EECS 545: Homework #3

Mingliang Duanmu
duanmuml@umich.edu

February 22, 2022

1 MAP estimates and weight decay

Take the log of both weights, we have

$$\mathbf{w}_{ML} = \operatorname*{argmax}_{\mathbf{w}} \sum_{i=1}^{N} \log(p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}))$$

$$\mathbf{w}_{MAP} = \operatorname*{argmax}_{\mathbf{w}} \sum_{i=1}^{N} \log(p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w})) + \log(p(\mathbf{w}))$$

Since $\mathbf{w} \sim \mathcal{N}(0, \tau^2 I)$,

$$\begin{aligned}\log(p(\mathbf{w})) &= \log(\mathcal{N}(0, \tau^2 I)) \\ &= \log(\frac{1}{\sqrt{2\pi\tau}}) + \log(\exp(-\frac{1}{2}(\mathbf{w}^T(\tau^{-1}I)^{-1}\mathbf{w})^2)) \\ &= \log(\frac{1}{\sqrt{2\pi\tau}}) + \log(\exp(-\frac{\tau}{2}\mathbf{w}^T\mathbf{w})) \\ &= \log(\frac{1}{\sqrt{2\pi\tau}}) - \frac{\tau}{2}||\mathbf{w}||^2\end{aligned}$$

So we can rewrite $\mathbf{w}_{MAP}$ as

$$\mathbf{w}_{MAP} = \operatorname*{argmax}_{\mathbf{w}} \sum_{i=1}^{N} \log(p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w})) - \frac{\tau}{2}||\mathbf{w}||^2$$

Due to the regularized term, the distance of $\mathbf{w}_{MAP}$ to zero need to be closer than $\mathbf{w}_{ML}$ to maximize the formula. Therefore, we conclude

$$||\mathbf{w}_{MAP}||_2 \leq ||\mathbf{w}_{ML}||_2$$

2 Direct construction of valid kernels

We assume

$$K_{ij}^{(n)} = k_n(x_i, z_j) \text{ and } K_{ij} = k(x_i, z_j)$$

where k_n is a kernel so $\mathbf{K}^{(n)}$ is symmetric and positive semi-definite, in this case we judge if k is a kernel.

a

$$z^T\mathbf{K}z = z^T(\mathbf{K}^{(1)} + \mathbf{K}^{(2)})z = z^T(\mathbf{K}^{(1)} + \mathbf{K}^{(2)})z = z^T\mathbf{K}^{(1)}z + z^T\mathbf{K}^{(2)}z \geq 0$$

Therefore, $k(\mathbf{x}, \mathbf{z})$ is a kernel.

b

$k(\mathbf{x}, \mathbf{z})$ is not a kernel. Suppose $\mathbf{K}^{(1)} = 0$ and $\mathbf{K}^{(2)} = 1$,

$$z^T\mathbf{K}z = z^T(\mathbf{K}^{(1)} - \mathbf{K}^{(2)})z = -z^Tz \leq 0$$

c

$$z^T\mathbf{K}z = z^T(a\mathbf{K}^{(1)})z = az^T\mathbf{K}^{(1)}z \geq 0$$

Therefore, $k(\mathbf{x}, \mathbf{z})$ is a kernel.

d

$k(\mathbf{x}, \mathbf{z})$ is not a kernel. Suppose $\mathbf{K}^{(1)} = 1$,

$$z^T\mathbf{K}z = z^T(-a\mathbf{K}^{(1)})z = -az^Tz \leq 0$$

e

$$z^T\mathbf{K}z = \sum_{i,j} z_i K_{ij}^{(1)} K_{ij}^{(2)} z_j$$

Since $\mathbf{K}^{(1)}$ is positive semi-definite, we have

$$\mathbf{K}^{(1)} = X\Lambda X^T = \sum_{k=1}^{n}(\lambda_k X_k X_k^T)$$

where $\lambda_i \geq 0$. So

$$K_{i,j}^{(1)} = \sum_{k=1}^{n} \lambda X_{k,i} X_{k,j}$$

We have

$$\begin{aligned} z^T\mathbf{K}z &= \sum_{i,j} z_i K_{ij}^{(1)} K_{ij}^{(2)} z_j \\ &= \sum_{i,j} z_i \sum_{k=1}^{n} \lambda X_{k,i} X_{k,j} K_{ij}^{(2)} z_j \\ &= \sum_{k=1}^{n} (\lambda_k \sum_{i,j} z_i X_{k,i} K_{ij}^{(2)} X_{k,j} z_j) \end{aligned}$$

