Homework 1
Naive Bayes and Logistic Regression[1]

CMU 10-315: Introduction to Machine Learning (Fall 2019)

https://piazza.com/class/jzqjbzyfzu32p2
OUT: September 4 2019

DUE: September 18 2019 11:59 pm.
TAs: Aliaa Essameldin, Fabricio Flores, Siddharth Ancha, Yue Wu

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section on the course site for more information: https://www.cs.cmu.edu/~aarti/Class/10315_Fall19/index.html
- **Submitting your work:**
  - **Gradescope:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using Gradescope (https://gradescope.com/). Please use the provided template. Submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Alternatively, submissions can be written in LaTeX. Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. Each derivation/proof should be completed on a separate page. For short answer questions you **should not** include your work in your solution. If you include your work in your solutions, your assignment may not be graded correctly by our AI assisted grader.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For LaTeXusers, use ■ (blacksquare) and ●(blackcircle) for shaded boxes and circles, and don't change anything else.

[1]Compiled on Wednesday 2nd October, 2019 at 22:47

Instructions for Specific Problem Types

For "Select One" questions, please fill in the appropriate bubble completely:

Select One: Who taught this course?

- ● Aarti Singh
- ○ Marie Curie
- ○ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

Select One: Who taught this course?

- ● Aarti Singh
- ○ Marie Curie
- ⊗ Noam Chomsky

For "Select all that apply" questions, please fill in all appropriate squares completely:

Select all that apply: Which are scientists?

- ■ Stephen Hawking
- ■ Albert Einstein
- ■ Isaac Newton
- □ I don't know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

Select all that apply: Which are scientists?

- ■ Stephen Hawking
- ■ Albert Einstein
- ■ Isaac Newton
- ⊠ I don't know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

Fill in the blank: What is the course number?

10-315	10-~~7~~315

1 Bayes Classifiers (55 points)

1.1 Optimal classifier (10 points)

In Lecture 2 we stated that the optimal classifier for binary classification takes the following form

$$f^*(x) = \underset{Y=y}{\operatorname{argmax}} \; P(Y = y|X = x), \tag{1}$$

here we are going to prove this.

Let be f a prediction rule, for binary classification, the loss is given by

$$loss(f(X), Y) = \mathbf{1}_{\{Y \neq f(X)\}} = \begin{cases} 1 & \text{if } f(X) \neq Y \\ 0 & \text{if } f(X) = Y \end{cases}$$

In order to prove (1), we first define the risk of a prediction rule f as:

$$R(f) = \mathbb{E}[loss(Y, f(X))],$$

and our objective is to find the function f^* that minimizes this risk among all possible functions. In other words

$$f^* = \underset{f}{\operatorname{argmin}}\, R = \underset{f}{\operatorname{argmin}}\, \mathbb{E}[loss(Y, f(X))]$$

1. (3 points) Let be $Y \in \{c_1, c_2\}$ the two class values that this random variable can take and $X \in \mathcal{X}$, being $\mathcal{X}$ the input space. Starting from $\mathbb{E}[loss(Y, f(X))]$ expand the expectation ($\mathbb{E}$) in terms of two indicator functions $\mathbf{1}_{\{f(x)=c_1\}}$ and $\mathbf{1}_{\{f(x)=c_2\}}$.

   Hint: The loss function can also be written as:

$$loss(Y, f(X)) = \begin{cases} 1 & \text{if } f(X) = c_1 \text{ and } Y = c_2 \\ 1 & \text{if } f(X) = c_2 \text{ and } Y = c_1 \\ 0 & \text{if } f(X) = Y \end{cases}$$

Solution

$$\begin{aligned} \mathbb{E}[loss(y, f(X))] &= \mathbb{E}[\mathbf{1}_{\{f(X)\neq Y\}}] \\ &= \mathbb{E}_X \mathbb{E}_{Y|X}\left[\mathbf{1}_{\{f(X)\neq Y\}}\right] = \mathbb{E}_X \mathbf{P}(\mathbf{1}_{\{f(X)\neq Y\}}) \\ &= \mathbb{E}_X \left[\mathbf{P}(y = c_2|x)\mathbf{1}_{f(x)=c1} + \mathbf{P}(y = c_1|x)\mathbf{1}_{f(x)=c2}\right] \end{aligned}$$

