CSCI 567 Homework 1

Furong (Flora) Jia & Qilin Ye

September 13, 2022

Link to problem set: `https://vatsalsharan.github.io/fall22/hw1.pdf`.

1 Perceptron Convergence

Solution (1.1). As stated, let k be given and let x_i be the one selected during that iteration (i.e., $\operatorname{sgn}(w_k^T x_i) \neq y_i$). By assumption $y_i(w_{\text{opt}}^T x_i) \geqslant \gamma$. Since $w_{k+1} = w_k + y_i x_i$, linearity gives

$$w_{k+1}^T w_{\text{opt}} = w_k^T w_{\text{opt}} + y_i(x_i^T w_{\text{opt}}) \geqslant w_k^T w_{\text{opt}} + \gamma = w_k^T w_{\text{opt}} + \gamma \|w_{\text{opt}}\|.$$

Solution (1.2). Following 1.1 we assume $y_i(w_k^T x_i) < 0$. Using definition of w_{k+1} again,

$$\begin{aligned} \|w_{k+1}\|^2 &= \|w_k + y_i x_i\|^2 = \|w_k\|^2 + 2y_i(w_k^T x_i) + \|x_i\|^2 \\ &\leqslant \|w_k\|^2 + \|x_i\|^2 \leqslant \|w_k\|^2 + R^2. \end{aligned}$$

Solution (1.3). For the lower bound, inductively applying the inequality derived in 1.1 implies (assuming $\|w_{\text{opt}}\| = 1$)

$$w_{k+1}^T w_{\text{opt}} \geqslant \gamma M + w_0^T w_{\text{opt}} = \gamma M.$$

It remains to notice that by Cauchy-Schwarz

$$\gamma M \leqslant w_{k+1}^T w_{\text{opt}} \leqslant \|w_{k+1}\| \|w_{\text{opt}}\| = \|w_{k+1}\|.$$

For the upper bound, inductively applying the inequality derived in 1.2 implies

$$\|w_{k+1}\|^2 \leqslant \|w_0\|^2 + R^2 M = R^2 M \implies \|w_{k+1}\| \leqslant R\sqrt{M}. \tag{**}$$

Solution (1.4). Trivial. (*) in conjunction with (**) imply $\gamma M \leqslant R\sqrt{M}$, so $\sqrt{M} \leqslant R/\gamma$ and $M \leqslant R^2/\gamma^2$. □

2 Logistic Regression

$$F(x; A, k, b) \coloneqq \frac{A}{1 + e^{-k(x-b)}}.$$

Solution (2.1).

- A describes the asymptotic behaviors of F on the "nonzero" side, i.e.,

$$\lim_{x \to -\infty} F(x) = \begin{cases} 0 & \text{if } k > 0 \\ A/2 & \text{if } k = 0 \\ A & \text{if } k < 0 \end{cases} \quad \text{and} \quad \lim_{x \to \infty} F(x) = \begin{cases} A & \text{if } k > 0 \\ A/2 & \text{if } k = 0 \\ 0 & \text{if } k < 0. \end{cases}$$

- b describes the value when F evaluates to 0.5. The graph of F is also radially symmetric across the point $(b, 0.5)$.
- k describes (i) if the asymptotic behaviors of F are "flipped" based on k's sign and (ii) how "steep" does quick does the function change value between it's two asymptotic limits near $x = b$. Large k corresponds to quick shift.

Solution (2.2). From 2.1 we require $A = 1$ and $k > 0$. There is no restriction on b.

Solution (2.3). By a proposition shown in lecture, minimizing the logistic loss is equivalent to finding the MLE for the signoid model. That is,

$$\operatorname*{argmin}_w F(w) = w^* = \operatorname*{argmin}_w \sum_{i=1}^{n} \ell_{\text{logistic}}(y_i(kx_i - b)) = \operatorname*{argmin}_w \sum_{i=1}^{n} \log(1 + \exp[-y_i(kx_i - b))].$$

This quantity is minimized when y_i shares the same sign with $kx - b$ for all $i \in [n]$. From this identity, we see that $y_i > 0$ if and only if $F(x; A, k, b) > 0.5$, so $F = 0.5$ is the minimizer for this classification error.

