

CS 475 Machine Learning (Fall 2023): Assignment 2

Due on **October 18th**, 2023 at 4:00 PM ET

Instructions: Please read these instructions carefully and follow them precisely. Feel free to ask the instructor if anything is unclear!

1. Please submit your solutions electronically via [Gradescope](#).
2. Please submit a PDF file for the written component of your solution including derivations, explanations, etc. You can create this PDF in any way you want: typeset the solution in LATEX (recommended), type it in Word or a similar program and convert/export to PDF. You may not hand write the solution. We recommend that you restrict your solutions to the space allocated for each problem; you may need to adjust the white space by tweaking the argument to `\vspace{xpt}` command. Please name this document `<firstname-lastname>-sol2.pdf`.
3. Submit the empirical component of the solution (Python code and the documentation of the experiments you are asked to run, including figures) in a Jupyter notebook file.
4. **Late submissions:** You have a total of 96 late hours for the entire semester that you may use as you deem fit. There is no additional grace period for this submission. After you have used up your quota of 96 late hours, there will be a penalty of 100% of your grade on a late submission for this assignment.
5. **What is the required level of detail?** When asked to derive something, please clearly state the assumptions, if any, and strive for balance: justify any non-obvious steps, but try to avoid superfluous explanations. When asked to plot something, please include the figure as well as the code used to plot it (and clearly explain in the README what the relevant files are). If multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers). When asked to provide a brief explanation or description, try to make your answers concise, but do not omit anything you believe is important. When submitting code, please make sure it's reasonably documented, and describe succinctly in the written component of the solution what is done in each `py`-file.

Name: _____

I. Error Decomposition

We saw in class that we can decompose the true risk of the least squares solution in terms of approximation error and the estimation error. Note though that risk of the least squares is a random variable since it is obtained on random draw of an i.i.d. sample from the population. Here we investigate the generalization error (aka, the expected risk) of the least squares procedure, where the expectation is over the distribution over the training data.

Learning algorithm \mathcal{A} . We consider the usual setting with $\mathcal{X} \subseteq \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$. Let P_{XY} be a joint distribution over $\mathcal{X} \times \mathcal{Y}$, and $S \sim P_{XY}^n$ denote an i.i.d. sample of size n . Let $\mathcal{H} = \{h : \mathcal{X} \rightarrow \mathcal{Y}\} \subset \mathcal{Y}^{\mathcal{X}}$ represent the hypothesis class and $\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^* \rightarrow \mathcal{H}$ denote the learning algorithm (e.g., empirical risk minimization) that given a dataset $S = \{(x_i, y_i)\}_{i=1}^n \sim P_{XY}^n$ returns the predictor

$$\hat{h}_S \triangleq \mathcal{A}(S). \quad (1)$$

Expected Output \bar{h} of the Learning Algorithm. Given the learning rule \mathcal{A} , we denote by \bar{h} the expected output of the algorithm where the expectation is with respect to the distribution over the training dataset:

$$\bar{h} = \mathbb{E}_{S \sim P_{XY}^n} [\hat{h}_S]. \quad (2)$$

Bayes Optimal Regression Function. The Bayes optimal rule $h^* \in \mathcal{Y}^{\mathcal{X}}$ is the unrestricted map from $\mathcal{X} \rightarrow \mathcal{Y}$ that minimizes the population risk, $h^* = \arg \min_{h: \mathcal{X} \rightarrow \mathcal{Y}} \mathbb{E}_{(x,y) \sim P_{X,Y}} [(y - h(x))^2]$. We showed in class that

$$h^*(x) = \mathbb{E}_{Y|X}[Y|X = x], \quad (3)$$

where expectation is with respect to the conditional distribution $P_{Y|X}$ of the labels given the inputs. In other words, the Bayes optimal predictor assigns the label you would expect to obtain, given a feature vector x .