2. (3 points) Using the expression that you just derived, rewrite it in terms of a single indicator function $\mathbf{1}_{\{f(x)=c_1\}}$. What is the expression that you obtained? **Select one:**

○ $\mathbb{E}_X\Big[P(Y=c_1|X=x)+[P(Y=c_2|X=x)-P(Y=c_1|X=x)]\cdot\mathbf{1}_{\{f(x)=c_1\}}\Big]$

○ $\mathbb{E}_X\Big[P(Y=c_1|X=x)+[P(Y=c_2|X=x)+P(Y=c_1|X=x)]\cdot\mathbf{1}_{\{f(x)=c_1\}}\Big]$

○ $\mathbb{E}_X\Big[P(Y=c_1|X=x)+[P(Y=c_2|X=x)-P(Y=c_2|X=x)]\cdot\mathbf{1}_{\{f(x)=c_1\}}\Big]$

○ $\mathbb{E}_X\Big[[P(Y=c_2|X=x)-P(Y=c_1|X=x)]\cdot\mathbf{1}_{\{f(x)=c_1\}}\Big]$

Solution 1.

3. (4 points) Finally, you can take the argmin of the expression that you just selected (i.e $\mathbb{E}[loss(Y,f(X))]$). Write down the expression that you obtained for $f^* = \underset{f}{\text{argmin}}\,\mathbb{E}[loss(Y,f(X))]$ and re-write it in such a way that you can obtain (1).

Solution

$$f^* = \underset{f}{\text{argmin}}\,\mathbb{E}_X\Big[P(Y=c_1|X=x)+[P(Y=c_2|X=x)-P(Y=c_1|X=x)]\cdot\mathbf{1}_{\{f(x)=c_1\}}\Big]$$
$$= \underset{f}{\text{argmin}}\,\mathbb{E}_X\Big[[P(Y=c_2|X=x)-P(Y=c_1|X=x)]\cdot\mathbf{1}_{\{f(x)=c_1\}}\Big]$$
$$= \begin{cases} c_1 & \text{if } P(y=c_1|x) > P(y=c_2|x) \\ c_2 & \text{otherwise} \end{cases}$$
$$= \underset{Y=y}{\text{argmax}}\,P(Y=y|X=x)$$

1.2 MLE vs. MAP estimation of probabilities (7 points)

Probability estimation is a fundamental problem when using the Bayes rule for classification as in (1). Let's assume that the following dataset of observations of a Bernoulli process (H/T outcomes) is given: $\mathcal{D} = \{H, H, H, H, H\}$. In addition to the data, we also have a *prior* about the probability of observing H or T as an outcome. The prior is quantified in terms of *pseudo-observations* for H and T, where we have $\alpha = 3$ pseudo-observations for the outcome H, and $\beta = 2$ pseudo-observations for the outcome T (note: a pseudo-observation is not a real observation, but rather a sort of "imaginary" observation that reflects our beliefs).

Since we know that data in $\mathcal{D}$ is drawn from a Bernoulli distribution, we can safely assume that each $x_i \in \mathcal{D}$ is $x_i \sim Bernoulli(x|\theta)$. Our problem is therefore the estimation of the parameter θ (where θ is the probability of H, why?).

(2 points)**Select one:** The MLE estimator for θ is

○ 0.5

○ 0

○ 1

○ 3/2

Solution 1

$P(\mathcal{D} \mid \theta) = \theta^H(1-\theta)^T$.
MLE: $\theta = \frac{H}{H+T}$.

