TTIC 31020: Introduction to Machine Learning

Problem Set 4

Hung Le Tran

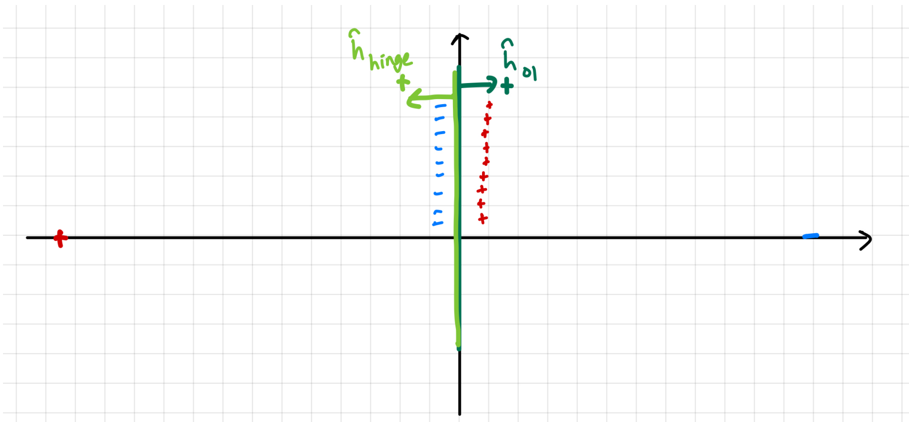
30 Jan 2024

Problem 4.1 (Problem 1)

(a) Choose

$$S = \{((1,k),+1),((-1,k),-1) : k \in [9]\} \cup \{((1000,-1),-1),((-1000,-1),+1)\}$$

See picture below.



Then a predictor that minimizes L_S^{01} is $h_{w=(1,0)}(x) = \langle (1,0), x\rangle = x[1]$ (1-index), which has $L^{01}(h) = \frac{2}{20} \le 0.1$. It thus follows that $\inf_{h\in\mathcal{H}} L_S^{01}(h) \le 0.1$. Meanwhile, $L^{hinge}(h) = \frac{1}{20} \times 2 \times [1-(-1000)]_+ = \frac{2002}{20} = 100.1$.

Then, the predictor $\hat{h}_{hinge}$ is the one that corresponds to $w_{hinge} = (-1,0)$, with $L^{01}(\hat{h}_{hinge}) = \frac{18}{20} = 0.9$, and $L^{hinge}(\hat{h}_{hinge}) = \frac{1}{20} \times 18 \times [1-(-1)]_+ = 1.8$

(b) No. $\inf_{h\in\mathcal{H}} L_S^{01}(h) = 0$, for finite S, means that there exists $\hat{w}$ such that $L_S^{01}(h_{\hat{w}}) = 0$. Then there exists a hinge loss minimizer $\hat{h}_{hinge}$, which achieves a hinge loss of 0, namely the one that corresponds to $w = M\hat{w}$ where $M \in \mathbb{R}_+$ is sufficiently large, so that the hinge loss is assured to go to 0. This $\hat{h}_{hinge}$ has the same decision boundary as $h_{\hat{w}}$ and so has 0 zero-one loss.

Problem 4.2 (Problem 2)

(a) Want to show that $\forall\, t, w_t \in S \coloneqq \operatorname{span}\{\phi(x_1), \dots, \phi(x_m)\}$.

Base case: $t = 0$. $w_0 = 0 \in S$ trivially.

Induction step: Assume that $w_t \in S$ for $t = k$. WTS $w_{k+1} \in S$. Indeed, w_{k+1} only exists if there still exists a misclassification, say, for $\phi(x_j)$ and y_j. Then the update rule is:

$$w_{t+1} = w_t + y_j \phi(x_j)$$

$w_t, y_j\phi(x_j) \in S \Rightarrow w_{t+1} \in S$.

By mathematical induction, we therefore have that $w_t \in S$ for all t, i.e.,

$$w_t = \sum_{i=1}^{m} \alpha_t[i]\phi(x_i) = \Phi^T \alpha_t$$

(b) We reiterate the original Perceptron algorithm:

$w_0 \leftarrow 0$
while $\exists\, i \in [m]$ such that $\operatorname{sign}(\langle w_t, \phi(x_i)\rangle) \neq y_i$ **do** $w_{t+1} \leftarrow w_t + y_i\phi(x_i)$
end while

We have that $\alpha_0 = 0 \in \mathbb{R}^m$. Then, we have

$$w_t\phi(x_i) = \sum_{j=1}^{m} \alpha_t[j]\phi(x_j)\phi(x_i) = \sum_{j=1}^{m} \alpha_t[j]K(x_i, x_j)$$

and

$$w_{t+1} = w_t + y_i\phi(x_i) = \sum_{j=1}^{m} \alpha_t[j]\phi(x_j) + y_i\phi(x_i) = \sum_{j=1}^{m} (\alpha_t[j] + \delta_{ij}y_j)\phi(x_j)$$

so we have that $\alpha_{t+1}[j] = \alpha_t[j] + \delta_{ij}y_j$ where δ_{ij} is the Kronecker delta.