Expected Risk (aka, generalization error) of \mathcal{A} . We measure the performance of any learning algorithm \mathcal{A} for least squares regression in terms of its expected risk,

$$\mathbb{E}_{S \sim P_{XY}^n} \mathbb{E}_{(x,y) \sim P_{XY}} [(y - \hat{h}_S(x))^2]. \quad (4)$$

In this problem, we will decompose the generalization error into three meaningful terms.

Problem 1 [20 points]

(a) We begin by showing that

$$\mathbb{E}_{S \sim P_{XY}^n} \mathbb{E}_{(x,y) \sim P_{XY}} [(y - \hat{h}_S(x))^2] = \mathbb{E}_{S \sim P_{XY}^n} \mathbb{E}_{(x,y) \sim P_{XY}} [(\bar{h}(x) - \hat{h}_S(x))^2] + \mathbb{E}_{(x,y)} [(y - \bar{h}(x))^2]. \quad (5)$$

We refer to the first term in the decomposition as “variance.” Discuss why.

Hint: Write $y - \hat{h}_S(x)$ as $y - \bar{h}(x) + \bar{h}(x) - \hat{h}_S(x)$. Expand the square and argue that the cross-term is zero.

- (b) Show that we can further express the second term in the decomposition above (on the RHS of Equation 5) as follows.

$$\mathbb{E}_{(x,y) \sim P_{XY}} [(y - \bar{h}(x))^2] = \mathbb{E}_{(x,y) \sim P_{XY}} [(y - h^*(x))^2] + \mathbb{E}_{(x,y) \sim P_{XY}} [(h^*(x) - \bar{h}(x))^2]. \quad (6)$$

We refer to the terms on the RHS above as “noise” and “bias-squared.” Discuss why.

Hint: Write $y - \bar{h}(x)$ as $y - h^*(x) + h^*(x) - \bar{h}(x)$. Expand the square and argue that the cross-term is zero.

- (c) Discuss how the decomposition of the expected error in terms of bias, variance and the noise terms informs the design of a good learning algorithm.

II. Ridge Regression

We saw in class that maximizing log-likelihood under the Gaussian noise model is equivalent to minimizing squared loss. In this problem, we will show that maximizing the log-posterior is equivalent to ridge (L_2 -regularized) linear least squares regression. We consider the following Gaussian noise model: $y = \mathbf{w}^\top \mathbf{x} + \nu$, where $\nu \sim \mathcal{N}(\nu; 0, \sigma^2)$.

Additionally, we assume some belief about the value of \mathbf{w} before seeing any data. In particular, we assume a Gaussian prior distribution on $\mathbf{w} \in \mathbb{R}^d$ (note that we do not regularize for the coefficient, w_0 , of the constant term): $\mathbf{w} \sim \mathcal{N}(\mathbf{w}; 0, \frac{1}{\mu} \mathbf{I})$, where $\mu > 0$ is the “precision” (inverse variance) of the prior distribution.

Given n training examples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$, let $\mathbf{X} \in \mathbb{R}^{n \times d}$ denote the design matrix, and $\mathbf{y} \in \mathbb{R}^n$ denote the response vector. Instead of log-likelihood, the objective becomes log-posterior, $p(\mathbf{w}|\mathbf{y}, \mathbf{X})$, and we denote the maximum a-posteriori (MAP) estimate as

