CS567 HW1

Ritu Pravakar and Charlene Yuen

September 2022

1 Perceptron Convergence

1.1

$$w_{k+1}^T w_{opt} = (w_k + y_i x_i)^T w_{opt} = (w_k^T + x_i^T y_i) w_{opt} = w_k^T w_{opt} + x_i^T y_i w_{opt}$$

Since y_i is a scalar and $x_i^T w_{opt} = w_{opt}^T x_i$ we can rearrange the terms to use the given statement that $y_i(w_{opt}^T x_i) \geq \gamma$.

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

1.2

$$\|w_{k+1}^T\| = \|w_k + y_i x_i\|$$

By the triangle inequality:

$$\begin{aligned} \|w_{k+1}^T\| &\leq \|w_k\| + \|y_i x_i\| \\ &\leq \|w_k\| + y_i \|x_i\| \end{aligned}$$

Since $\|x_i\| \leq R$

$$\begin{aligned} \|w_{k+1}^T\| &\leq \|w_k\| + y_i R \\ \|w_{k+1}\|^2 &\leq \|w_k^2\| + 2 y_i R \|w_k\| + yi^2 R^2 \\ &\leq \|w_k\|^2 + R^2 \end{aligned}$$

1.3

From 1.2:

$$\begin{aligned} \|w_{k+1}\|^2 &\leq \|w_k\|^2 + R^2 \\ &\leq (\|w_k\|^2 + R^2)M \\ \|w_{k+1}\| &\leq \sqrt{\|w_k\|^2 M + R^2 M} \\ &\leq \sqrt{R^2 M} = R\sqrt{M} \end{aligned}$$

From 1.1:

$$w_{k+1}^T w_{opt} \leq \|w_{k+1}\|\|w_{opt}\| \leq \|w_{k+1}\|$$

since $\|w_{opt}\| = 1$

$$\begin{aligned} \|w_{k+1}\| &\geq w_k^T w_{opt} + \gamma\|w_{opt}\| = w_k^T w_{opt} + \gamma \\ &\geq w_k^T w_{opt} M + \gamma M \\ &\geq \gamma M \end{aligned}$$

Therefore,

$$\gamma M \leq \|w_{k+1}\| \leq R\sqrt{M}$$

1.4

From 1.3

$$\begin{aligned} \gamma M &\leq R\sqrt{M} \\ \gamma^2 M^2 &\leq R^2 M \\ M &\leq R^2/\gamma^2 \end{aligned}$$

2 Logistic Regression

2.1

The parameter A shifts the upper bound of the function up or down from [0, A]. The parameter B shifts the function horizontally across the x-axis. The parameter k adjusts the curvature of the graph and reflects the graph depending on if it is positive or negative.

2.2

The properties of a CDF include values bounded between 0 and 1, the limit to infinity being 1, and the limit to negative infinity being 0. For *F(x;A,k,b)* to be a CDF, it must satisfy those three properties with A=1, $b \in \mathbb{R}$, and k<0.

2.3

Intuitively, setting the threshold for *F(x;A,k,b)* to be 1/2 or more minimizes the classification error because you would want to be at least 50% certain before predicting a label. The higher the threshold, the more likely your prediction is correct, which would minimize classification error.

2.4

1) level sets => a) parallel to w; since the level sets are a tangent parallel to w, and one can see that w being in R^2 lays parallel to the curves on the surface.
2) direction of gradient at any point => e) orthogonal to w; since gradients are defined as orthogonal to level sets (parallel to w).
3) distance of level set F(x)=1/2 from origin => c) $|b|/\|w\|$; since both b and w impact its distance. For example, with values $w_1 = 0.1, w_2 = -5, b = 20$, we arrive at $20/5.001 \approx 4$. We take the absolute value of b because F(x)=1/2 just ends up on different sides of the axis depending on if it's negative or not with the same distance. The norm of w also scales the distance accordingly.

3 Learning rectangles

3.1

Loss function: 0-1 loss, Function class: axis-aligned rectangles
Next, we look at the empirical risk:

$$R_{emp}(f_{a_1,b_1,a_2,b_2}(x)) = (1/n)\sum_{i=1}^{n} l(f_{a_1,b_1,a_2,b_2}(x_i))$$

In order to get the corners of the smallest rectangle B*, we minimize over possible values of a and b for each possible f:

$$(a_1^*, b_1^*, a_2^*, b_2^*) = \underset{(a_1,b_1,a_2,b_2)}{\operatorname{argmin}} R_{emp}(f_{(a_1,b_1,a_2,b_2)})$$

The algorithm can thus be framed as an empirical risk minimizer.

