CS229 Problem Set 2

Tianyu Du

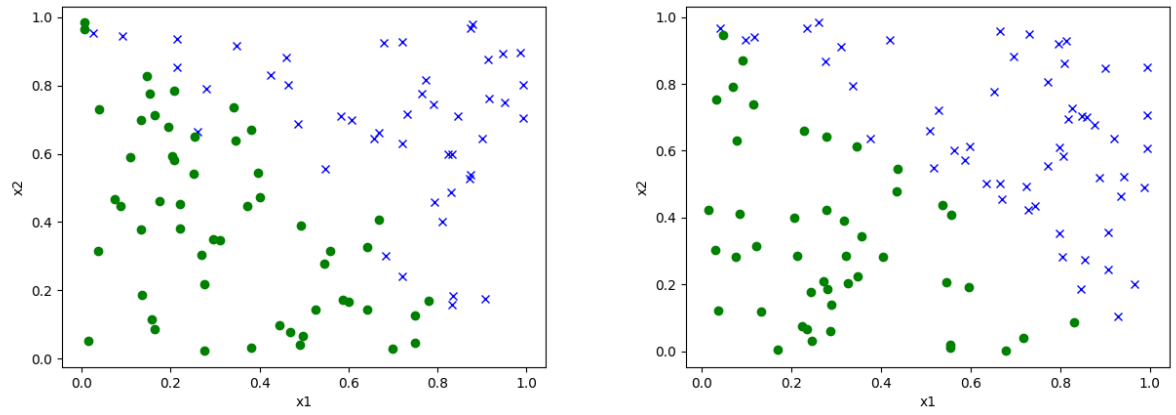
Friday 26$^{\text{th}}$ July, 2019

1 Problem 1 Logistic Regression: Training Stability

1.1 1(a)

Comment The parameter trained on dataset A converges after around 30,000 iterations of gradient ascent. In contrast, on dataset B, the training algorithm does not reach convergence within a reasonable amount of time.

1.2 1(b)



(a) Dataset A (b) Dataset B

Figure 1: Two Datasets

Comment From the plot of two datasets we can see that dataset B is more *linearly separable* than dataset A. There exists intermixing of data points belonging to both classes in dataset A, but fewer can be observed in dataset B. The linear separability of dataset B causes the instability of logistic regression on it. In the scenario of dataset B, we claim that the gradient descent algorithm increases θ indefinitely without convergence.

Proof. Let $\mathcal{A}$ and $\mathcal{B}$ denote two datasets respectively. And the log-likelihood looks like

$$\ell(\theta) = \sum_{i=1}^{n} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right) \tag{1.1}$$

$$= \underbrace{\sum_{i:y^{(i)}=0} \log\left(1 - h_\theta(x^{(i)})\right)}_{P} + \underbrace{\sum_{i:y^{(i)}=1} \log h_\theta(x^{(i)})}_{Q} \tag{1.2}$$

Suppose the threshold for classification is 0.5, and dataset $\mathcal{B}$ can be separated perfectly with some $\theta \in \mathbb{R}^d$. That's, for (almost) every sample $i \in \mathcal{B}$ such that $y^{(i)} = 0$, $\theta^T x^{(i)} < 0$ and for (almost) every $y^{(i)} = 1$, $\theta^T x^{(i)} > 0$. As a result, for every θ^t that separate dataset $\mathcal{B}$ almost perfectly, suppose θ^t is inflated to $\theta^{t+1} := (1+\varepsilon)\theta^t$ for some $\varepsilon > 0$, for those negative samples ($y^{(i)} = 0$), $\theta^T x^{(i)}$ changes to $(1+\varepsilon)\theta^T x^{(i)} < \theta^T x^{(i)} < 0$, and $h_\theta(x^{(i)})$ decreases. As a result, P increases. Similarly, for those positive samples, $\theta^T x^{(i)}$ changes to $(1+\varepsilon)\theta^T x^{(i)} > \theta^T x^{(i)} > 0$, and $h_\theta(x^{(i)})$, so does Q.

