CSCI567 HW1

Angzhan He, Gaoyuan Jiang

4334670919, 2198646040

February 3, 2024

1.1

$$\begin{aligned} w_{k+1}^T w_{opt} &= (w_k + y_i x_i)^T w_{opt} \\ &= w_k^T w_{opt} + y_i x_i^T w_{opt} \\ &= w_k^T w_{opt} + y_i w_{opt}^T x_i \\ &\geq w_k^T w_{opt} + \gamma ||w_{opt}||_2 \end{aligned}$$

1.2

$$\begin{aligned} w_{k+1}^T w_{k+1} &= (w_k + y_i x_i)^T (w_k + y_i x_i) \\ &= w_k^T w_k + w_k^T y_i x_i + y_i x_i^T w_k + y_i^2 x_i^T x_i \\ &= ||w_k||_2^2 + 2 y_i w_k^T x_i + y_i^2 ||x_i||_2^2 \\ &\leq ||w_k||_2^2 + y_i^2 R^2 \\ &\leq ||w_k||_2^2 + R^2 \end{aligned}$$

1.3

$$w_{k+1}^T w_{opt} \geq w_k^T w_{opt} + \gamma ||w_{opt}||_2$$
$$M updates$$
$$w_{k+1}^T w_{opt} \geq \gamma M$$
$$||w_{k+1}||_2 ||w_{opt}||_2 \geq w_{k+1}^T w_{opt} \geq \gamma M$$
$$||w_{k+1}||_2 \geq \gamma M$$

$$||w_{k+1}||_2^2 \leq ||w_k||_2^2 + R^2$$
$$M updates$$
$$||w_{k+1}||_2^2 \leq R^2 M$$
$$||w_{k+1}||_2 \leq R\sqrt{M}$$

1.4

$$\gamma M \leq ||w_{k+1}||_2 \leq R\sqrt{M}$$
$$\gamma^2 M^2 \leq R^2 M$$
$$M \leq R^2/\gamma^2$$

2.1

Algorithm: For all positive data points in the training set, find the smallest and the largest values of

x_1 and x_2, which correspond to a_1, b_1, a_2 and b_2 respectively.
Proof: The realizable assumption shows that there exists a rectangle B^* that perfectly classifies the training data. The rectangle B_S we get by the algorithm have an empirical risk of 0, which is the minimum possible. Thus, the rectangle B_S is an empirical risk minimizer.

2.2

From a probabilistic perspective and with respect to $0-1$ loss, $R(f_{S'}^{ERM}) \geq 0.5$ indicates the misclassifying probability is greater than 0.5. If we need to let this model classify every data points in training set $\{(\mathbf{x}, y), i \in [n]\}$ correctly, it means we can not select any data point from $B^* - B_{S'}$. The probability mass (with respect to D) of $B^* - B_{S'}$ is larger than 0.5. If we draw data point i.i.d from distribution D, then the possibility of selecting such a bad training set of size n is less than 0.5^n, which is non-zero, but very small when n is large enough.

2.3

Step1: According to the definition of empirical risk minimizer and realizability assumption, B_S must be contained within B^* to have zero empirical risk.
Step2: B_i has a probability mass of $\varepsilon/4$ by construction, since there are four such rectangles, the combined B_S where f_S^{ERM} could potentially fail to classify correctly is less than $4 \times \varepsilon/4 = \varepsilon$.
Step3: The probability that none of the n examples in S are in B_i is $(1-\varepsilon/4)^n$.

$$
\begin{aligned}
P &= (1-\varepsilon/4)^n \\
log(P) &= nlog(1-\varepsilon/4) \\
log(P) &\leq -n(\varepsilon/4) \\
log(P) &\leq log(\delta/4) \\
P &\leq \delta/4
\end{aligned}
$$

Step4: The union bound states that the probability of at least one of a set of events occurring is on greater than the sum of the probabilities of the individual events. Apply this to the probability that S does not contain an example from each B_i, the sum of the probability is $4 \times (\delta/4) = \delta$. Thus, the probability that S contains all examples from each B_i is at least $1-\delta$.

2.4

In R^d, define $2d$ critical regions (similar to the B_i rectangles from the 2-dimensional case) surrounding the true B^*, each with a probability mass of $\varepsilon/(2d)$ with respect to the distribution D.
If S contains positive examples in all of the critical regions, then $R(f_S^{ERM}) \leq (2d) \times (\varepsilon/(2d)) = \varepsilon$.
According to the union bound, for each of the $2d$ critical regions, we want the probability that the sample S does not contain a positive example from that region to be less than $\delta/(2d)$.

$$
\begin{aligned}
P &= (1-\varepsilon/(2d))^n \\
log(P) &= nlog(1-\varepsilon/(2d)) \\
log(P) &\leq -n(\varepsilon/(2d))
\end{aligned}
$$

To make $P \leq \delta/(2d)$, we get $n \geq \frac{2dlog(2d/\delta)}{\varepsilon}$.

