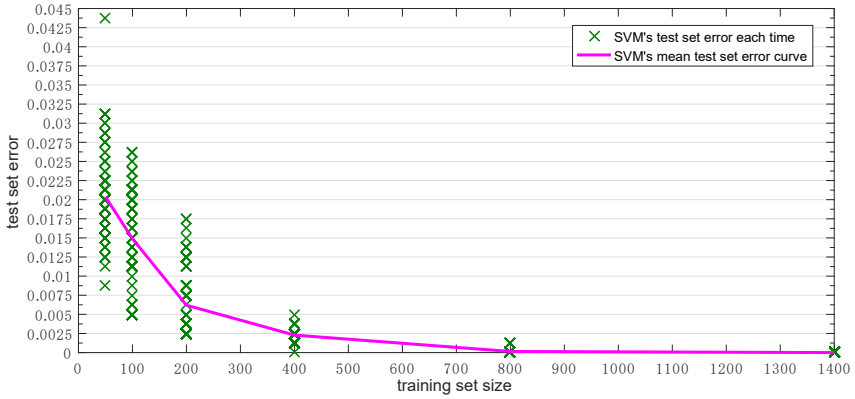

Fig. 2: SVM's test set error with different training set size.

(e) SVM performs better than naive Bayes as the size of training set grows. (Figure 3.)

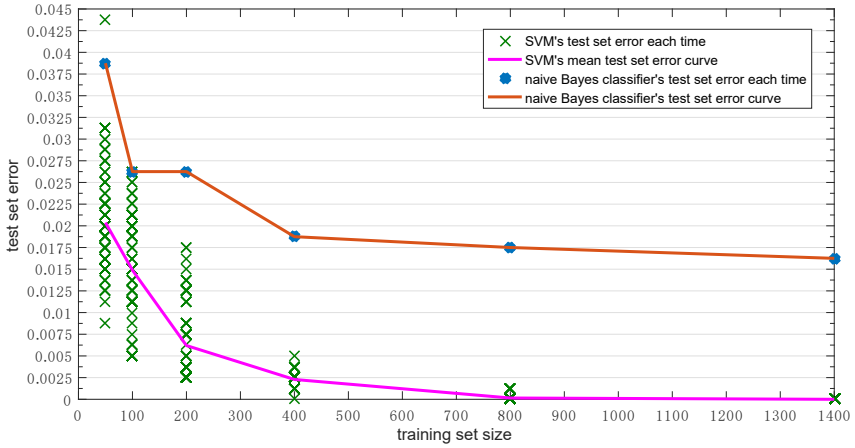

Fig. 3: SVM vs. naive Bayes.

4. (a) *Proof.* $\exists H$ s.t. $H = H_2 \backslash H_1$, $\mathrm{VC}(H_1) \leqslant \max\{\mathrm{VC}(H), \mathrm{VC}(H_1)\} = \mathrm{VC}(H2)$. □

(b) *Proof.* $\mathrm{VC}(\{h_1, \cdots, h_k\}) \leqslant k \Rightarrow \mathrm{VC}(H_1) \leqslant \mathrm{VC}(H_2) + \mathrm{VC}(\{h_1, \cdots, h_k\}) \leqslant \mathrm{VC}(H_2) + k$. □

(c) *Proof.* $\mathrm{VC}(H_1) = \max\{\mathrm{VC}(H_2), \mathrm{VC}(H_3)\} \leqslant \mathrm{VC}(H_2) + \mathrm{VC}(H_3)$. □

5. (a) $\varepsilon_\tau(h) = \varepsilon_0(h)(1-\tau) + (1-\varepsilon_0(h))\tau \Rightarrow \varepsilon_0(h) = \dfrac{\varepsilon_\tau(h) - \tau}{1-2\tau}$.

   (b) $P(\forall h \in H, \mid \varepsilon_\tau(h) - \hat{\varepsilon}_\tau(h) \mid \leqslant \mu) = P(\forall h \in H, \mid \varepsilon_0(h) - \hat{\varepsilon}_0(h) \mid \leqslant \dfrac{\mu}{1-2\tau}) \geqslant 1 - 2\,|H|\, e^{-2\mu^2 m}$, let $\gamma = \dfrac{\mu}{1-2\tau}$, $\delta = 2\,|H|\, e^{-2\mu^2 m}$, for $\varepsilon_0(\hat{h}) \leqslant \varepsilon_0(h^*) + 2\gamma$ to hold with probability $1-\delta$, it suffices that $m \geqslant \dfrac{1}{2(1-2\tau)^2\gamma^2} \log \dfrac{2\,|H|}{\delta}$.