It is shown that for each θ^t separates dataset perfectly (or at least almost perfectly, because if the misclassified group is small, we can safely ignore the contribution to likelihood function from samples from the misclassified group), the log likelihood is increased when θ is inflated, and the

resulted θ^{t+1} is still a (almost) perfect-separating parameter. Also, there is no upper bound on $||\theta||$, therefore, the gradient ascend algorithm will run indefinitely to increase $\ell(\theta)$ by inflation the norm of θ.

For dataset like $\mathcal{A}$, for each θ, the misclassified group provides significant contributions to $\ell(\theta)$, inflating θ comes with the cost of reduced likelihood on samples from the misclassified groups. By the nature of log function, the cost of likelihood reduction on misclassified group will eventually overcome the the likelihood gain by expanding θ, and $\ell(\theta)$ cannot be risen to infinity on this kind of datasets. Therefore, the algorithm stops at some θ^*. ■

1.3 1(c)

Comment

(i) (NO) A different constant learning rate won't help because the $\nabla_\theta \ell(\theta)$ does not vanish, and the update step $\alpha \nabla_\theta \ell(\theta)$ is always significant. Therefore, convergence is still not achievable.

(ii) (NO) Annealing the learning rate will not mitigate the problem because this does not change the fact that the maximum of ℓ is achieved when $\theta \to \infty$.

(iii) (NO) Linear scaling does not change the fact that dataset is well (linearly) separable, θ is still going to explode indefinitely.

(iv) (YES) Adding regularization helps. As mentioned before, the main cause of interminability of gradient ascent is $\ell(\theta)$ can always be increased by inflating θ (i.e. increasing $||\theta||$). The regularizing term prevents θ from exploding and enforce the convergence.

(v) (YES) Give the variance of noise is large enough, the noise helps eliminate the linear separability of data, so that the dataset becomes intermixing.

1.4 1(d)

Comment SVM is more robust against linearly separable dataset. SVM is seeking to construct a hyperplane $w^T x + b$ to maximize the geometric margin γ, while controlling $||w|| = 1$. Note that changing b would shift the position of the hyperplane, therefore at optimal, $b^* < \infty$. Also, the performance measure of SVM using geometric margin is invariant to scales of (w, b). Therefore, given the constraint on $||w||$, SVM using geometric margin does not suffer from the parameter exploding problem (optimal is achieved when $\theta \to \infty$), and the optimal (w^*, b^*) will both be finite.

2 Problem 2 Spam Classification

2.1 2(a)

Result `Size of dictionary: 1722`

2.2 2(b)

Result `Naive Bayes had an accuracy of 0.978494623655914 on the testing set`

2.3 2(c)

Result `The top 5 indicative words for Naive Bayes are: ['claim', 'won', 'prize', 'tone', 'urgent!']`

2.4 2(d)

Result `The optimal SVM radius was 0.1`

3 Problem 3 Constructing Kernels

Notation 3.1. In the question, we use $K(\cdot)$ to denote the kernel mapping, and use K to denote the corresponding kernel matrix generated by some collection $(x^{(i)})_{i=1}^n \subset \mathbb{R}^d$. Specifically, $K_{i,j} := K(x^{(i)}, x^{(j)})$.

3.1 3(a)

Proof. $K(\cdot)$ is a kernel. Let $(x^{(i)})_{i=1}^n \subset \mathbb{R}^d$ be any sequence of n vectors from $\mathbb{R}^d$, let K_1 and K_2 denote the corresponding kernel metrics. By definition, the (j,k) term of the kernel matrix generated by K is

$$K_{j,k} = K(x^{(j)}, x^{(k)}) \tag{3.1}$$
$$= K_1(x^{(j)}, x^{(k)}) + K_2(x^{(j)}, x^{(k)}) \tag{3.2}$$
$$= K_{1,j,k} + K_{2,j,k} \tag{3.3}$$