(3 points)**Select one:** Using a $Beta(\theta|\alpha, \beta)$ distribution to model the prior over the parameter θ, the MAP estimate for θ is

○ 3/2

◯ 7/8

◯ 1

○ 6/8

Solution 7/8

Prior: $\theta \sim \text{Beta}(\theta \mid \alpha, \beta)$. Then $P(\theta \mid \alpha, \beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha,\beta)}$.
Likelihood: $P(\mathcal{D} \mid \theta) = \theta^H(1-\theta)^T$.
Posterior:

$$\begin{aligned} P(\theta \mid \mathcal{D}) &= P(\theta \mid \alpha, \beta) \cdot P(\mathcal{D} \mid \theta)/P(\mathcal{D} \mid \alpha, \beta) \\ &\propto P(\theta \mid \alpha, \beta) \cdot P(\mathcal{D} \mid \theta) \\ &= \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha,\beta)} \cdot \theta^H(1-\theta)^T \\ &\propto \theta^{\alpha+H-1}(1-\theta)^{\beta+T-1} \end{aligned}$$

MAP: $\theta = \frac{\alpha+H-1}{\alpha+H+\beta+T-2}$.

(2 points) **Fill in the blank:** Using a $Beta(\theta|1, 1)$ distribution to model the prior over the parameter θ, the MAP estimate for θ is:

Solution The same as MLE.
$P(\theta \mid \alpha = 1, \beta = 1) = U(0, 1)$.
Uniform prior results in posterior equal to likelihood.

i	n_i	Possible Values
00	2	b, a
01	-	continuous
02	-	continuous
03	4	u, y, l, t
04	3	g, p, gg
05	14	c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff
06	9	v, h, bb, j, n, z, dd, ff, o
07	-	continuous
08	2	t, f
09	2	t, f
10	-	continuous
11	2	t, f
12	3	g, p, s
13	-	continuous
14	-	continuous
label	2	"+"/"-"

Table 1: Description of attribute information in provided data files

1.3 Implementing Naïve Bayes (38 points)

In this question you will implement a Naïve Bayes classifier for a simple credit-screening problem. In the provided input files you will find a processed collection of old applications for credit cards along with a corresponding label stating whether they were accepted or rejected by the bank. The goal in this problem is to learn the underlying function of these decisions and be able to classify (label) any new application as *accepted* or *rejected*.

In the data files, each line (credit card application data) consists of 16 data points separated by a comma ",". The first 15 data points represent attribute values and the last point is the corresponding label for that instance data point. Table 1 gives more information about each attribute in terms of the type and values it can take. In each of the data files, all attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. The features are a mix of multi-nominal and continuous attributes.

Each credit card application in your data can be therefore modeled as a feature vector $X = \{x_0, x_1, x_2, x_3, \ldots, x_{14}\}$ that includes different types of values. The general assumption is that the features are conditionally independent given the application label. We can therefore employ a Naïve Bayes classifier.

Note that, using the Bayes rule, your class prediction is:

$$\hat{y}_X = \underset{y}{\operatorname{argmax}}\, P(X|y) \cdot P(y), \tag{2}$$

where the class y is the screening result, "+" (accepted) or "-" (rejected). Since we are using

the Naïve Bayes model, the following holds:

$$P(X|y) = \prod_{x_i \in X} P(x_i|y). \tag{3}$$

Therefore, the problem becomes the estimation of the coordinate-wise conditional probabilities $P(x_i|y)$, for $i \in \{0, 2, 3, \ldots, 14\}$ as well as estimating the class prior distributions, and combining them with the Bayes rule to find the most probable class for given input data. Since the feature vector X includes hybrid data, you'll need to adopt different approaches for estimating the different probability distributions. To simplify your task and guide your work, we broke down the problem into separate tasks described below, distinguishing between continuous- and discrete-valued features, considering training, classification, and testing. Each task includes some questions that will also help you to make decisions in your implementation. In practice, you will have to make a few design choices and justify them. For obtaining full score in this section, you should turn in your final code (upload your code on Gradescope), as well as answer all the questions below (upload your handout on Gradescope). It is important that the code you submit and your answers below are consistent with each other!