Since $\mathbf{K}^{(2)}$ is also positive semi-definite,

$$z^T\mathbf{K}z = \sum_{k=1}^{n} (\lambda_k \sum_{i,j} z_i X_{k,i} K_{ij}^{(2)} X_{k,j} z_j) \geq 0$$

Therefore, $k(\mathbf{x}, \mathbf{z})$ is a kernel.

f

$$\begin{aligned} z^T\mathbf{K}z &= \sum_i \sum_j z_i k(x_i, x_j) z_j \\ &= \sum_i \sum_j z_i f(x_i) f(x_j) z_j \\ &= \sum_i (f(x_i) z_i)^2 \geq 0 \end{aligned}$$

Therefore, $k(\mathbf{x}, \mathbf{z})$ is a kernel.

g

Since k_3 is a kernel over $\mathbb{R}^M \times \mathbb{R}^M$ and $\phi : \mathbb{R}^D \to \mathbb{R}^M$, we have

$$\mathbf{K}_{ij}^{(3)} = k_3(x_i, z_j)$$

Therefore, $k(\mathbf{x}, \mathbf{z})$ is a kernel.

h

$$z^T\mathbf{K}z = \sum_i \sum_j z_i p(K_{ij}^{(1)}) z_j$$

Since $p(K_{ij}^{(1)})$ is a polynomial with positive coefficients, according to the conclusions in **a**, **c**, **e**, $k(\mathbf{x}, \mathbf{z})$ is a kernel.

i

Since $D = 2$, we have

$$(\mathbf{x}^T z + 1)^2 = (x_1 z_1 + x_2 z_2 + 1)^2 = x_1^2 z_1^2 + x_2^2 z_2^2 + 1 + 2x_1 z_1 x_1 z_1 + 2x_1 z_1 + 2x_2 z_2$$

So we can construct a map of

$$\phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)^T$$

3 Kernelizing the Perceptron

a

i

If $y^{(n)}h \geq 0$, $\mathbf{w}_{t+1} = \mathbf{w}_t$, which is in the form of $\Phi^T \alpha_{t+1}$.
If $y^{(n)}h < 0$, $\mathbf{w}_{t+1} = \mathbf{w} + y^{(n)}\Phi(x^{(n)}) = \Phi^T \alpha_t + y^{(n)}\Phi^T e_n$ where $e_n = (0, 0, \cdots, 1, 0, 0, \cdots)$. Therefore we have $\mathbf{w}_{t+1} = \Phi^T(\alpha_t + y^{(n)}e_n)$

ii

We initialize $\mathbf{w}_0 = \Phi^T \alpha_t = 0$ where $\alpha_t = 0$.
When $t = 0$, if $y^{(n)}h \geq 0$, then $\mathbf{w}_1 = 0$; if $y^{(n)}h < 0$, $\mathbf{w}_1 = y^{(i)}\Phi^T e_1$, which is the in the form of $\Phi^T \alpha_t$.
We assume $\mathbf{w}_t$ is in the form of $\Phi^T \alpha_t$, from the conclusion of **i** we know that $\mathbf{w}_{t+1}$ is also of the form $\Phi^T \alpha_{t+1}$.
Therefore, we prove the conclusion by induction.

b

i

From the solution of **a**, we find:
For $y^{(n)}h \geq 0$, $\alpha_{t+1} = \alpha_t, \mathbf{w}_{t+1} = \mathbf{w}_t$.
For $y^{(n)}h < 0$, $\alpha_{t+1} = \alpha_t + y^{(n)}e_n, \mathbf{w}_{t+1} = \mathbf{w}_t + \Phi^T y^{(n)}e_n$.
The maximum number of elements that differ between α_t and α_{t+1} is 1.

ii

$$\begin{aligned} h(\phi(\boldsymbol{x}^{(n)}), \boldsymbol{w}_t) &= \boldsymbol{w}_t^T \phi(\boldsymbol{x}^{(n)}) = \alpha_t^T \Phi \phi(\boldsymbol{x}^{(n)}) \\ &= \sum_{i=1}^N \alpha_{t,i} \phi(\boldsymbol{x}^{(i)}) \phi(\boldsymbol{x}^{(n)}) \\ &= \sum_{i=1}^N \alpha_{t,i} k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(n)}) \end{aligned}$$

c

Algorithm 1: PERCEPTRON TRAINING ALGORITHM

```
$\alpha_0 \leftarrow 0$;
for t = 0 to T - 1 do
    Pick a random training example $(x^{(n)}, y^{(n)})$ from D (with replacement)
    $h \leftarrow \sum_{i=1}^{N} \alpha_{t,i} k(x^{(i)}, x^{(n)})$
    if $y^{(n)} h < 0$ then
        $\alpha_{t+1} \leftarrow \alpha_t + y^{(n)} e_n$
    End
End
return $\alpha_t$
```