Therefore, $K = K_1 + K_2$. Because $K_1(\cdot)$ and $K_2(\cdot)$ are valid kernels, K_1 and K_2 are symmetric, so is K. Let $z \in \mathbb{R}^n$, note that both K_1 and K_2 are PSD, thus,

$$z^T K z = z^T K_1 z + z^T K_2 z \geq 0 \tag{3.4}$$

Therefore, for any $(x^{(i)})_{i=1}^n \subset \mathbb{R}^d$, the corresponding kernel matrix generated by $K(\cdot)$ is symmetric and PSD, hence $K(\cdot)$ is a valid kernel. ■

3.2 3(b)

Proof. $K(\cdot)$ is not necessarily a kernel. One counter-example can be constructed as following:

$$K_1(x, y) := \mathbb{1}\{x = y\} \tag{3.5}$$

$$K_2(x, y) := 2 \times \mathbb{1}\{x = y\} \tag{3.6}$$

Then for any $(x^{(i)})_{i=1}^n \subset \mathbb{R}^d$ sequence, the corresponding kernel matrices are

$$K_1 = I_n \tag{3.7}$$

$$K_2 = 2I_n \tag{3.8}$$

Both K_1 and K_2 are symmetric and PSD, so $K_1(\cdot)$ and $K_2(\cdot)$ are valid kernels. Similarly to part (a), $K = K_1 - K_2$ for any $(x^{(i)})_{i=1}^n \subset \mathbb{R}^d$. In this case, for any $z \in \mathbb{R}^n$, $z^T K z \ngeq 0$, so K is not PSD. Hence, $K(\cdot)$ is not necessarily a valid kernel. ■

3.3 3(c)

Proof. $K(\cdot)$ is a kernel. For any $(x^{(i)})_{i=1}^n \subset \mathbb{R}^d$, let K_1 denote the kernel matrix generated by kernel $K_1(\cdot)$, which is PSD and symmetric. And the kernel matrix $K = aK_1$ is symmetric. And because for every $z \in \mathbb{R}^n$,

$$z^T K_1 z \geq 0 \implies z^T K_1 z \geq 0 \tag{3.9}$$

$$\implies z^T K z \geq 0 \tag{3.10}$$

so K is PSD and $K(\cdot)$ is a valid kernel. ■

3.4 3(d)

Proof. $K(\cdot)$ is not a kernel. Let K and K_1 denote the corresponding kernel matrices with some $(x^{(i)})_{i=1}^n \subset \mathbb{R}^d$. Then, K_1 is PSD. Therefore, for every $z \in \mathbb{R}^n$,

$$z^T K_1 z \geq 0 \implies (-a) z^T K_1 z \leq 0 \tag{3.11}$$

$$\implies z^T K z \leq 0 \tag{3.12}$$

Therefore K is not PSD, and $K(\cdot)$ is not a valid kernel. ■

3.5 3(e)

Proof. $K(\cdot)$ is a kernel. Given that $K_1(\cdot)$ and $K_2(\cdot)$ are valid kernels, there exists some feature mappings corresponding to these kernels, say $\phi_1 : \mathbb{R}^d \to \mathbb{R}^M$ and $\phi_2 : \mathbb{R}^d \to \mathbb{R}^N$, where M and N are potentially infinite. Consider a collection $(x^{(i)})_{i=1}^n \subset \mathbb{R}^d$ and generated kernel matrices K_1, K_2, and K. Let $\phi_{1,\ell}(x^{(i)})$ denote the ℓ^{th} element of $\phi_1(x^{(i)})$. By definition,

$$
\begin{aligned}
K_{i,j} &= K(x^{(i)}, x^{(j)}) && (3.13)\\
&= K_1(x^{(i)}, x^{(j)}) K_2(x^{(i)}, x^{(j)}) && (3.14)\\
&= \langle \phi_1(x^{(i)}), \phi_1(x^{(j)}) \rangle \langle \phi_2(x^{(i)}), \phi_2(x^{(j)}) \rangle && (3.15)\\
&= \left[\sum_{m=1}^{M} \phi_{1,m}(x^{(i)}) \phi_{1,m}(x^{(j)}) \right] \left[\sum_{n=1}^{N} \phi_{2,n}(x^{(i)}) \phi_{2,n}(x^{(j)}) \right] && (3.16)\\
&= \sum_{m=1}^{M} \left\{ \phi_{1,m}(x^{(i)}) \phi_{1,m}(x^{(j)}) \left[\sum_{n=1}^{N} \phi_{2,n}(x^{(i)}) \phi_{2,n}(x^{(j)}) \right] \right\} && (3.17)\\
&= \sum_{m=1}^{M} \sum_{n=1}^{N} \left\{ \phi_{1,m}(x^{(i)}) \phi_{1,m}(x^{(j)}) \phi_{2,n}(x^{(i)}) \phi_{2,n}(x^{(j)}) \right\} && (3.18)\\
&= \sum_{m \in [M], n \in [N]} \phi_{1,m}(x^{(i)}) \phi_{1,m}(x^{(j)}) \phi_{2,n}(x^{(i)}) \phi_{2,n}(x^{(j)}) && (3.19)
\end{aligned}
$$