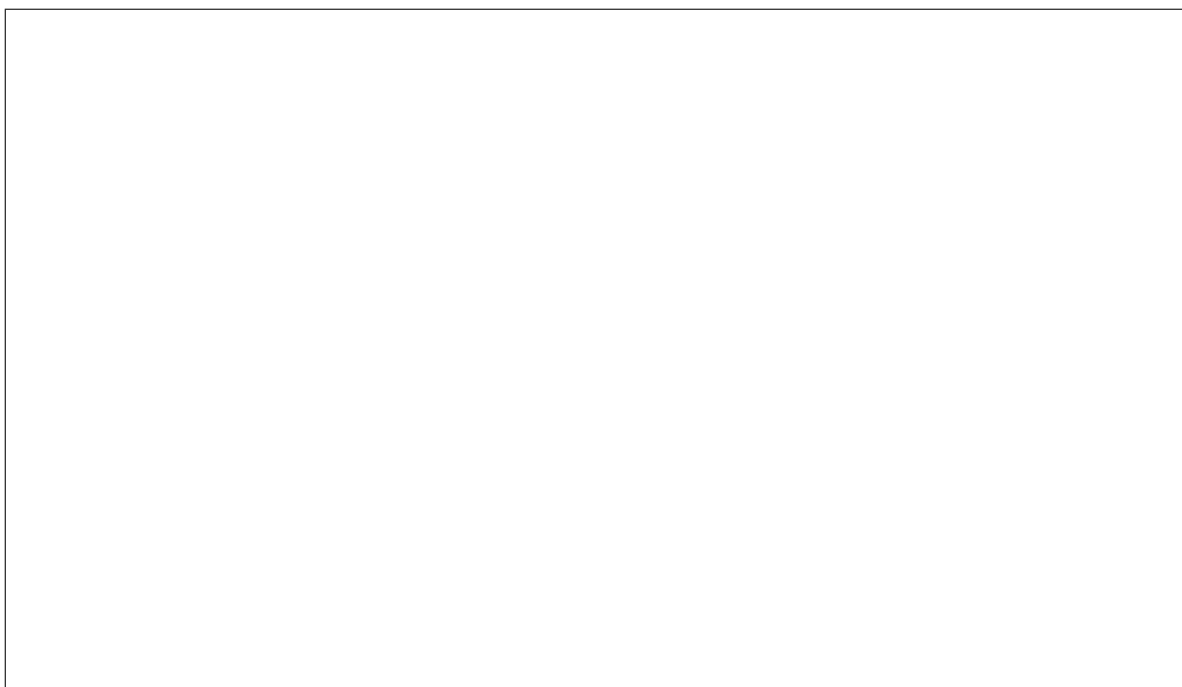
$$\hat{\mathbf{w}}_{\text{MAP}} = \arg \max_{\mathbf{w} \in \mathbb{R}^{d+1}} \sum_{i=1}^n \log p(y_i | \mathbf{x}_i, \mathbf{w}) + \log p(\mathbf{w}).$$

Note: The vector $\mathbf{w} \in \mathbb{R}^{d+1}$ includes the offset w_0 . The prior $p(\mathbf{w})$ is defined only over w_1, \dots, w_d .

Problem 2 [20 points]

- (a) Show that maximizing log-posterior with Gaussian noise model and Gaussian prior is equivalent to ridge regression problem, with regularization parameter $\mu\sigma^2$:

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg \min_{\mathbf{w} \in \mathbb{R}^{d+1}} \left\{ \sum_{i=1}^n (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \mu\sigma^2 \sum_{j=1}^d w_j^2 \right\}. \quad (7)$$



(b) Give a closed-form solution for $\hat{\mathbf{w}}_{\text{MAP}}$.

(c) In this part, we consider the logistic model, which is defined as

$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the logistic function. Now consider the Laplace distribution, $\mathcal{L}(\mu, b)$, whose probability density given by

$$f_{\mathcal{L}}(z | \mu, b) = \frac{1}{2b} \exp\left(-\frac{|z - \mu|}{b}\right).$$

Here μ is referred to as the location parameter and b is the scale parameter. We assume a Laplacian prior on $\mathbf{w} \in \mathbb{R}^{d+1}$, i.e., each component w_i of the vector \mathbf{w} is independently and identically distributed as $w_i \sim \mathcal{L}(0, b)$. Show that maximizing log-posterior with the logistic regression model and Laplacian prior leads to L1 regularization, i.e., maximizing the log-posterior is equivalent to solving the following problem:

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg \min_{\mathbf{w}} \left\{ - \sum_{i=1}^n \left(y_i \log(\sigma(\mathbf{w}^\top \mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^\top \mathbf{x}_i)) \right) + \lambda \|\mathbf{w}\|_1 \right\}.$$

What is the value of λ ?