Solution (2.4). Since

$$\frac{\partial}{\partial x_i} \frac{1}{1 + \exp(-\sum w_i x_i + b)} = \frac{\exp(-\sum w_i x_i + b) w_i}{(1 + \exp(-\sum w_i x_i + b))^2},$$

we have

$$\frac{\mathrm{d}}{\mathrm{d}x} \frac{1}{1 + \exp(-w^T x + b)} = \frac{w^T \exp(-w^T x + b)}{(1 + \exp(-w^T x + b))^2}.$$

- Level set: in order for F to stay constant, $w^T x$ needs to be constant, so the level set is orthogonal to w.
- Direction of gradient: since
$$\frac{\exp(-w^T x + b)}{1 + \exp(-w^T x + b)} \in \mathbb{R},$$
the gradient of F at any point is a scalar multiple of w, i.e., parallel to w.
- The level set corresponds to when $-w^T x + b = 0$, i.e., $w^T x = b$. Projection gives the shortest distance $|b|/\|w\|$.

3 Learning Rectangles

Solution (3.1). Let S the set of data points be given. Let R be the rectangle proposed by our algorithm and let R' be an ERM.

It is clear that $R' \cap S$ cannot contain points absent in $R \cap S$. By assumption, any sample $x \notin R$ corresponds to a negative example. Therefore, $R' \cap S$ had x, R' would have misclassified at least one more point than R, contradicting it's identity as an ERM.

On the other hand, since we have assumed that the training set is indeed generated by some axis-aligned rectangle, $R \cap S$ cannot contain any negative example, for otherwise it would be inside the ground-truth rectangle. Therefore, if $x \in R \cap S$ but not $R' \cap S$, then R' would have again misclassified a positive example as negative, thereby increasing the empirical risk.

Combining the results of the previous two paragraphs, we see $R' \cap S = R \cap S$, i.e., our proposed solution indeed guarantees an ERM.

Solution (3.2). Let D be uniform on $[0,1]^2$. Let B^* be the right half, i.e., $[0.5,1]\times[0,1]$. Let S' be such that $S'\cap B^* = \{(0.5,0.5)\}$, i.e., all but one point in S' is in the right half. In this case $B_{S'}$ would be the degenerate singleton rectangle $[0.5,0.5]\times[0.5,0.5]$, which leads to risk

$$R(f^{\text{ERM}}_{S'}) = \mathbb{P}(x\in B^*, x\neq(0.5,0.5)\mid x\in[0,1]^2) = 0.5$$

since the algorithm will classify all points but $(0.5,0.5)$ as outside the rectangle and any points in B^* will get misclassified (except $(0.5,0.5)$, but a singleton is of null measure in a continuous distribution).

Solution (3.3). Let B_i, $1\leqslant i\leqslant 4$, be given as suggested in the hint. Let n be the sample size. With probability $\leqslant(1-\epsilon/4)^n$, all n samples will fall outside B_1. Applying the same reasoning to other B_i's and using union bound, we see that with probability $4(1-\epsilon/4)^n$, S fails to contain points in at least B_i's, i.e., $S\cap B_i=\varnothing$ for some B_i. Setting an upper bound δ to the probability of such event and using the inequality $1-x\leqslant e^{-x}$, we obtain a sufficient bound

$$4(1-\epsilon/4)^n\leqslant 4e^{-n\epsilon/4}\leqslant\delta$$

which, with some algebra, yields

$$n\geqslant\frac{4\log(4/\delta)}{\epsilon}.$$

For sufficiently large n as specified above, with probability $\geqslant 1-\delta$, S will contain points in B_i for all i, and in doing so

$$B\setminus\bigcup_{i=1}^4 B_i\subset B_S\subset B^*\implies 1-\epsilon\leqslant\mathbb{P}\Big(B\setminus\bigcup_{i=1}^n B_i\Big)\leqslant\mathbb{P}(B_S\mid B^*)\implies R(f^{\text{ERM}}_S)\leqslant\epsilon.$$

Solution (3.4). In this case, we need to consider 2^k k-dimensional boxes. Similar to how each strip in the $\mathbb{R}^2$ case shares one edge with the rectangular B^*, here we require each k-dimensional B_i to share one ($(k-1)$-dimensional) facet with the k-dimensional B^*. In addition the height of each B_i is chosen such that $\mathbb{P}(B_i)\leqslant\epsilon/2^k$. The lower bound for $n=n(\epsilon,\delta)$ is correspondingly

$$n\geqslant\frac{2^k\log(2^k/\delta)}{\epsilon}.$$

□

4 *k*-Nearest Neighbor

`compute_l2_distance`

```python
def compute_l2_distances(Xtrain, X):
    dists = np.zeros((X.shape[0], Xtrain.shape[0]))
    for index, _ in enumerate(X):
        dists[index, :] = np.linalg.norm(X[index, :] - Xtrain, axis=1) # auto shape broadcast
    return dists
```