Now we construct a new feature mapping $\phi_3 : \mathbb{R}^d \to \mathbb{R}^{M \times N}$. For each component of $\phi(x^{(i)})$, it can be located using a double index (m, n), where $m \in [M], n \in [N]$. Define ϕ_3 as

$$
\phi_{3,(m,n)}(x^{(i)}) := \phi_{1,m}(x^{(i)}) \phi_{2,n}(x^{(i)}) \tag{3.20}
$$

Let $\mathcal{I}$ denote the set of all (m, n) indices for components of $\phi_3(x^{(i)})$, note that $|\mathcal{I}| = M \times N$, then

$$
\begin{aligned}
K_{i,j} &= \sum_{(m,n) \in \mathcal{I}} \phi_{1,m}(x^{(i)}) \phi_{1,m}(x^{(j)}) \phi_{2,n}(x^{(i)}) \phi_{2,n}(x^{(j)}) && (3.21)\\
&= \sum_{(m,n) \in \mathcal{I}} \phi_{3,(m,n)}(x^{(i)}) \phi_{3,(m,n)}(x^{(j)}) && (3.22)\\
&= \langle \phi_3(x^{(i)}), \phi_3(x^{(j)}) \rangle && (3.23)
\end{aligned}
$$

And K has been shown to be the the kernel matrix generated by feature mapping ϕ_3, so $K(\cdot)$ is a valid kernel with corresponding feature mapping ϕ_3. ■

3.6 3(f)

Proof. $K(\cdot)$ is a kernel. Let $(x^{(i)})_{i=1}^n \subset \mathbb{R}^d$ be an arbitrary finite set,

$$K_{i,j} = f(x^{(i)})f(x^{(j)}) \tag{3.24}$$
$$= f(x^{(j)})f(x^{(i)}) = K_{j,i} \tag{3.25}$$

So the generated kernel matrix is symmetric. Let $z \in \mathbb{R}^n$, then

$$z^T K z = \sum_{i=1}^n \sum_{j=1}^n z_i z_j K_{i,j} \tag{3.26}$$
$$= \sum_{i=1}^n \sum_{j=1}^n z_i f(x^{(i)}) z_j f(x^{(j)}) \tag{3.27}$$
$$= \langle (f(x^{(i)}))_{i=1}^n, z \rangle^2 \geq 0 \tag{3.28}$$

Therefore, K is PSD. So $K(\cdot)$ is a valid kernel. ■

3.7 3(g)

Proof. $K(\cdot)$ is a valid kernel. Since K_3 is a kernel over $\mathbb{R}^p \times \mathbb{R}^p$, there exists a feature mapping $\lambda : \mathbb{R}^p \to \mathbb{R}^l$ for some l, such that $K_3(x, y) = \langle \lambda(x), \lambda(y) \rangle$ for every $x, y \in \mathbb{R}^p$. I am going to show that $K(\cdot)$ is a valid kernel on $\mathbb{R}^d \times \mathbb{R}^d$ by showing it can be written as an inner product of feature mapping $\lambda \circ \phi : \mathbb{R}^d \to \mathbb{R}^l$. Let $x, z \in \mathbb{R}^d$, then by definition

$$K(x, z) = K_3(\phi(x), \phi(z)) \tag{3.29}$$
$$= \langle \lambda(\phi(x)), \lambda(\phi(z)) \rangle \tag{3.30}$$
$$= \langle \lambda \circ \phi(x), \lambda \circ \phi(z) \rangle \tag{3.31}$$