To classify the data, we need to calculate $sign(\sum_{i=1}^{N} \alpha_{t,i} k(x^{(i)}, x^{(n)}))$ for the test data.

4 Implementing Soft Margin SVM by Optimizing Primal Objective

a

Since

$$t^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b) \geq 1 - \xi_i$$

where $\xi_i \geq 0, \forall i = 1, \dots, N$, we have

$$\xi_i \geq 1 - t^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b)$$

Rewrite the expression, we set the minimum of ξ_i as

$$\xi_i = \max(0, 1 - t^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b))$$

Now we get the latter equation from the former one.
For the reverse case, the process is the same, so we prove that both equations are indeed equivalent optimization problems.

b

Since

$$\frac{\partial}{\partial x} \max(0, f(x)) = \mathbf{I} \frac{\partial f(x)}{\partial x}$$

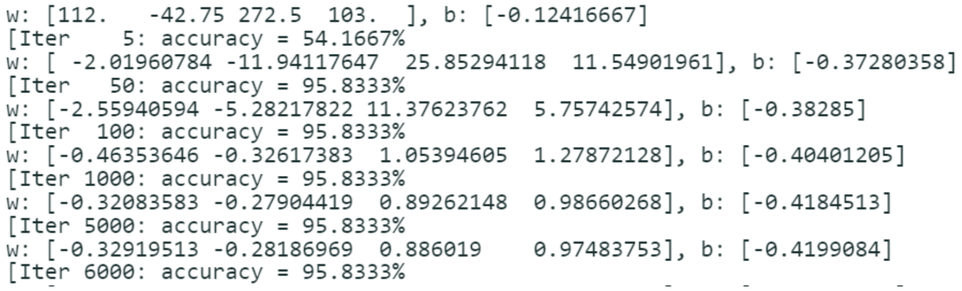
We have

$$\begin{aligned} \nabla_{\mathbf{w}} E(\mathbf{w}, b) =& \frac{\partial}{\partial \mathbf{w}} (\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \max(0, 1 - y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b))) \\ =& \mathbf{w} - C \sum_{i=1}^{N} \mathbf{I}[y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b) < 1] y^{(i)} \mathbf{x}^{(i)} \end{aligned}$$

Similarly we have

$$\begin{aligned} \frac{\partial}{\partial b} E(\mathbf{w}, b) =& \frac{\partial}{\partial b} (\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \max(0, 1 - y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b))) \\ =& - C \sum_{i=1}^{N} \mathbf{I}[y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b) < 1] y^{(i)} \end{aligned}$$

c



```
w: [112.   -42.75 272.5  103.  ], b: [-0.12416667]
[Iter    5: accuracy = 54.1667%
w: [ -2.01960784 -11.94117647  25.85294118  11.54901961], b: [-0.37280358]
[Iter   50: accuracy = 95.8333%
w: [-2.55940594 -5.28217822 11.37623762  5.75742574], b: [-0.38285]
[Iter  100: accuracy = 95.8333%
w: [-0.46353646 -0.32617383  1.05394605  1.27872128], b: [-0.40401205]
[Iter 1000: accuracy = 95.8333%
w: [-0.32083583 -0.27904419  0.89262148  0.98660268], b: [-0.4184513]
[Iter 5000: accuracy = 95.8333%
w: [-0.32919513 -0.28186969  0.886019    0.97483753], b: [-0.4199084]
[Iter 6000: accuracy = 95.8333%
```