III. Optimal Classification

We have seen in class that the minimal risk for a particular joint distribution $p(\mathbf{x}, y)$ under 0/1 loss

$$L_{0/1}(\hat{y}, y) = \begin{cases} 0, & \text{if } \hat{y} = y, \\ 1, & \text{if } \hat{y} \neq y \end{cases}$$

is attained by the Bayes classifier $h^*(\mathbf{x}) = \operatorname{argmax}_c p(c|\mathbf{x})$. One may suspect that this bound is limited to *deterministic* classifiers. An attempt to “beat” this bound, then, could be based on the following, *randomized* classifier. Define, for any data point \mathbf{x} , a probability distribution $q(c|\mathbf{x})$ over class labels c conditioned on the input \mathbf{x} . The resulting randomized classifier (for which q serves as a parameter), given a data point \mathbf{x} , draws a random class label from q :

$$h_r(\mathbf{x}; q) = c_r, \quad c_r \sim q(c|\mathbf{x})$$

To express the risk of this classifier we need to take the expectation over all possible outcomes of the random decision:

$$R(h_r; q) = \int_{\mathbf{x}} \sum_{c=1}^C \sum_{c'=1}^C L_{0/1}(c', c) q(c_r = c'|\mathbf{x}) p(\mathbf{x}, y = c) d\mathbf{x}$$

Problem 3 [15 points] Show that for any q ,

$$R(h_r; q) \geq R(h^*),$$

that is, the risk of the randomized classifier h_r defined above is at least as high as the Bayes risk.

Advice: As we saw in class, it is enough to show that the inequality holds for the conditional risk, i.e., that $R(h_r|\mathbf{x}) \geq R(h^*|\mathbf{x})$ for any \mathbf{x} .

IV. Softmax

In this section we will consider a discriminative model for a multi-class setup, in which the class labels take values in $\{1, \dots, C\}$. A principled generalization of the logistic regression model to this setup is the *softmax* model. It requires that we maintain a separate parameter vector for each class. The estimate for the posterior for class c , $c = 1, \dots, C$ is

$$\hat{p}(y = c | \mathbf{x}; \mathbf{W}) = \text{softmax}(\mathbf{w}_c \cdot \mathbf{x}) \triangleq \frac{\exp(\mathbf{w}_c \cdot \mathbf{x})}{\sum_{y=1}^C \exp(\mathbf{w}_y \cdot \mathbf{x})},$$

where \mathbf{W} is a $C \times d$ matrix, the c -th row of which is a vector \mathbf{w}_c associated with class c .

Problem 4 [15 points]

- (a) Show that the Softmax model corresponds to modeling the log-odds between any two classes $c_1, c_2 \in \{1, \dots, C\}$ by a linear function. In this problem, without loss of generality, you can assume that there is no bias ($\mathbf{b} = 0$), since it can be incorporated into \mathbf{W} using a constant column of ones in \mathbf{x} .

- (b) Furthermore, consider the binary case ($C = 2$). Show that in that case, for any two D -dimensional parameter vectors \mathbf{w}_1 and \mathbf{w}_2 in the Softmax model, there exists a single D -dimensional parameter vector \mathbf{v} such that

$$\frac{\exp(\mathbf{w}_1 \cdot \mathbf{x})}{\exp(\mathbf{w}_1 \cdot \mathbf{x}) + \exp(\mathbf{w}_2 \cdot \mathbf{x})} = \sigma(\mathbf{v} \cdot \mathbf{x}),$$

i.e., in the binary case the softmax model is equivalent to the logistic regression model.

