School of Computing and Information Systems
The University of Melbourne
COMP90049 Introduction to Machin Learning
(Semester 1, 2021)

Week 6: Sample Solution

1. What is **gradient descent**? Why is it important?

   Gradient descent is an instance of iterative optimization algorithms: it finds the parameters corresponding to optimal points of a target function step-by-step, by starting out with some initial parameter value, and incrementally modifying this value in the `most promising way', i.e., in the way that leads to the largest improvement of the target function. This step-by-step procedure is important for optimization problems with no closed-form solution. This has numerous applications - most intuitively, in determining the regression weights which minimise an error function over some training data set.

2. What is **Logistic Regression**? What is "logistic"? What are we "regressing"?

   We build a Binary Logistic Regression model, where the target (y) is (close to) 1 for instances of the positive class, and (close to) 0 for instance of the negative class. (Suitably can be re-interpreted for multi-class problems.)

   The goal of binary logistic regression is to train a classifier that can make a binary decision about the class of an input observation. Consider a test instance X, which we will represent by a vector of features $[x_1, x_2,\ldots,x_n]$. The output y can be 1 or 0 (i.e., winning / not winning).

   The model calculates the probability P(y=1|x) (from which we can trivially derive P(y=1|x). A logistic regression classifier additionally defines a 'decision boundary', which is typically set at 0.5. If the model predicts the probability P(Y=1|x) > 0.5, we classify x as class 1. Otherwise, x is classified as class 0.

   In statistics, regression describes how an independent variable is numerically related to the dependent variable. In Logistic Regression we find the regression output z ($z = \vec{\theta}.\vec{X} = \theta_0 + \theta_1 x_1 + \cdots + \theta_F x_F$), where θ is the weight of the features. θ represents the numeric correlation between each feature (attribute) with the label (class), where the feature (attributes) of our dataset is assumed as our independent variables and the label (class) is our dependent variable.

   Since Logistic (or sigmoid) function has an easy–to–calculate derivative (that makes it easy to calculate the optimum parameters), and has a range of [0, 1], we apply the logistic function $\sigma = \frac{1}{1+e^{-z}}$ to the regression output z and call it Logistic Regression.

3. Bob tries to gather information about this year's apple harvest and ran a search in his favourite online news outlet. He retrieved a number of articles but found that a large portion of the retrieved articles are about the Apple laptops and computers -- and hence irrelevant to his search. He wants to build a logistic regression classifier, which uses the counts of selected words in the news articles to predict the class of the news article (fruit vs. computer). He built the following data set of 5 training instances and 1 test instance. Develop a logistic regression classifier to predict label $\hat{y} = 1$ (fruit) and $\hat{y} = 0$ (computer).

   For the moment, we assume that we already have an estimate of the model parameters, i.e., the weights of the 4 features (and the bias θ_0) is $\widehat{\theta} = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4] = [0.2, 0.3, -2.2, 3.3, -0.2]$.

ID	*apple*	*ibm*	*lemon*	*sun*	CLASS
TRAINING INSTANCES					
A	1	0	1	5	1 FRUIT
B	1	0	1	2	1 FRUIT
C	2	0	0	1	1 FRUIT
D	2	2	0	0	0 COMPUTER
E	1	2	1	7	0 COMPUTER
TEST INSTANCES					
T	1	2	1	5	0 COMPUTER

(i). Explain the intuition behind the model parameters, and their meaning in relation to the features

In this dataset we want identify if a piece of writing is about `computer` or `fruit` (e.g. '*new apple iPhone is very expensive*' vs. '*an apple a day, keeps the doctor away*'). To do so, we are using 4 terms (`apple`, `ibm`, `lemon`, `sun`) and the count of their occurrences in a piece of writing. So, for example we know that `Doc A` includes `apple` once and sun five times.

The decision to include exactly these for terms (and no others) as attributes, and to define their values as word occurrence counts is the decision of the modeller, and the outcome of a process called **Feature Engineering**.

Based on the definition, we know that in Logistic Regression, we model P(y = 1|x_1, x_2, ..., x_F) directly as subject to parameter θ. Using the following:

$$P(y = 1|x_1, x_2, \dots, x_F) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \cdots + \theta_F x_F)}} = \sigma(\theta_0 + \theta_1 x_1 + \cdots + \theta_F x_F)$$

Here, we have a binary output label *y = {0 (`computer`), 1 (`fruit`)}*, and four features: `apple`, `ibm`, `lemon`, `sun`.