Tasks *Starter Code for parsing data and evaluating the classifier will be made available soon through Piazza!*

1. **Pre-processing**

   You should start by reading the training data file `training.dat` and parsing it line-by-line to collect data about each feature separately. As mentioned earlier, data points are separated by a comma ",". For each of the 15 features you want to construct a `feature_values` structure, which is simply a tuple `(n, data)`:

   - `n`, is an integer representing the range of values for the given feature. You can conveniently set `n` to `-1` to indicate that the feature is a real number (encoded as a float).
   - `data` is an array of all values of the feature as per the data set. For continuous features this is an array of floats, and for discrete features with n possible values this is an array of integers $\in 0, 1, \ldots n-1$ that map to the possible values of the feature. It is up to you how to represent missing data (associated to a '?' symbol in the data file) in either case.

   The last, 16th attribute in each data entry is the class label. You should collect the labels separately into an array `labels` that you will use to estimate the probability distributions for all attributes.

   (2 points) How should your classifier deal with a missing entry in a feature vector X while reading the data? (Note that this is not an open question) **Select One:**

   ○ It can discard the entry corresponding to '?' as if the data for that specific feature didn't exist.

○ It should treat '?' as an extra value and include it in the data used for probability distribution estimation.

○ It has to discard an equal number of data points from all features.

○ It has to discard the entire data entry X.

Solution 1 because the probabilities are conditionally independent, so omitting an attribute will not affect other calculations. Some implementations could do "4", but in our case it'd be wasteful.

2. **Estimating Probability Distributions for Continuous-Valued Features**

For continuous-valued features, write a function `estimate_continuous(feature_values, labels)` that takes as inputs a feature structure `feature_values` (described above) which contains all training data for that attribute, and `labels`, the array of corresponding labels.
`estimate_continuous(feature_values, labels)` should return a pair of tuples, ($P_{i|acc}$, $P_{i|rej}$), where, for feature i, $P_{i|acc}$ provides the estimated values of the parameters for the probabilistic distribution of class *accepted*, and $P_{i|acc}$ provides the estimated parameter values for the distribution of class *rejected*. Parameter estimation for these continuous distributions must be done using **Maximum Likelihood Estimation**.

(2 points) We do parametric estimation of probability distributions. This makes the task simple(r), but we need to set an inductive bias by making a hypothesis about the class of the distribution (e.g., Gaussian, Bernoulli, Binomial). In absence of additional knowledge about the data, what could be a suitable choice for the class of the distribution that you should assume in `estimate_continuous(feature_values, labels)`? How many parameters will you have to estimate for each one of such distributions?

Solution Univariate Gaussian distribution because the values are continuous and we consider individual attributes, a univariate Gaussian distribution is defined by two parameters.

(8 points) What are the parameters that your function `estimate_continuous(feature_values, labels)` returns when ran on the attribute with identifier ($i = 10$) in the table?

$P_{10|acc}$

$P_{10|rej}$

3. **Estimating Probability Distributions for Discrete-Valued Features**

For discrete-valued features, write a function `estimate_discrete(feature_values, labels)` that takes as inputs a feature structure (described above) `feature_values` which contains all training data for that feature, and `labels`, the array of corresponding labels. `estimate_discrete(feature_values, labels)` should return a pair of tuples, $(P_{i|acc}, P_{i|rej})$, where, for feature i, $P_{i|acc}$ provides the estimated values of the parameters for the probabilistic distribution of class *accepted*, and $P_{i|acc}$ provides the estimated parameter values for the distribution of class *rejected*.

(2 points) What is the most suitable functional form that you should assume for the parametric probability distribution in `estimate_discrete(feature_values, labels)`? Note that the choice should be suitable to represent discrete random variables that can take on n possible values, $n \geq 2$. Given your choice, how many parameters will you have to estimate for each distribution?