Therefore $K(\cdot)$ is a valid kernel. ■

3.8 3(h)

Proof. $K(\cdot)$ is a valid kernel. By the result from (e) and induction, it can be shown that $K(x,z) := \prod_i K_i(x,z)$, where all $K_i(\cdot)$ are kernels, is a valid kernel. This result can be reduced by setting all $K_i(\cdot)$ equal: let $K(\cdot)$ be a kernel, $n \in \mathbb{N}$, then $K'(x,z) := K(x,z)^n$ is also a kernel. Let $(\alpha_i)_{i=1}^P$ denote the coefficients of $p(\cdot)$, and $\alpha_i > 0$ for every i. Then by result from (c) and (e), for every i, $K_i(x,z) := \alpha_i K_1(x,z)^i$ is a valid kernel. And by result from (a) and induction, $K(x,z) := \sum_i K_i(x,z)$ is a valid kernel, therefore since

$$K(x,z) := \sum_{i=1}^{P} \alpha_i K_1(x,z)^i = \sum_{i=1}^{P} K_i(x,z) \tag{3.32}$$

$K(\cdot)$ is also a valid kernel. ■

4 Problem 4 Kernelizing the Perceptron

4.1 4(a)

Answer The kernel trick firstly represents each $\theta^{(i)}$ as a linear combination of features of observations:

$$\theta^{(i)} = \sum_{\ell=1}^{n} \beta_\ell^{(i)} \phi(x^{(\ell)}) \tag{4.1}$$

$\theta^{(0)}$ can be represented simply by setting $\beta_\ell^{(0)} = 0$ for every ℓ. To make prediction, $\theta^{(i)} \cdot \phi(x^{(i+1)})$ is firstly computed as

$$\theta^{(i)} \cdot \phi(x^{(i+1)}) = \left[\sum_{\ell=1}^{n} \beta_\ell^{(i)} \phi(x^{(\ell)}) \right] \cdot \phi(x^{(i+1)}) \tag{4.2}$$

$$= \sum_{\ell=1}^{n} \left[\beta_\ell^{(i)} \phi(x^{(\ell)}) \cdot \phi(x^{(i+1)}) \right] \tag{4.3}$$

$$= \sum_{\ell=1}^{n} \beta_\ell^{(i)} K\left(\phi(x^{(\ell)}), \phi(x^{(i+1)}) \right) \tag{4.4}$$

$$\tag{4.5}$$

The prediction is given by

$$h_{\theta^{(i)}}(x^{(i+1)}) = \text{sign}\left(\sum_{\ell=1}^{n} \beta_\ell^{(i)} K\left(\phi(x^{(\ell)}), \phi(x^{(i+1)}) \right) \right) \tag{4.6}$$

According to the given update rule,

$$\theta^{(i+1)} = \sum_{\ell=1}^{n} \beta_\ell^{(i+1)} \phi(x^{(\ell)}) \tag{4.7}$$

$$:= \theta^{(i)} + \alpha \left(y^{(i+1)} - h_{\theta^{(i)}}(x^{(i+1)}) \right) \phi(x^{(i+1)}) \tag{4.8}$$

$$= \sum_{\ell=1}^{n} \beta_\ell^{(i)} \phi(x^{(\ell)}) + \alpha \left(y^{(i+1)} - h_{\theta^{(i)}}(x^{(i+1)}) \right) \phi(x^{(i+1)}) \tag{4.9}$$

Note that at this epoch, only β_{i+1} is modified,

$$\beta_{i+1}^{(i+1)} := \beta_{i+1}^{(i)} + \alpha \left(y^{(i+1)} - h_{\theta^{(i)}}(x^{(i+1)}) \right) \tag{4.10}$$

$$= \beta_{i+1}^{(i)} + \alpha \left(y^{(i+1)} - g(\theta^{(i)} \cdot \phi(x^{(i+1)})) \right) \tag{4.11}$$