For notational convenience, let $z_c = \mathbf{w}_c \cdot \mathbf{x}$ denote the score of the linear predictor for class c for a given input \mathbf{x} , for $c = 1, \dots, C$. Then,

$$p_c = \hat{p}(y = c | \mathbf{x}; \mathbf{W}) = \frac{\exp(z_c)}{\sum_{i=1}^C \exp(z_i)}, \text{ for } c = 1, \dots, C.$$

Problem 5 [15 points] Show that $\frac{\partial p_c}{\partial z_k}$ is positive if $k = c$, and negative otherwise. Argue that increasing the value of z_k must increase the corresponding probability value p_k while decreasing all other probability values p_i for any $i \neq k$.

In class, we discussed how the posteriors under the Softmax model are invariant to shifting scores and how we can leverage this property to deal with the overflow. Here we consider an alternative wherein we scale the scores. Let $s > 0$ be any positive value. Let

$$q_c(s) = \frac{\exp(s \cdot z_c)}{\sum_{i=1}^K \exp(s \cdot z_i)}, \text{ for } c = 1, 2, \dots, C.$$

Let $M = \max\{z_1, z_2, \dots, z_C\}$ and suppose that there are K entries among z_1, z_2, \dots, z_C whose value is equal to M .

Problem 6 [15 points] Show that the following holds as we scale s :

$$\lim_{s \rightarrow +\infty} q_j(s) = \begin{cases} 0 & \text{if } z_j < M \\ \frac{1}{K} & \text{if } z_j = M \end{cases}.$$

We now turn to a practical exercise in learning the softmax model, which can be done very similarly to learning logistic regression - via (stochastic) gradient descent. We will consider the L_2 regularization

$$(\mathbf{W}^*, \mathbf{b}^*) = \operatorname{argmax}_{\mathbf{W}, \mathbf{b}} \left\{ \frac{1}{N} \sum_{i=1}^N \log \hat{p}(y_i | \mathbf{x}_i; \mathbf{W}, \mathbf{b}) - \lambda \|\mathbf{W}\|^2 \right\},$$

where $\|\mathbf{W}\|$ is the Frobenius norm of the matrix \mathbf{W} - look it up if you are not familiar with this term.

Problem 7 [20 points] Write down the log-loss of the L_2 -regularized softmax model, and its gradients with respect to \mathbf{W} and \mathbf{b} (in the stochastic setting, i.e., computed over a single training example). Then, write the update equation(s) for the stochastic gradient descent, assuming learning rate η .