Solution Should assume Multinomial distribution since we do not know the number of values and are not sure if they represent multiple trials. However, Multinoulli is an acceptable answer as well. Either case, parameters will be $2n$.

Again, you must use **Maximum Likelihood Estimation** to estimate the parameters of the parametric probability distributions. However, this time, your probability estimation will be presented differently, each probability distribution is parametrized into a tuple of size n such that $P_{i|y}[j] = P(x_i = j|y)$ where j is the j-th possible value of the feature.

(4 points) Run your function `estimate_discrete(feature_values, labels)` on the attribute with $(i = 0)$ in Table 1. What is the log-probability that $x_0 = 1$ ('a', in the specific case)? [2]

$\log P_{x_0|acc}[1]$

$\log P_{x_0|rej}[1]$

(4 points) Run your function `estimate_discrete(feature_values, labels)` on the attribute with $(i = 5)$ in Table 1. What is the log-probability that $x_5 = 8$ ('q')?

[2] **Note on log-probability vs probability**: Whenever implementing probability distributions in code, it is always advisable to work with log-probabilities instead of probabilities. Raw probabilities can be very small, especially if many small probabilities are multiplied together. This can cause numerical issues. Instead, we should represent log probabilities and add them whenever raw probabilities need to be multiplied.

$\log P_{x_5|acc}[8]$

$\log P_{x_5|rej}[8]$

4. **Estimating Class Distribution**

   For the case of the class prior probability distribution, write a function `probability_acc(labels)` that takes in input the array of labels from the data set and uses it to estimate the prior probability that *any* credit card application would be *accepted*. In this case, you must use a MAP estimate for the parameter of the distribution. At this aim, you are provided with a *Beta* prior for the parameters, where the hyper-parameters of the *Beta* distribution are $\alpha = 7$ and $\beta = 9$.

   (2 points) Given that you are asked to use a MAP estimate, and a *Beta* prior is given for the parameters, what is your choice for the parametric class distribution? (e.g., Gaussian, Binomial, Bernoulli). Justify your answer. How many parameters does it take?

   Solution Bernoulli, it takes one parameter

   (4 points) Run your function `probability_acc(labels)` on the provided training data. What is the estimated log-probability that a credit card application is *accepted*?

   Solution

5. **Classify new instances**

   Write a function `estimate(trained_model,X)` that takes in input:

   - `trained_model`: a tuple of 16 elements, where first 15 are the attribute probability estimations computed using `estimate_discrete` and `estimate_continuous`, and the last one is the value returned from `probability_acc`.
   - `X`: an unlabeled feature vector represented as an array of 15 attribute values. Continuous attributes are represented as floats and discrete attributes are represented as integers $\in 0, 1, \ldots n-1$ that map to the possible values of the attribute.

   `estimate(trained_model,X)` should return a tuple of three values `class, log_prob_acc, log_prob_rej`:

- `class`: a string representing the chosen class "+" if the classifier accepts the application with the provided attributes and "-" otherwise.
- `log_prob_acc`: the conditional log-probability of the feature vector `X` given that the application is *accepted*, $P(X|y = acc)$.
- `log_prob_rej`: the conditional log-probability of the feature vector `X` given that the application is *rejected*, $P(X|y = rej)$.

(1 points) Run your functions on the following values of `X` (after mapping them to the correct representation) and report the classification and the log-probabilities returned:

(a) $[b, 28.25, 0.875, u, g, m, v, 0.96, t, t, 03, t, g, 396, 0]$

Solution

(b) $[b, 42.75, 4.085, u, g, aa, v, 0.04, f, f, 0, f, g, 108, 100]$

Solution

(c) $[a, 46.08, 3, u, g, c, v, 2.375, t, t, 8, t, g, 396, 4159]$

Solution

6. **Evaluate your Classifier**

Last part in building any machine learning model is evaluating it. You are provided with an incomplete function `ClassificationEvaluation(Filename)` that you should complete using the functions you implemented above. You will then run `ClassificationEvaluation()` using the training dataset in `training.dat` and testing dataset in `testing.dat`. Report the errors below (2 points).