Additionally, by the realizability assumption, the f* corresponding to the B* from the algorithm also minimizes risk: $\hat{R}_s(f^*) = 0$

3.2

With D as a Gaussian distribution and B* with values from far left or far right of the distribution, it is possible that there is a non-zero probability of seeing a bad training set. Operating on the i.i.d assumption on D, it may be possible that a non-uniform distribution like a normal distribution may result in numbers closer to be mean to be more likely, so a B* with values far from the mean may result in errors and bias. Concentrated data used for training may also result in points outside of the training set having incorrect labels after prediction, resulting in the ERM to not generalize as well.

3.3

To show $B_s \subseteq B^*$, B_s is defined as the rectangle returned by the algorithm based on sample S, which should be the smallest possible rectangle with positive samples. Therefore, it must within or equal to the valid rectangle defined by the true labels.

$R(f_s^{ERM})$ is the total risk from the algorithm if it misclassifies outside B_S and in B*. Since landing in the rectangles B_1, B_2, B_3, and B_4 are bounded by a probability $\frac{\epsilon}{4}$ for when they are entirely outside B_S:

If S contains examples in B_i from i=1...4, from the union bound:

$$\begin{aligned} R(f_S^{ERM}) = Pr_D(B^* \backslash B_S) &\leq \sum_{i=1}^{4} Pr_D(B_i) = 4 * \frac{\epsilon}{4} \\ &\leq \epsilon \end{aligned}$$

Upper bound the probability S doesn't contain example from B_i for all n points, i.e. when $R(f_S^{ERM}) > \epsilon$:

$$Pr_D(S \notin B_i) < (1 - \frac{\epsilon}{4})^n$$

$$Pr_D(S \notin B_{1,2,3,4}) < \sum_{i=1}^{4} Pr_D(S \notin B_i) = 4(1 - \frac{\epsilon}{4})^n$$

Since $1 - x \leq e^{-x}$:

$$\leq 4e^{-n\epsilon/4}$$

Using the assumption $n \geq \frac{4\log(4/\delta)}{\epsilon}$, the probability that S is a bad training set can be shown and $R(f_S^{ERM}) > \epsilon$:

$$Pr_D(S \notin B_{1,2,3,4}) \leq 4e^{-(\epsilon/4)*\frac{4\log(4/\delta)}{\epsilon}} = 4e^{log(\delta/4)} = \delta$$

Therefore, with probability $1-\delta$, $R(f_S^{ERM}) \leq \epsilon$ with this value of n.

3.4

To generalize the problem to 3.3, we will simply redefine the classifier to contain d points. For real numbers $a_1 \leq b_1, ...a_d \leq b_d$, the classifier $f_{(a_1,b_1,...,a_d,b_d)}$ on input x with coordinates $(x_1, ..., x_d)$:

$$f_{(a_1,b_1,...,a_d,b_d)}(x_1, ..., x_d) = \begin{cases} 1, & a_i \leq x_i \leq b_i, \forall i \in 1...d \\ -1, & \text{otherwise} \end{cases}$$

We also redefine the function class of all axis-aligned rectangles:

$\mathcal{F}_{rec}^2 = \{f_{(a_1,b_1,...,a_d,b_d)}(x_1, ..., x_d) : a_i \leq x_i \leq b_i, \forall i = 1...d\}$

From here, the proof becomes essentially the same but just with d points instead of 2, so $B_s \subseteq B^*$ also holds. We proceed with the steps similarly:

If S contains examples in B_i from i=1...d, from the union bound:

$$\begin{aligned} R(f_S^{ERM}) = Pr_D(B^* \backslash B_S) &\leq \sum_{i=1}^{2d} Pr_D(B_i) = 2d * \frac{\epsilon}{2d} \\ &\leq \epsilon \end{aligned}$$

Upper bound the probability S doesn't contain example from B_i for all n points, i.e. when $R(f_S^{ERM}) > \epsilon$:

$$\begin{aligned} Pr_D(S \notin B_i) &< (1 - \frac{\epsilon}{2d})^n \\ Pr_D(S \notin B_{1,...,d}) &< \sum_{i=1}^{2d} Pr_D(S \notin B_i) = 2d(1 - \frac{\epsilon}{2d})^n \end{aligned}$$

Since $1 - x \leq e^{-x}$:

$$\leq 2de^{-n\epsilon/2d}$$

We can thus see that $n \geq \frac{2d \log(2d/\delta)}{\epsilon}$ should show probability that S is a bad training set and $R(f_S^{ERM}) > \epsilon$:

$Pr_D(S \notin B_{1,...,d}) \leq 2de^{-(\epsilon/2d) * \frac{2d \log(2d/\delta)}{\epsilon}} = 2de^{log(\delta/2d)} = \delta$

Therefore, with probability $1-\delta$, $R(f_S^{ERM}) \leq \epsilon$ with this value of n.