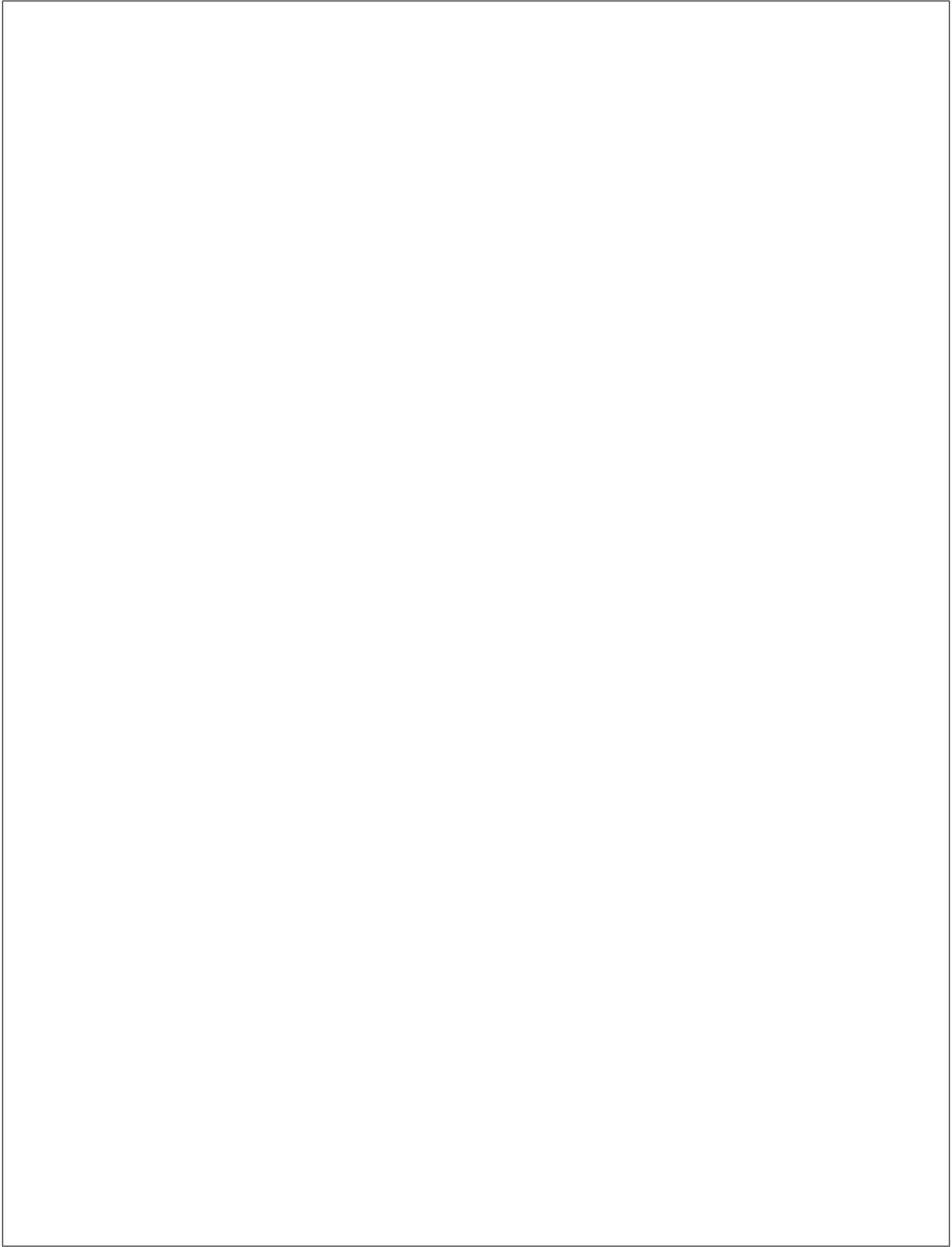
Advice: You may find it helpful, both in derivation and in coding, to convert the scalar representation of the labels $y \in \{1, \dots, C\}$ to a vector representation $\mathbf{t} \in \{0, 1\}^C$, in which if $y_i = c$ then $t_{ij} = 0$ for all $j \neq c$. This is sometimes called “one-hot” encoding of the labels: among C elements of the 0/1 label vector, exactly one element is “hot”, i.e., set to 1.

We can then collect all the parameters in the $d \times C$ matrix \mathbf{W} ; the c -th column is \mathbf{w}_c . Let $\hat{\mathbf{p}}_i$ be the vector of estimated log-posterior values on the i -th example, i.e., $\hat{p}_{i,c} = \log \hat{p}(y_i = c | \mathbf{x}_i; \mathbf{W}, \mathbf{b})$. Note that

$$\hat{p}_{i,c} = \mathbf{w}_c^T \mathbf{x} + b_c - \log \sum_{j=1}^C e^{\mathbf{w}_j^T \mathbf{x} + b_j}.$$

With the notation above we have that the log-likelihood on a single example is

$$\log p(y_i | \mathbf{x}_i; \mathbf{W}, \mathbf{b}) = \mathbf{t}_i^T \hat{\mathbf{p}}_i$$



We are now ready to apply the softmax model to the problem of classifying handwritten digits. We will work with the MNIST data set, which has served as a popular benchmark for classification methods over many years. Each example is a 28 by 28 pixel grayscale image; we will be working

with a vectorized representation of the images, converted to a 576-dimensional vector with values between 0 (black) and 1 (white). The data set has four partitions you will work with:

- Small training set of 400 examples;
- Large training set of 7000 examples;
- Validation set of 2000 examples;
- Test set of 1000 examples (no labels).

Each set is divided roughly equally among 10 classes for digits 0 through 9. There are two training sets so we could investigate the effect of data scarcity (or relative abundance) on training a linear model for this task.