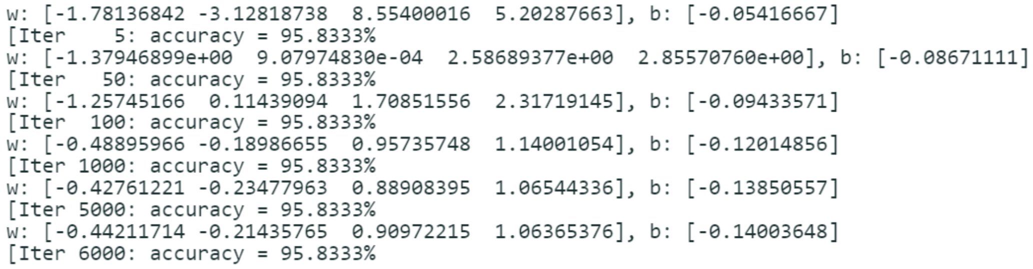
d

Similarly, we have

$$\nabla_{\mathbf{w}} E^{(i)}(\mathbf{w}, b) = \frac{\mathbf{w}}{N} - C\mathbf{I}[y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) < 1]y^{(i)}\mathbf{x}^{(i)}$$

$$\frac{\partial}{\partial b} E^{(i)}(\mathbf{w}, b) = -C\mathbf{I}[y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) < 1]y^{(i)}$$

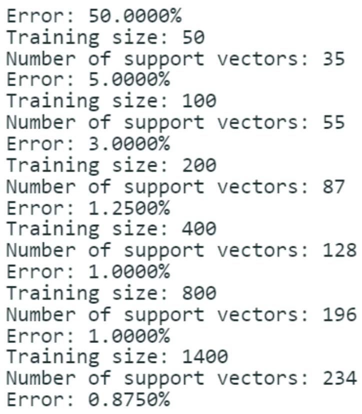
e



```
w: [-1.78136842 -3.12818738  8.55400016  5.20287663], b: [-0.05416667]
[Iter    5: accuracy = 95.8333%
w: [-1.37946899e+00  9.07974830e-04  2.58689377e+00  2.85570760e+00], b: [-0.08671111]
[Iter   50: accuracy = 95.8333%
w: [-1.25745166  0.11439094  1.70851556  2.31719145], b: [-0.09433571]
[Iter  100: accuracy = 95.8333%
w: [-0.48895966 -0.18986655  0.95735748  1.14001054], b: [-0.12014856]
[Iter 1000: accuracy = 95.8333%
w: [-0.42761221 -0.23477963  0.88908395  1.06544336], b: [-0.13850557]
[Iter 5000: accuracy = 95.8333%
w: [-0.44211714 -0.21435765  0.90972215  1.06365376], b: [-0.14003648]
[Iter 6000: accuracy = 95.8333%
```

5 SciKit-Learn SVM for classifying SPAM

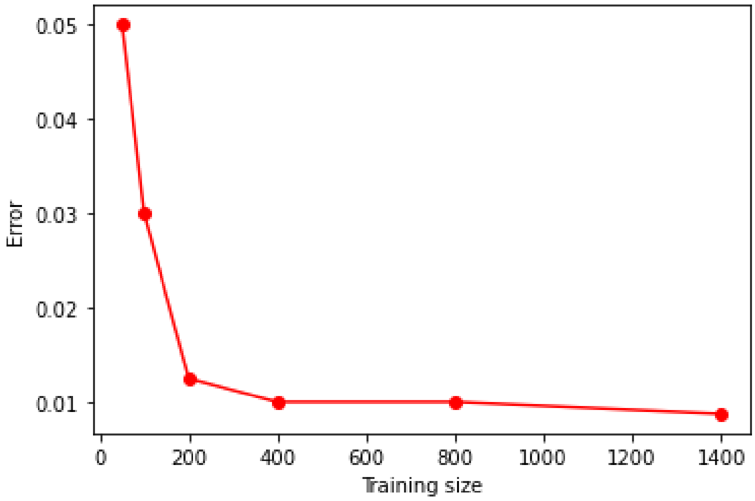
a



```
Error: 50.0000%
Training size: 50
Number of support vectors: 35
Error: 5.0000%
Training size: 100
Number of support vectors: 55
Error: 3.0000%
Training size: 200
Number of support vectors: 87
Error: 1.2500%
Training size: 400
Number of support vectors: 128
Error: 1.0000%
Training size: 800
Number of support vectors: 196
Error: 1.0000%
Training size: 1400
Number of support vectors: 234
Error: 0.8750%
```

b

Please check the code output of `q5.py` for error and number of support vectors.



The training set size of 1400 gives the best test set error.

c

Compared to naive Bayes, the error of SVM is a bit higher at first when training size is small. However, with the training size increasing, SVM converges faster than naive Bayes, and the final error is smaller for large training size.