   (c) $\tau = 0.5 \Rightarrow \varepsilon_\tau(h) \equiv 0.5$, which means for any hypothesis h, the generalization error won't change.

6. (a) For each threshold s, assume that $x^{(1)} > x^{(2)} > \cdots > x^{(\lambda)} \geqslant s > x^{(\lambda+1)} > \cdots > x^{(m)}$,

$$\begin{aligned}
\sum_{i=1}^{m} p_i \mathbf{1}\left\{\phi_{s,+}\left(x^{(i)}\right) \neq y^{(i)}\right\} &= \sum_{i=1}^{\lambda} p_i \mathbf{1}\left\{y^{(i)} = -1\right\} + \sum_{i=\lambda+1}^{m} p_i \mathbf{1}\left\{y^{(i)} = 1\right\} \\
&= \sum_{i=1}^{\lambda} \frac{1}{2}\left(1 - y^{(i)}\right) p_i + \sum_{i=\lambda+1}^{m} \frac{1}{2}\left(1 + y^{(i)}\right) p_i \\
&= \frac{1}{2} - \frac{1}{2}\left(\sum_{i=1}^{\lambda} y^{(i)} p_i - \sum_{i=\lambda+1}^{m} y^{(i)} p_i\right),
\end{aligned}$$

   let $m_0(s) = \lambda \in \{0, 1, \cdots, m\}$.

$$\begin{aligned}
\sum_{i=1}^{m} p_i \mathbf{1}\left\{\phi_{s,-}\left(x^{(i)}\right) \neq y^{(i)}\right\} &= 1 - \sum_{i=1}^{m} p_i \mathbf{1}\left\{\phi_{s,+}\left(x^{(i)}\right) \neq y^{(i)}\right\} \\
&= \frac{1}{2} - \frac{1}{2}\left(\sum_{i=m_0(s)+1}^{m} y^{(i)} p_i - \sum_{i=1}^{m_0(s)} y^{(i)} p_i\right).
\end{aligned}$$

   (b) $|f(m_0)| + |f(m_0+1)| \geqslant |f(m_0) - f(m_0+1)| = |-2y^{(m_0+1)} p_{m_0+1}| = 2p_{m_0+1}$,
   $p_i \geqslant 0$, $\sum_{i=1}^{m} p_i = 1 \Rightarrow \exists m_0$, s.t. $p_{m_0+1} \geqslant \dfrac{1}{m}$,
   $|f(m_0)| + |f(m_0+1)| \geqslant \dfrac{2}{m} \Rightarrow \exists m_0$, s.t. $|f(m_0)| \geqslant \dfrac{1}{m}$, so $|f(m_0)| \geqslant 2\gamma$, where $\gamma = \dfrac{1}{2m}$.

   (c) $\phi(x) \triangleq \begin{cases} \phi_{m_0,+}(x) & f(m_0) > 0 \\ \phi_{m_0,-}(x) & f(m_0) < 0 \end{cases} \Rightarrow \sum_{i=1}^{m} p_i \mathbf{1}\left\{\phi\left(x^{(i)}\right) \neq y^{(i)}\right\} \leqslant \dfrac{1}{2} - \dfrac{1}{2m}$, the edge γ is $\dfrac{1}{2m}$.
   The number of thresholds $t = \left\lceil \dfrac{2\log m}{-\log\left(1-4\gamma^2\right)} \right\rceil = \left\lceil \dfrac{2\log m}{2\log m - \log\left(m^2-1\right)} \right\rceil$ gives an upper bound to achieve zero error on a given training set.

   (d) i. My implementation of `find_best_threshold.m`

```
function [ind,thresh,err,id] = find_best_threshold(Xs,idx,y,p_dist,G)
[mm,nn] = size(Xs); err = zeros(1,nn); id = zeros(1,nn);
for i = 1:nn
    [err(i),id(i)] = min(1 - ((y(idx(:,i)).*p_dist(idx(:,i)))' * G));
end
err = err/2; [~,ind] = min(min([err;1-err]));
% make thresh to avoid "="
if      id(ind) == 1,       thresh = Xs(1,ind) + 1;
elseif id(ind) == mm + 1, thresh = Xs(mm,ind) - 1;
else    thresh = (Xs(id(ind)-1,ind) + Xs(id(ind),ind)) / 2;
end
err = err(ind); id = id(ind);
```