Training Error **Solution**

Testing Error **Solution**

(3 points) Which Error is more representative of the error we would expect on a new collection of applications? Does Naive Bayes attempt to minimize the training error? **Select One:**

- ○ Training, Yes
- ○ Training, No
- ○ Testing, Yes
- ○ Testing, No

Solution C.

2 Logistic Regression (45 points)

2.1 Softmax regression (35 points)

In this question, we will derive the "multi-class logistic regression" algorithm (the MLE and its gradient). We assume the dataset $\mathcal{D}$ is d-dimensional (has d features) with n entries.

Given a training set $\{(x^i, y^i)|i = 1, \ldots, n\}$ where $x^i \in \mathbb{R}^{d+1}$ is a feature vector and $y^i \in \mathbb{R}^k$ is a binary (one-hot) vector with k entries (classes). Note that in a one-hot vector, the corresponding class label is 1 and all the rest entries are 0s. For example, if the label of X is 3, then the corresponding y should look something like $[0, 0, 1, ..., 0] \in \mathbb{R}^k$

Note that x^i is an vector of length $(d + 1)$ because we pad the d features by 1 to vectorize computing the bias, that is $x^i = [1, (x^i)']$, where $(x^i)' \in \mathcal{D}$.

We want to find the parameters $\hat{w} \in \mathbb{R}^{k\times(d+1)}$ (one weight vector for each class) that maximize the likelihood for the training set, assuming a parametric model of the form

$$p(y_c^i = 1|x^i; w) = \frac{\exp(w_c^T x^i)}{\sum_{c'} \exp(w_{c'}^T x^i)}. \tag{4}$$

Note that $\frac{\exp(w_c^T x^i)}{\sum_{c'} \exp(w_{c'}^T x^i)}$ is always between 0 and 1, and $\sum_c p(y_c^i = 1|x^i; w)$ is always 1, which are desired properties of a probability distribution. Therefore, $\frac{\exp(w_c^T x^i)}{\sum_{c'} \exp(w_{c'}^T x^i)}$ is also known as the softmax function.

Since we know the probability sums to 1, we don't care about predicting the probability of the last (k^{th}) class, since we can calculate $p(y_k^i = 1|x^i; w)$ by:

$$p(y_k^i = 1|x^i; w) = 1 - \frac{\sum_{c'=1}^{k-1} \exp(w_{c'}^T x^i)}{\sum_{c'} \exp(w_{c'}^T x^i)}. \tag{5}$$

1. (10 points) Show the equivalence between the two equations (6) and (7). Provide a short justification for why each line follows from the previous one in your derivation. (This is how we store less weights by making use of the fact that the probabilities sum to 1)

$$p(y_c^i = 1|x^i; w) = \frac{\exp(w_c^T x^i)}{\sum_{c'} \exp(w_{c'}^T x^i)}. \tag{6}$$

$$= \begin{cases} \frac{\exp(w_c^T x^i)}{1+\sum_{c'=1}^{k-1} \exp(w_{c'}^T x^i)}, & \text{if } c < k \\ \frac{1}{1+\sum_{c'=1}^{k-1} \exp(w_{c'}^T x^i)}, & \text{if } c = k \end{cases} \tag{7}$$

Solution Note that adding the same value to all the weights does not affect the value of the equation. Therefore, we shift the k^{th} weight to 0.

$$\begin{aligned} p(y_c^i = 1|x^i; w) &= \frac{\exp(w_c^T x^i)}{\sum_{c'} \exp(w_{c'}^T x^i)} \\ &= \frac{\exp((w_{c'} - w_k)^T x^i)}{\sum_{c'} \exp((w_{c'} - w_k)^T x^i)}. \\ &= \frac{\exp((w_{c'} - w_k)^T x^i)}{1 + \sum_{c'=1}^{k-1} \exp((w_{c'} - w_k)^T x^i)}. \\ &= \begin{cases} \frac{\exp(w_c^T x^i)}{1+\sum_{c'=1}^{k-1} \exp(w_{c'}^T x^i)}, & \text{if } c < k \\ \frac{1}{1+\sum_{c'=1}^{k-1} \exp(w_{c'}^T x^i)}, & \text{if } c = k \end{cases} \end{aligned}$$