For any $\ell \neq i+1$,

$$\beta_\ell^{(i+1)} := \beta_\ell^{(i)} \tag{4.12}$$

Given that $\beta_\ell^{(0)}$s are initialized to zero. And β_ℓ is updated at and only at step ℓ. Therefore, $\beta_{i+1}^{(i)} = \beta_{i+1}^{(0)} = 0$. The update rule can also be written as

$$\beta_{i+1}^{(i+1)} := \alpha \left(y^{(i+1)} - g(\theta^{(i)} \cdot \phi(x^{(i+1)})) \right) \tag{4.13}$$

Moreover, because $\beta_\ell^{(i)} = 0$ for every $\ell > i$, the linear combination representation of $\theta^{(i)}$

$$\theta^{(i)} = \sum_{\ell=1}^{n} \beta_\ell^{(i)} \phi(x^{(\ell)}) \tag{4.14}$$

$$= \sum_{\ell=1}^{i} \beta_\ell^{(i)} \phi(x^{(\ell)}) + \sum_{\ell=i+1}^{n} \beta_\ell^{(i)} \phi(x^{(\ell)}) \tag{4.15}$$

$$= \sum_{\ell=1}^{i} \beta_\ell^{(i)} \phi(x^{(\ell)}) + 0 \tag{4.16}$$

$$= \sum_{\ell=1}^{i} \beta_\ell^{(i)} \phi(x^{(\ell)}) \tag{4.17}$$

and the prediction can be reduced to

$$\theta^{(i)} \cdot \phi(x^{(i+1)}) = \text{sign} \left(\sum_{\ell=1}^{i} \beta_\ell^{(i)} K \left(\phi(x^{(\ell)}), \phi(x^{(i+1)}) \right) \right) \tag{4.18}$$

4.2 4(b) Coding

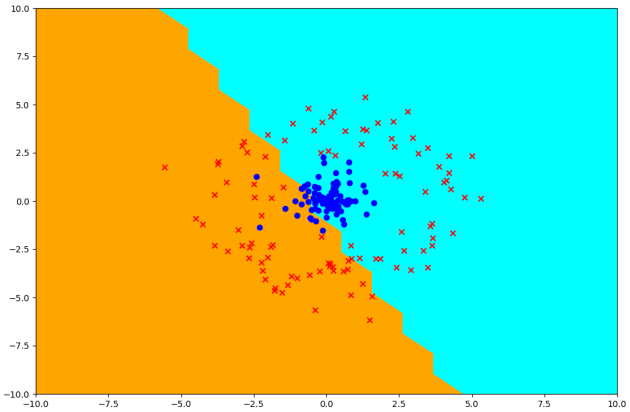


Figure 2: Classification with Dot Kernel

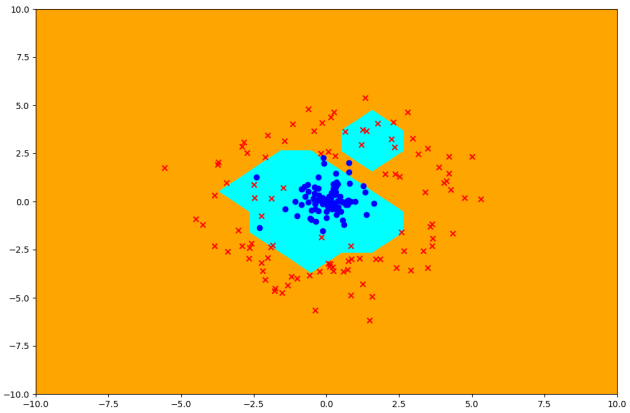


Figure 3: Classification with RBF Kernel

4.3 4(c)

Answer The dot kernel performed poorly. The dot kernel effectively exerted no feature engineering (i.e. $\phi(x) = x$). The prediction is then $\text{sign}(\theta \cdot x)$. The classifier is seeking for a best separating hyperplane (i.e. a linear boundary) in the space of raw x for classification. From the distribution of samples in the graph, this is obviously infeasible. And there might be a separating hyperplane in the feature space generated by feature mapping ϕ represented RBF kernel, so RBF kernel worked.