Problem 8 [80 points] In this problem you will implement the gradient update procedure in the previous problem in order to classify images of handwritten digits. We have provided skeleton code in the Jupyter notebook that you will have to modify in order to get the best possible prediction results.

You will have to:

- Write the code to compute softmax predictions from model “scores” in `softmax`.
- Write the computation for the gradient with respect to **W** and **b** in `calcGrad`.
- Write the update rule using the gradients and a step size in `modelUpdate`
- Fill in a set of values for the regularization parameter `lambda` for which you will evaluate the performance of the model.

We have labeled parts of the skeleton code `YOUR CODE HERE`, where you will need to make changes.

We have provided some suggested values for the “hyper-parameters” of the learning algorithm: size of the mini-batch, stopping criteria (currently just limit on number of iterations), the settings for the initial learning rate and for decreasing its value over iterations (or not). These should be a reasonable starting point, but you are encouraged to experiment with their values, and to consider adding additional variations: changing the mechanism for selection of examples in the mini-batch (how should the data be sampled? should the mini-batch be constrained to be representative of all the classes?), additional stopping criteria, etc.

Feel free to guess appropriate values, or to tune them on the validation set. We have already provided code that evaluates the error rate on the training set and the validation set after the training has finished.

Your tuning procedure and any design choices, and the final set of values for all the hyper-parameters chosen for the final model, should be clearly documented in the notebook; please write any comments directly in the notebook rather than in the PDF file.

Please report the following statistics for your model in the write-up: the validation error and the confusion matrix. The confusion matrix in a classification experiment with C classes is

a $C \times C$ matrix \mathbf{M} in which M_{ij} is the number of times an example with true label i was classified as j .

In addition, visualize the parameters of the learned model. Since these are in the same domain as the input vectors, we can visualize them as images. Specifically, ignore the bias term, and use for instance `plt.imshow(W[:, i].reshape(24, 24))` to show the vector w_i associated with class i . Try to develop and write down an intuitive explanation of what the resulting images show.

Finally, compare and contrast the behavior of training, in particular the role of regularization, in the two data regimes (small vs large data set). Write your observations and conclusions in the notebook.

For the final evaluation, we have set up two Kaggle competitions to which you will be submitting your final predictions on a held-out testing set:

- MNIST small: <https://www.kaggle.com/t/47492cba7db440ea9e93bfd9e6efb5c9>
- MNIST large: <https://www.kaggle.com/t/cfeb17ffb868424fadedf024b49921f89>

First, you will have to create a Kaggle account (with your `jhu.edu` email). Once you have access to the competition pages (when you have an account follow the invite links above to gain access), download the data file by clicking Data on the left-side menu. The file named `mnist.h5` is listed here in both competitions. This file contains `small_train`, `large_train`, `val`, and `kaggle` (test) partitions, which can be accessed using the provided `load_data` function. The data loaded from these partitions is of dimension

$$N \times 576$$

with elements between 0 and 1 (where N is the number of examples in the partition), and the label matrix loaded from the training and validation partitions is of size

$$N \times 10$$

(since there are 10 digit classes).

Read through all three information pages carefully (Description, Evaluation and Rules) and accept the rules. You will now be ready to make submissions. The two competitions have the same goal and structure; one is for models trained on 400 examples (small) and the other for the models trained on 20 times as many examples (large). You should treat these two data sets separately when deciding your hyper-parameters.

Our code will automatically evaluate your model and produce a Kaggle submission file for you, e.g. `submission-small.csv`. Once you have accepted the rules, there will be an option to “Make a submission”, where you can upload this CSV file. The testing set that is evaluated for the Kaggle submission contains an additional 1,000 samples with unknown labels, to make sure you did not overfit the testing set. To ensure you do not overfit this held-out set, *we have limited your submissions to two per day, so start early and you will get more chances if you make mistakes. Your score will appear on a leaderboard that everyone can see.*