`predict_labels`

```python
def predict_labels(k, ytrain, dists):
    for index in range(dists.shape[0]):
        # arg sort all distances between training labels and the corresponding test label (index)
        # take the first k elements after sorting
        # bincount find the number of occurences of labels 0 and 1
        # argmax find the label with more occurences
        ypred[index] = np.argmax(np.bincount(ytrain[np.argsort(dists[index])[:k]]))
```

```
    return ypred
```

`compute_error_rate`

```
def compute_error_rate(y, ypred):
    return np.sum(y != ypred) / y.shape[0]
```

Result (4.1): the error rate of our k-NN algorithm in the validation set with $k = 4$ and $\|\cdot\|_2$ is ≈ 0.34667.

`data_processing_with_transformation`

```
    # Min-Max scaling
    def minmax(Xtrain, Xval, Xtest):
        Xtrain_min = np.min(Xtrain, axis=0)
        Xtrain_max = np.max(Xtrain, axis=0)
        Xtrain = (Xtrain - Xtrain_min) / (Xtrain_max - Xtrain_min)
        Xval = (Xval - Xtrain_min) / (Xtrain_max - Xtrain_min)
        Xtest = (Xtest - Xtrain_min) / (Xtrain_max - Xtrain_min)
        return Xtrain, Xval, Xtest
    if do_minmax_scaling:
        Xtrain, Xval, Xtest = minmax(Xtrain, Xval, Xtest)

    # Normalization
    def normalization(X):
        for i in range(X.shape[0]):
            X[i, :] = X[i, :] / np.linalg.norm(X[i, :]) if np.linalg.norm(X[i, :]) != 0 else X[i, :]
        return X

    if do_normalization:
        Xtrain = normalization(Xtrain)
        Xval = normalization(Xval)
        Xtest = normalization(Xtest)
```

Results (4.2): with the same settings as in (4.1), normalized featured vectors yield an error rate of $1/3$ and min-max scaling 0.30667, assuming the scaling on the training data is applied to all three sets.

`compute_cosine_distances`

```
   def compute_cosine_distances(Xtrain, X):
    dists = np.zeros((X.shape[0], Xtrain.shape[0]))
    for index, _ in enumerate(X):
        if (np.linalg.norm(X[index, :]) != 0 and np.linalg.norm(Xtrain, axis=1) != 0).all(): # avoid bool array
            dists[index, :] = 1 - np.dot(X[index, :], Xtrain.T) / (np.linalg.norm(X[index, :]) *
                 np.linalg.norm(Xtrain, axis=1))
        else: dists[index, :] = 1
    return dists
```

Result (4.3): under the same settings as in the previous parts, cosine distance results in an error rate of $1/3$.

`find_best_k`

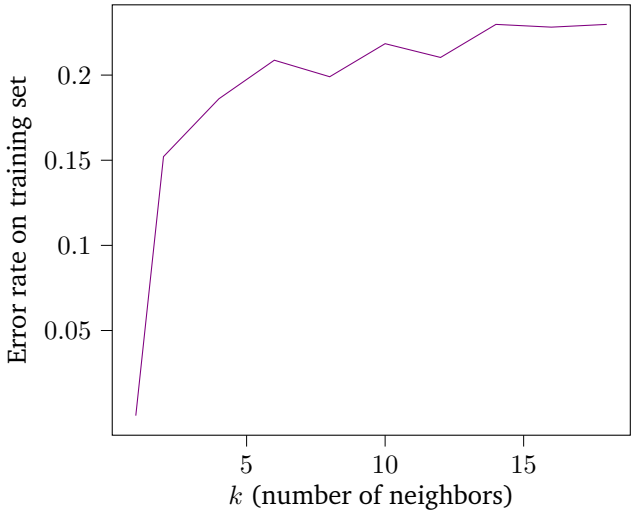
```
def find_best_k(K, ytrain, dists, yval):
```