3.1

The validation error rate is 0.07692307692307693 when k = 4 using Euclidean distance

3.2

k nearest neighbor algorithm in the validation set for k = 4 using Euclidean distance when data is using

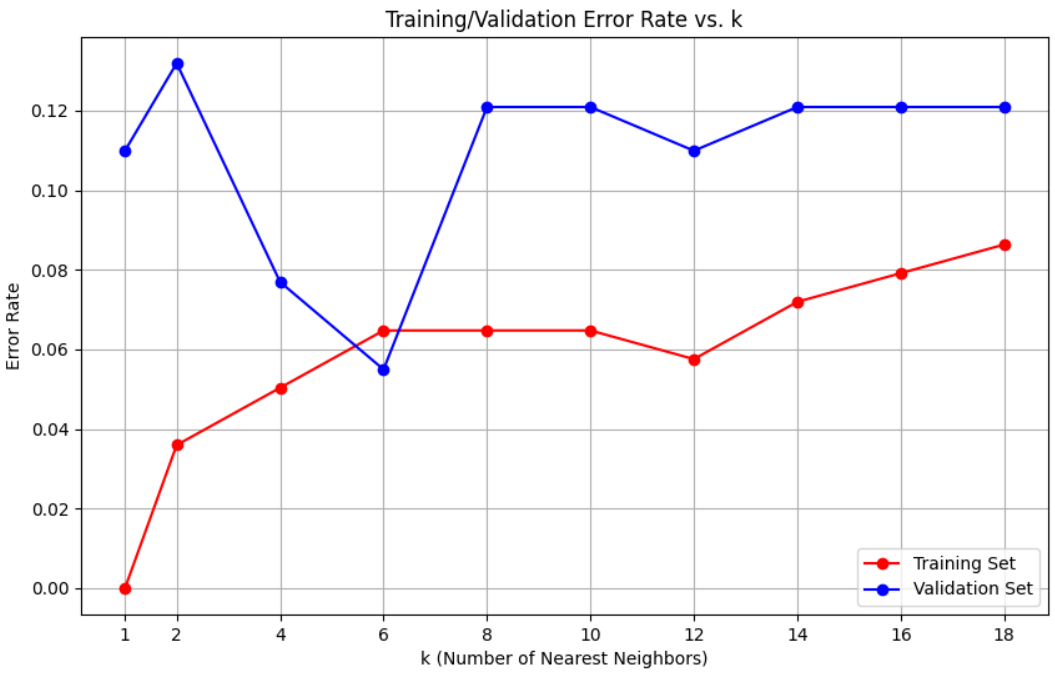
(1) Normalized featured vector, error rate is 0.08791208791208792

(2) Min-max scaling featured vector, error rate is 0.054945054945054944

3.3

The error rate of k nearest neighbor algorithm in the validation set for k = 4 using cosine distance for *original data without data transformation* is 0.04395604395604396

3.4



(1) The error rate is zero when k=1. The error rate goes up when k grows.

(2) Among the tested values, the best k is 6.

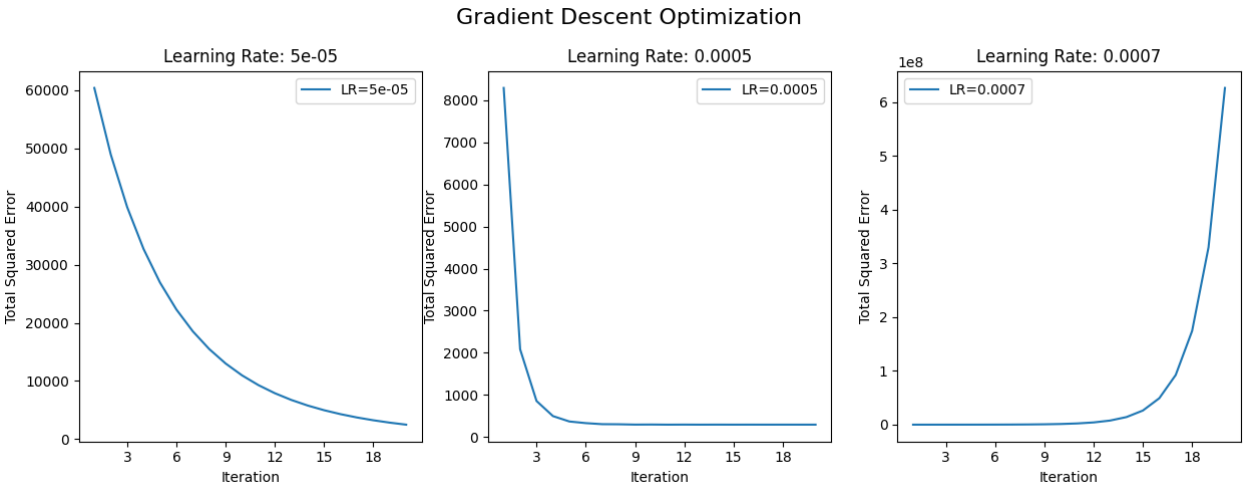
(3) Generally, the training error rate is lower than the validation error rate for the same k. When k = 6, the error rate of the validation set is a little smaller than training set. The best k values for training set and validation set are different.
(4) The best test set error rate by using best k = 6 is 0.07100591715976332
(5) When using training set to tune k, you will get the lowest error rate when k = 1. This is caused by overfitting. If you use a set of data points to tune some hyperparameters, you cannot use the same set of data to validate your model. This will overfit your model. You can do well during validation, but the test error rate will be large when applying the model on unseen data points. This is consistent with the graph above. We can see when k = 1, the error rate by using the training set is zero, but the error rate by using a validation set of unseen date points is large. In other words, this overfitting model is not good in terms of generalization since the generalization gap is very large. To get the best k, we need to use a validation set to do hyper-parameter tuning. It will produce a real good hyper-parameter k.