4 k-nearest neighbor classification and the ML pipeline

4.1 Report 4-nearest neighbor accuracy

Report item:
The validation error rate is 0.3466666666666667 when k=4.

4.2 Data transformation

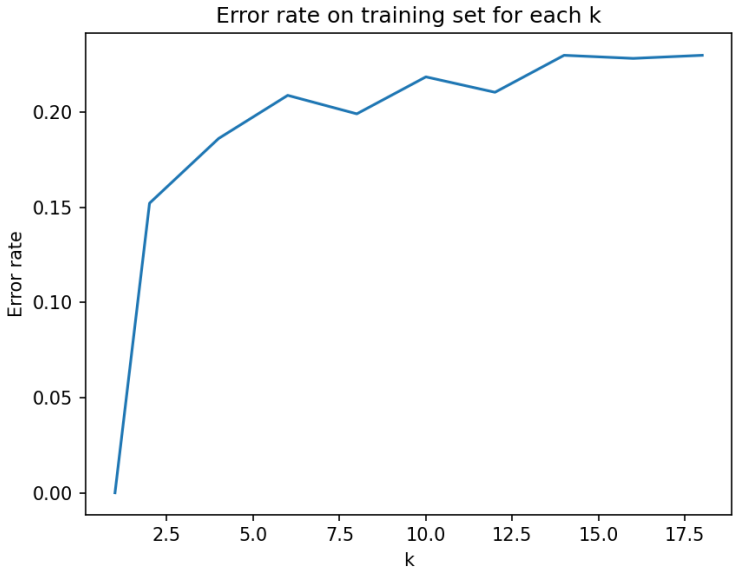
Report item:
(1) Normalized validation error: 0.3333333333333333
(2) Min-max scaling validation error: 0.32

4.3 Different distance measurement

Report item:
The validation error rate is 0.32 for cosine distance.

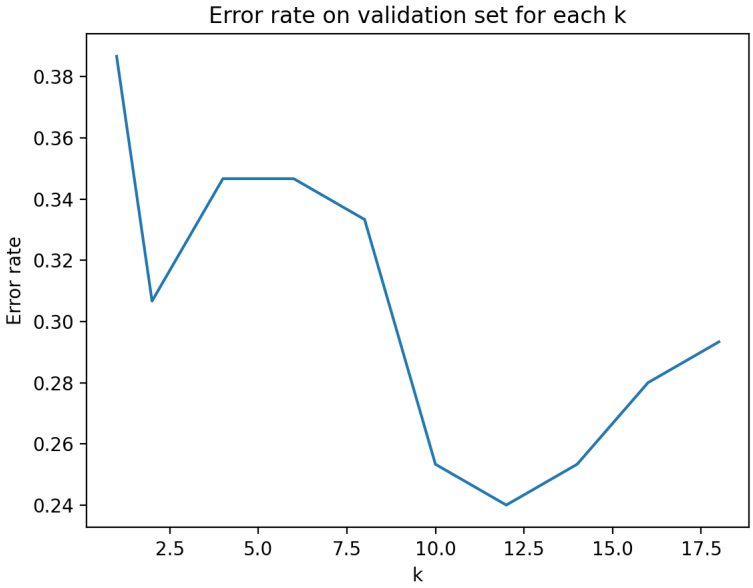
4.4 Tuning the hyper-parameter k

Report item:
(1) For the error rates on the training set:



We observe that as k increases, the error rate on the training set also spikes up and increases. There does not seem to be any particular minimum error rate with the best k, as it increases exponentially until k=2.5, then to k=7.5 and then sporadically until k=18.

(2) For the error rates on the validation set:



The best k=12 has the validation error rate of 0.24.

(3) There are no specific similarities between the two curves. The validation curve overall has a general trend of increasing k resulting in lower error rates and its error rate decreases exponentially until k=2.5, in comparison to the opposite happening with the training curve. However, despite the error rate increase in the training curve, its error caps at around 0.22, while the validation curve starts around 0.38 and decreases from there. The last increase in the validation curve from k=12.5 to k.18 also somewhat aligns with the training curve, as it also has a slight increase there.

(4) Using the best k, the final test error rate is 0.21333333333333335.

(5) Overall, the final validation error rate is a large improvement from the original 4-nearest neighbor accuracy, from 0.346 to 0.24 due to the increase in k=12 neighbors. The final test error rate of 0.213 was also a massive improvement, as it was even smaller than the validation error rate. The results on the training set and validation set allowed us to see that a test set similar to the training set will result in overall smaller error rates, and the validation results showed improvement with an increase in k so more neighbors are compared.