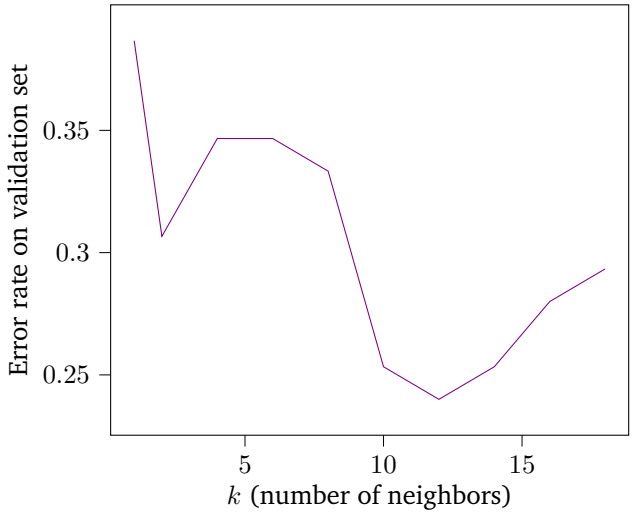
```
    all_err = []
    for single_k in K:
        ypred = predict_labels(single_k, ytrain, dists)
        err = compute_error_rate(yval, ypred)
        all_err.append(err)
    return K[np.argmin(np.array(all_err))], all_err, min(all_err)
```

Results (4.4):

Below are two plots representing our model's error rates on the training set (left) and validation set (right). As k increases, the error rates on the training set increases as well. On the validation set, however, the error rate reaches a minimum at $k = 12$ before increasing again. The final test set error rate using $k = 12$ is ≈ 0.21333.

Although a very weak relation, it seems like for relative small k's ($\leqslant 12$), there roughly exists a negative correlation between the two errors. Potential explanations include "the training set includes many data that are 'hard' to learn" or "the validation set consists of overly 'easy' examples." Nevertheless, even at $k = 12$, the error rates on the validation set exceeds that on the training set. This can be explained by the inevitable generalization gap once we move to data outside the training set.





5 Linear Regression

5.1. Below is the result of one run.

- Total squared error of w_{LS} on the training set: 222.6078655833394.
- Total squared error of the zero vector w_0: 83144.16480657135.
- Total squared error of w_{LS} on the test set: 262.1390061685291.

Due to the sample's randomness, each run results in different outcomes, but in general they follow the same trend, that w_{LS}'s error rate on the test set is only slightly higher than that on the training set (262 compared to 223), showing that w_{LS} is indeed a good choice (as we would expect).

5.2. The gradient of F is

$$\nabla F(w) = \sum_{i=1}^{n} \nabla f_i(w) = \sum_{i=1}^{n} 2x_i(w^T x_i - y_i).$$

Below is the outcome of one run:

(1) $\eta = 0.0005$, final total squared error = 3018.547479118958.

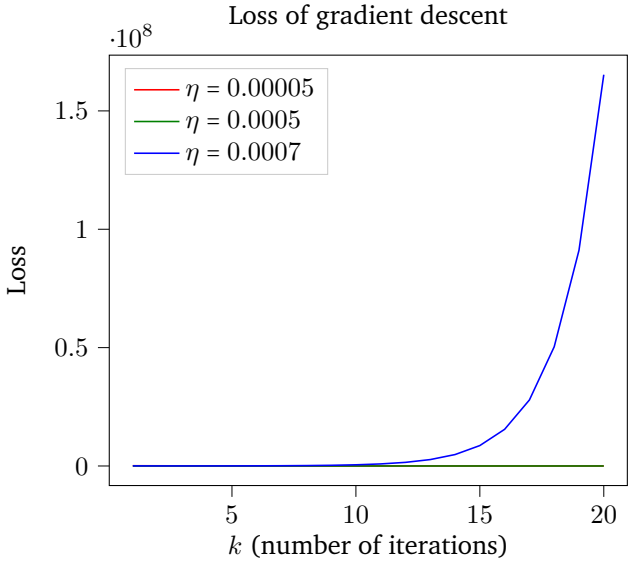
(2) $\eta = 0.005$, final total squared error = 232.14862115297325. (Total squared error of w_{LS} in this case is 232.1340536673857.)