ii. My implementation of `stump_booster.m`

```
function [theta, feature_inds, thresholds] = stump_booster(X, y, T)
mm = size(X,1); p_dist = ones(mm,1); p_dist = p_dist / sum(p_dist);
[Xs,idx] = sort(X,'descend'); G = 2*triu(ones(mm,mm+1),1) - 1;
theta = zeros(T,1); feature_inds = zeros(T,1); thresholds = zeros(T,1);
for iter = 1:T
    [feature_inds(iter), thresholds(iter), err, id] = ...
            find_best_threshold(Xs, idx, y, p_dist, G);
    theta(iter) = 1/2*log((1-err)/err);
    p_dist(idx(:,feature_inds(iter))) = ...
            p_dist(idx(:,feature_inds(iter))) .* ...
            exp(-1*theta(iter)*y(idx(:,feature_inds(iter))).*G(:,id));
    p_dist = p_dist / sum(p_dist);
end
```

iii. Here is the MATLAB code filling in `random_booster.m`

```
G = sign(X(:,ind)-thresh);
err = p_dist' * (G ~= y); new_theta = 1/2*log((1-err)/err);
p_dist = p_dist.*exp(-1*new_theta*y.*G); p_dist = p_dist / sum(p_dist);
```

iv. Boosted decision stumps error converges fast, but resource consumption is very high. Random boosting runs very fast.

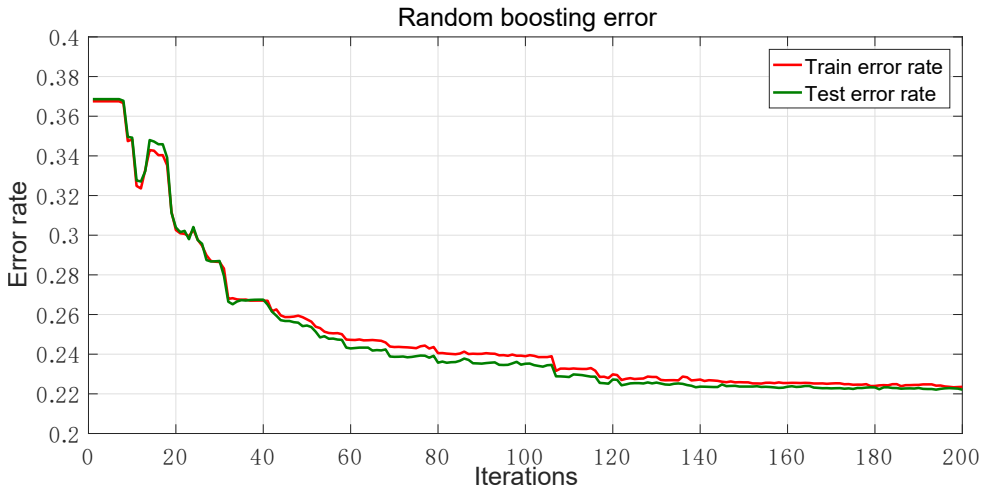

Fig. 4: Random boosting error.

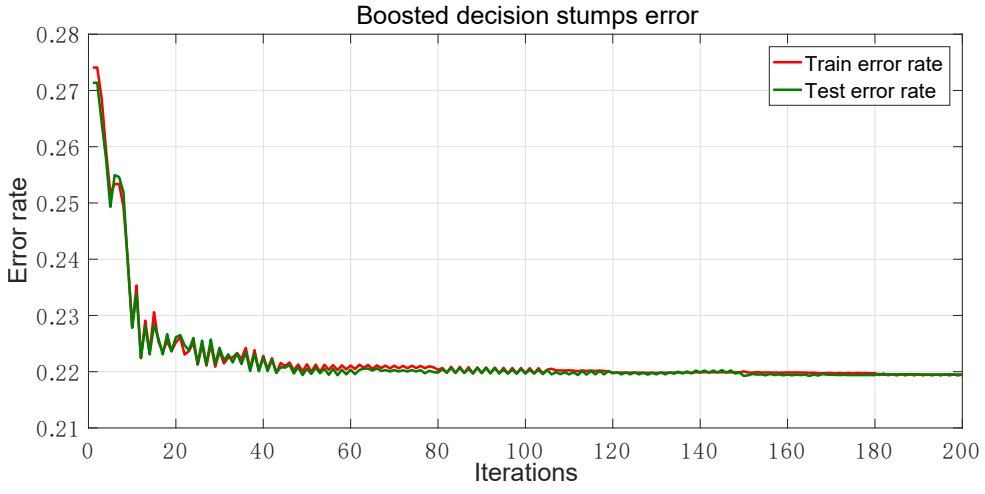

Fig. 5: Boosted decision stumps error.