5 Linear Regression

5.1 Least squares regression

$$F(w_{LS}) = 224.74341295$$

When w is an all zero vector:

$$F(w) = 96560.70322922$$

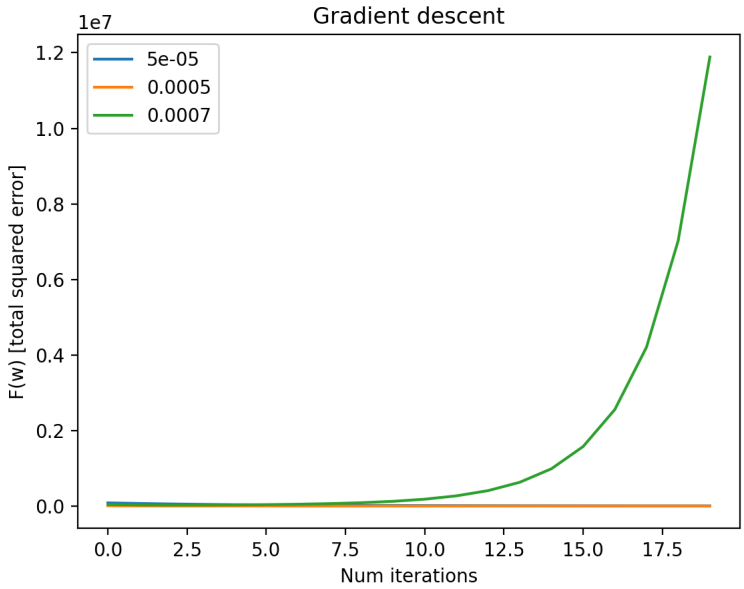
Drawing new 1000 data points:

$$F(w_{LS}) = 287.66763287$$

The gap between the training and objective values is 62.92421992, which seems reasonable as drawing new test data points will result in the w_{LS} from before having some amount as error as it was trained on the training data points.

5.2 Gradient descent

$$\nabla f_{w_t} = 2(w^T x_t - y_t)x_t$$
$$\nabla F_{w_i} = \sum_{i=1}^{n} 2(w^T x_i - y_i)x_i$$

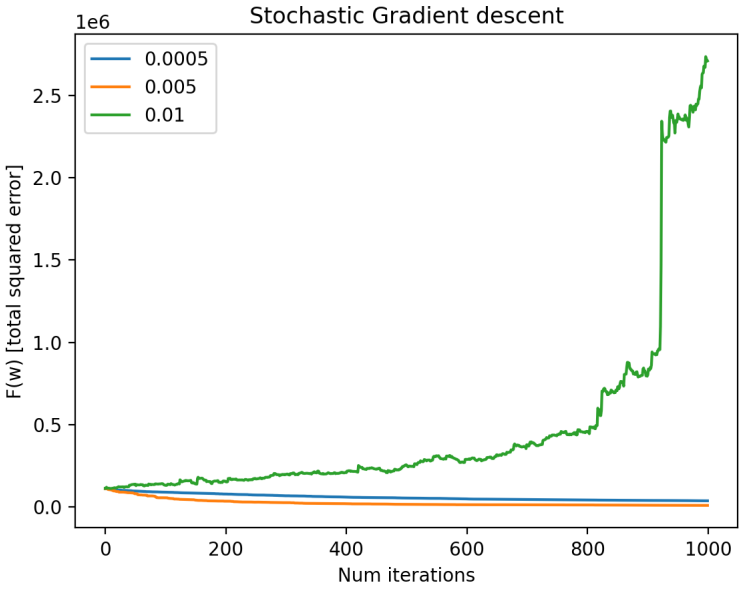


A larger stepsize tends to just exponentially grow to infinity, which was shown

by stepsize 0.0007. On the other hand, a small stepsize maintains a very high error rate and doesn't converge fast enough within the 20 iterations given, as shown by stepsize 0.00005. A reasonable stepsize not too large or small, found through trial and error, will start at a reasonable error and decrease the error rate at a good rate within the iterations given, minimizing the error, which was 0.0005 in this case.

Overall, the stepsize 0.0005 had the best objective function value of 242.083.

5.3 Stochastic Gradient Descent



Similar to GD, SGD also had the best performance with the middle stepsize which wasn't too large or too small (0.005), while the error blew up exponentially with a larger stepsize (0.01) and converged slower with a smaller stepsize (0.0005). The overall best final objective value from SGD was much larger than that of GD, due to stochasticity from picking a random data point, which may not update w as accurately. However, for this reason SGD is more efficient since it only uses each datapoint once for each iteration, while GD uses all n datapoints for each iteration.

Overall, the stepsize 0.005 had the best objective function value of 92598.304.