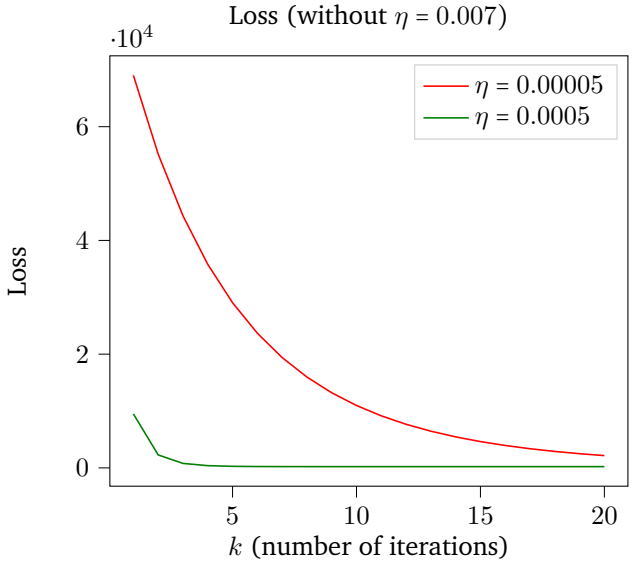
(3) $\eta = 0.007$, final total squared error = 4728265838.519294.

The first observation is $\eta = 0.005$ successfully approaches the minimizer of F in only 20 iterations. (In fact most of the times it only takes about 15 iterations to reach $\min F + 1$.) This turns out to the most optimal η among the three provided.

Comparing (1) and (2), along with the bottom right figure below, we see that the loss function of $\eta = 0.0005$ decreases, albeit at a much slower rate. In fact, if we iterate 80 times in total, $\eta = 0.0005$ also gives a squared loss error very close to that of w_{LS}.

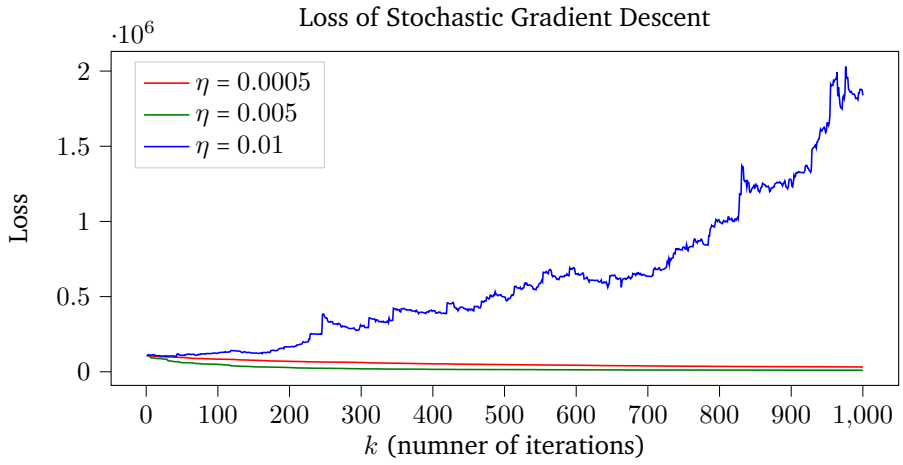
On the other hand, when $\eta = 0.0007$, the loss blows up usually early on (usually starting at second or third iteration). This is most likely due to the large learning rate that "overshoots" the global minimum, causing a chain reaction, and the algorithm deviates further and further from its goal.





5.3. See the plot on the next page. Comparing the red ($\eta = 0.0005$) line and the green ($\eta = 0.005$) line, we see that within a reasonable range (e.g., η not too large) smaller η leads to slower descent, but the algorithm still converges to the minimizer of F. The blue line ($\eta = 0.01$), however, similar to 5.2, shows that an overly large η will lead to SGD blowing up as well. Among the three η's provided, 0.005 is the most optimal.

In this run, after 1000 iterations, $\eta = 0.0005, 0.005$, and 0.01 lead to a squared error of $32035, 10406$, and 1839037, respectively. Compared to the previous problem, the descents are much slower but so is the blow up. In 5.2, each data point is used 20 times; here, on average, each data point is used once (1000 runs, each time picking one among 1000 data points). If we were to address this discrepancy and iterate $\eta = 0.005$ $20,000$ times, the resulting squared error is still in thousands, relatively far from the minimum of the loss function.


Loss of Stochastic Gradient Descent
$\cdot 10^6$
2
1.5
1
0.5
0
Loss
$\eta = 0.0005$
$\eta = 0.005$
$\eta = 0.01$
0
100
200
300
400
500
600
700
800
900
1,000
k (numner of iterations)