Therefore, we can rewrite the Perceptron algorithm in terms of α_t and only with accesses to K:

$\alpha_0 \leftarrow 0 \in \mathbb{R}^m$
while $\exists\, i \in [m]$ such that $\operatorname{sign}\left(\sum_{j=1}^{m} \alpha_t[j]K(x_i, x_j)\right) \neq y_i$ **do** $\alpha_{t+1}[i] \leftarrow \alpha_t[j] + y_i$
end while

(c) Each iteration starts with checking if there remains some $i \in [m]$ such that

$$\operatorname{sign}\left(\sum_{j=1}^{m} \alpha_t[j]K(x_i, x_j) \neq y_i\right)$$

which takes $O(m^2 \cdot TIME_K) = O(m^2)$. Each update to α_t then takes $O(1)$, so in total each iteration takes $O(m^2)$ which is independent from d.

(d) From the Perceptron analysis, we know that

$$T_{max} = \frac{\|w*\|_2^2 \sup_{i \in [m]} \|\phi(x_i)\|_2^2}{\gamma^2}$$

but $\|\phi(x_i)\|_2^2 = K(x_i, x_i)$ so

$$T_{max} = \frac{\|w*\|_2^2 \max_{i \in [m]} K(x_i, x_i)}{\gamma^2}$$

It follows that the overall runtime bound is $O\left(m^2 \cdot TIME_K \cdot T_{max}\right)$.

Overall memory requirement of Kernelized Perceptron is $O(m^2)$ (to store the Gram matrix $O(m^2)$ and the current weight $O(m)$, assuming that the kernel computation does not take up memory).

(e) It must store the last weight w_T and all training samples $\{x_i\}$. The prediction of the new point x may be computed as:

$$\begin{aligned}
\langle w_T, \phi(x)\rangle &= \langle \sum_{j=1}^{m} \alpha_T[j]\phi(x_j), \phi(x)\rangle \\
&= \sum_{j=1}^{m} \alpha_T[j] K(x, x_j) \\
\text{sign}\left(\langle w_T, \phi(x)\rangle\right) &= \text{sign}\left(\sum_{j=1}^{m} \alpha_T[j] K(x, x_j)\right)
\end{aligned}$$

Memory requirement: $O(md')$ where d' is the dimension of the original feature space, i.e., $len(x_1)$, to be able to compute $K(x, x_j)$. Prediction runtime: $O(m \cdot TIME_K)$

Problem 4.3 (Problem 3)

(a) Remark that G is symmetric, so $G^T = G$. We have:

$$\begin{aligned}
L_{S,\lambda}(\alpha) &= L_{S,\lambda}(w(\alpha)) \\
&= \frac{1}{m}\|\Phi w - y\|^2 + \frac{\lambda}{2}\|w\|^2 \\
&= \frac{1}{m}\|\Phi\Phi^T\alpha - y\|^2 + \frac{\lambda}{2}\|\Phi^T\alpha\|^2 \\
&= \frac{1}{m}\|G\alpha - y\|^2 + \frac{\lambda}{2}(\Phi^T\alpha)^T(\Phi^T\alpha) \\
&= \frac{1}{m}(G\alpha - y)^T(G\alpha - y) + \frac{\lambda}{2}\alpha^T\Phi\Phi^T\alpha \\
&= \frac{1}{m}\left(\alpha^T G^T G\alpha - 2\alpha^T G^T y + y^T y\right) + \frac{\lambda}{2}\alpha^T G\alpha \\
&= \frac{1}{m}\left(\alpha^T G^2\alpha - 2\alpha^T G y + y^T y\right) + \frac{\lambda}{2}\alpha^T G\alpha
\end{aligned}$$

(b) $\hat{\alpha}_\lambda = \arg\min L_{S,\lambda}(\alpha)$ has $\nabla_\alpha L = 0$: We do the calculation:

$$\nabla_\alpha L = \frac{1}{m}(2G^2\alpha - 2Gy) + \lambda G\alpha$$

then for this to be zero, we have:

$$\begin{aligned}
2G^2\alpha - 2Gy + m\lambda G\alpha &= 0 \\
\Rightarrow (G^2 + \frac{m\lambda}{2}G)\alpha &= Gy \\
\Rightarrow \hat{\alpha}_\lambda &= (G + \frac{m\lambda}{2}I)^{-1}y
\end{aligned}$$

(c) For any test point x, we have the prediction

$$
\begin{aligned}
\langle \hat{w}_\lambda, \phi(x) \rangle &= \sum_{i=1}^{m} \hat{\alpha}_\lambda[i] \langle \phi(x_i), \phi(x) \rangle \\
&= \sum_{i=1}^{m} \hat{\alpha}_\lambda[i] K(x_i, x)
\end{aligned}
$$