4.1

F(w_LS)= 217.48452613173987 on training data
F(w_0)= 78885.82819617869 on training data
F(w_LS)= 294.06836989399153 on testing data
The gap in the training and test objective function values is 76.58384376225135.

Comment: The total square error of setting w to be all 0's vector is much larger than the total square error of setting w to the closed form solution. The total squared error of w_{LS} on these test points set is almost the same with that on the training points set, indicating a very small generalization gap and a good model.

4.2 We use test dataset to get the final objective value!



(1) *the gradient of f at w_t:*

$$\nabla f(w) = 2(w_t^T x_i - y_i)x_i$$

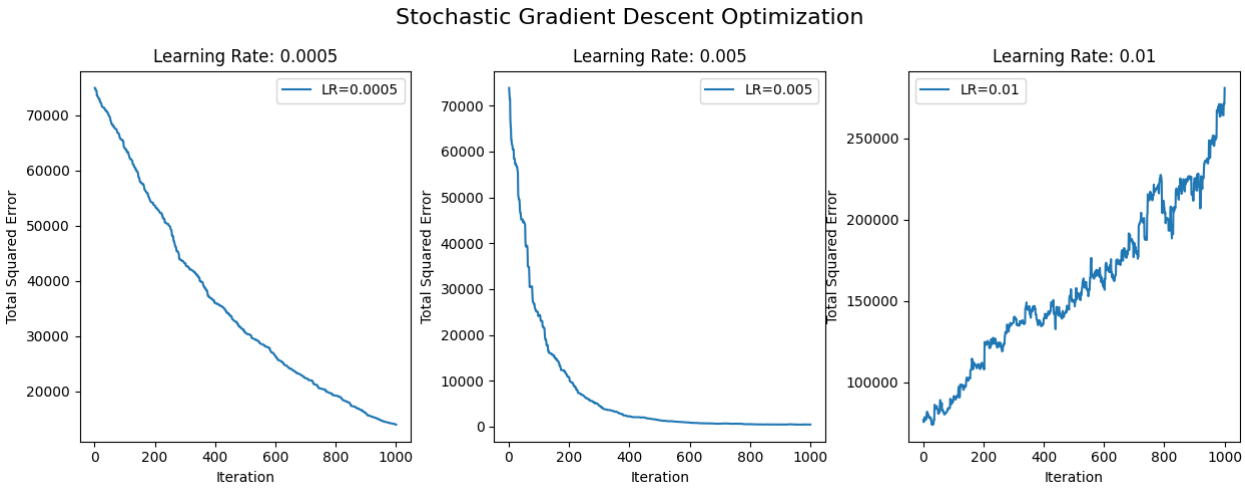
$$\nabla F(w) = \sum_{i=1}^{n} 2(w_t^T x_i - y_i)x_i$$

(2) *How the step size can affect the convergence of gradient descent*

Large step size can let GD converge faster than small step size. But at the same time, if the step size is too larger, GD will not converge at all. Therefore, there is a tradeoff between step size and converge speed. Small step size also has a little risk of getting stuck in local minimum.

(3) best step size is 0.0005 and its corresponding objective function value is 294.1191567177276

4.3



(1) *how the step size can affect the convergence of stochastic gradient descent and how it compares to gradient descent*

Effect of lr on convergence is generally similar to that of gradient descent.

Effect of same lr on convergence is not as strong as gradient descent.

(2) *Compare the performance of the two methods.*

SGD needs more iterations to converge than SD. However, every iteration of SGD takes less time. Usually, SGD is faster than SD.

(3) *How do the best final objective function values compare?*

The best final objective function value of GD is better than that of SGD.

(4) *How many times does each algorithm use each data point? Also report the step size that had the best final objective function value and the corresponding objective function value.*

GD used each data point N_iter times, while SGD used each data point (N_iter / n) times on average.

GD: final objective value is 294.1191567177276 and best step size is 0.0005.

SGD: final objective value is 467.0480113466759 and best step size is 0.005.