This shows that one can store only $k-1$ weight vectors by subtracting the kth vector from all other ones ((6) implies (7)). Equivalently, in logistic regression, it is acceptable to set one of the weight vectors to 0, and so you get 1 after exponentiating it ((7) implies (6)).

2. (5 points) Derive the conditional log likelihood for softmax regression. For the sake of simplicity, we only consider eq. (6) as $p(y_c^j \mid x^j, w)$.

$$\ell(w) \equiv \ln \prod_{j=1}^{n} p(y_c^j \mid x^j, w) \quad [\text{here } c \text{ is the true class of } x^j] \tag{8}$$

$$= \sum_{j=1}^{n} \sum_{c=1}^{k} \left[y_c^j \left(w_c^T x^j \right) - y_c^j \ln \left(\sum_{c'} \exp(w_{c'}^T x^j) \right) \right]. \tag{9}$$

Solution

$$\begin{aligned} \ell(w) &\equiv \ln \prod_{j=1}^{n} p(y_c^j \mid x^j, w) \quad [\text{here } c \text{ is the true class of } x^j] \\ &= \sum_{j=1}^{n} \sum_{c=1}^{k} y_c^i \ln p(y_c^j \mid x^j, w) \\ &= \sum_{j=1}^{n} \sum_{c=1}^{k} \left[y_c^j \ln \frac{\exp(w_c^T x^j)}{\sum_{c'} \exp(w_{c'}^T x^j)} \right] \\ &= \sum_{j=1}^{n} \sum_{c=1}^{k} \left[y_c^j \left(w_c^T x^j \right) - y_c^j \ln \left(\sum_{c'} \exp(w_{c'}^T x^j) \right) \right]. \end{aligned}$$

3. (5 points) Next, we will derive the gradient of the previous expression with respect to the c^{th} class of the weight matrix w_c, i.e., $\frac{\partial \ell(w)}{\partial w_c}$, where $\ell(w)$ denotes the log likelihood

from (9). We will perform a few steps of the derivation, and then ask you to do one step at the end. If we take the derivative of Expression 9 with respect to w_c, we get the following expression:

$$\nabla_{w_c} \ell(w) = \nabla_{w_c} \sum_{j=1}^{n} \sum_{c=1}^{k} \left[y_c^j \left(w_c^T x^j \right) - y_c^j \ln \left(\sum_{c'} \exp(w_{c'}^T x^j) \right) \right] \tag{10}$$

The blue expression is linear in w_c, so it can be simplified to $\sum_{j=1}^{n} y_c^j x^j$. For the red expression, first we consider a fixed $j \in [1, n]$. Use chain rule to verify that

$$\nabla_{w_c} \sum_{c=1}^{k} y_c^j \ln \left(\sum_{c'} \exp(w_{c'}^T x^j) \right) \tag{11}$$

$$= \frac{\exp(w_c^T x^j)}{\sum_{c'} \exp(w_{c'}^T x^j)} x^j \tag{12}$$

Solution First note that the c in ∇_{w_c} is not the same c in $\sum_{c=1}^{k} y_c^j \ln \left(\sum_{c'} \exp(w_{c'}^T x^j) \right)$. The later c iterates over 1 to k, and so the latter term is independent of c.

$$\begin{aligned}
\nabla_{w_c} \sum_{c=1}^{k} y_c^j \ln \left(\sum_{c'} \exp(w_{c'}^T x^j) \right) &= \nabla_{w_c} \left(\sum_{c=1}^{k} y_c^j \right) \ln \left(\sum_{c'} \exp(w_{c'}^T x^j) \right) \\
&= \nabla_{w_c} \ln \left(\sum_{c'} \exp(w_{c'}^T x^j) \right) \\
&= \frac{\exp(w_c^T x^j)}{\sum_{c'} \exp(w_{c'}^T x^j)} x^j \\
&= p(y_c^j = 1 \mid x^j; w) \cdot x_j
\end{aligned}$$

4. (5 points) Now use Equation 12 (and the previous discussion) to show that overall, Expression 10, i.e., $\nabla_{w_c} \ell(w)$, is equal to

$$\nabla_{w_c} \ell(w) = \sum_{j=1}^{n} x^j (y_c^j - p(y_c^j = 1 \mid x^j; w)) \tag{13}$$

Solution
Note that equation 12 equals $p(y_c^j = 1 \mid x^j; w) \cdot x_j$.
So, $\sum_{j=1}^{n} \frac{\exp(w_c^T x^j)}{\sum_{c'} \exp(w_{c'}^T x^j)} x^j = \sum_{j=1}^{n} p(y_c^j = 1 \mid x^j; w) \cdot x_j$.
Plugging into 10, we get

$$\nabla_{w_c} \ell(w) = \sum_{j=1}^{n} x^j (y_c^j - p(y_c^j = 1 \mid x^j; w))$$

5. (5 points) Since the log likelihood is concave, it is easy to optimize using gradient ascent. Derive the update rule for gradient ascent with respect to learning rate for w_c w.r.t. η, y^j, x^j, and $p(y^j = 1 \mid x^j; w^{(t)})$. Feel free to index into the vectors using subscripts.

Solution

$$w_c^{t+1} = w_c^t + \eta \sum_{j=1}^{n} x^j (y_c^j - p(y_c^j = 1 \mid x^j; w))$$

Note that c indexes the class i.e. $c \in \{1, ..., k\}$.
You should use only one of c or i but not both.

6. (5 points) Explain how the logistic regression you learned in class relate to the softmax regression you derived.

Solution Set $k = 2$ in logistic regression, and use the weight subtraction technique in equation 7.

$$p(y_c^i = 1 | x^i; w) = \frac{\exp(w^T x^i)}{\sum_{c'} \exp(w^T x^i)}.$$
$$= \begin{cases} \frac{\exp(w^T x^i)}{1+\exp(w^T x^i)}, & \text{if } c = 1 \\ \frac{1}{1+\exp(w^T x^i)}, & \text{if } c = 0 \end{cases}$$

2.2 General questions about logistic regression (10 points)

1. Explain why logistic regression is a discriminative classifier (as opposed to a generative classifier such as Naive Bayes).

   Solution Logistic regression directly learns a model for $P(Y|X)$, as opposed to learning models for $P(X)P(X|Y)$. It does not learn a model of the data $P(X)$. Therefore, logistic regression is a discriminative classifier.

2. What does the decision boundary of logistic regression look like when we have quadratic features? Justify your answer.

   Recall the prediction rule for logistic regression is if $p(y^j = 1 \mid x^j) > p(y^j = 0 \mid x^j)$, then predict 1, otherwise predict 0.

   (hint: consider the decision boundary as a function of w_0, w_1, w_2, w_3, w_4 and $x_1, x_2, x_1x_2, x_1^2, x_2^2$).

   Solution It is a Paraboloid. It either looks like a cone or a saddle. OR it can be a hyper-plane.

Collaboration Questions Please answer the following:

1. Did you receive any help whatsoever from anyone in solving this assignment?
   Yes / No.
   - If you answered 'yes', give full details: _____
   - (e.g. "Jane Doe explained to me what is asked in Question 3.4")
2. Did you give any help whatsoever to anyone in solving this assignment?
   Yes / No.
   - If you answered 'yes', give full details: _____
   - (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")
3. Did you find or come across code that implements any part of this assignment ?
   Yes / No. (See below policy on "found code")
   - If you answered 'yes', give full details: _____
   - (book & page, URL & location within the page, etc.).