Weights $\theta_1, \theta_2, \theta_3, \theta_4$ denote the respective importance of the features 1 to 4 for predicting the class `fruit` (1). For example, θ_2 (-2.2) indicates how important feature 2 (`ibm`) is for predicting $\hat{y}$ = 1 (fruit). θ_0 represent the bias for this model.

Here, the negative sign indicates that occurrence of the word `ibm` is *negatively correlated* with the class `FRUIT`. If we observe `ibm` in a document, it makes the label `FRUIT` *less likely*. The positive value of θ_3 indicates a *positive correlation* of feature `lemon` with `FRUIT`. If the word `lemon` is mentioned in a document, it *increases the probability* of its label being `FRUIT`.

(ii). Predict the test label.

Using the logistic regression formula above, we find

$$P(fruit|T) = P(\hat{y} = 1|T) = \sigma(\theta_0 + \theta_1 t_1 + \cdots + \theta_4 t_4)$$

$$= \sigma(0.2 + 0.3 \times 1 + (-2.2) \times 2 + 3.3 \times 1 + (-0.2) \times 5)$$

$$= \sigma(-1.6) = \frac{1}{1 + e^{-(-1.6)}} = 0.17$$

And consequently

$$P(computer|T) = 1 - P(fruit|T) = 1 - 0.17 = 0.83$$

Recall that we turn the logistic regression model into a classifier by predicting label y = 1 whenever p(y=1|x, θ) > 0.5, and predict y = 0 otherwise.

Since the for the test instance T the probability p(y=1) (FRUIT) is smaller than 0.5, we predict label y=0 (COMPUTER) for T. Comparing with our "ground truth" that we have in our dataset, we can see that our prediction is correct. ☺

(iii). Recall the conditional likelihood objective

$$-\log \mathcal{L}(\theta) = -\sum_{i=1}^{n} y_i \log\big(\sigma(x_i;\theta)\big) + (1-y_i)\log\big(1-\sigma(x_i;\theta)\big)$$

We want to make sure that the Loss (the negative log likelihood) our model, is lower when its prediction the correct label for test instance T, than when it's predicting a wrong label.

Compute the negative log-likelihood of the test instance (1) assuming the true label as y = 1 (fruit), i.e., our classifier made a mistake; and (2) assuming that the true label y = 0 (computer), i.e., our classifier predicted correctly.

- Assuming *y = 1* and we predicted $\hat{y} = 0$:

$$-\log \mathcal{L}(\theta) = -\sum_{i=1}^{n} 1\log\big(\sigma(0.2 + 0.3\times 1 + (-2.2)\times 2 + 3.3\times 1 + (-0.2)\times 5)\big) + (1-1)\log\big(1-\sigma(0.2 + 0.3\times 1 + (-2.2)\times 2 + 3.3\times 1 + (-0.2)\times 5)\big)$$

$$= -\sum_{i=1}^{n} 1\log\big(\sigma(-1.6)\big) + (1-1)\log\big(1-\sigma(-1.6)\big)$$

$$= -\big[1\, log\big(\sigma(-1.6)\big) + 0\big] = -log\ \frac{1}{1+e^{-(-1.6)}} = -\log(0.17) = 1.77$$

- Assuming *y = 0* and we predicted $\hat{y} = 0$:

$$-\log \mathcal{L}(\theta) = -\sum_{i=1}^{n} 0\log\big(\sigma(-1.6)\big) + (1-0)\log\big(1-\sigma(-1.6)\big)$$

$$= -\big[0 + 1\log\big(1-\sigma(-1.6)\big)\big] = -log\ (1-0.17) = -\log(0.83) = 0.19$$

Reassuringly, the negative log likelihood (aka loss) under predicted label y = 0 (correct) is **lower** than the loss for label = 1 which is the incorrect prediction.

4. For the model created in question 4, compute a single gradient descent update for parameter θ_1 given the training instances given above. Recall that for each feature j, we compute its weight update as

$$\theta_j \leftarrow \theta_j - \eta \sum_i (\sigma(x_i;\theta_i) - y_i)x_{ij}$$

Summing over all training instances *i*. We will compute the update for θ_j assuming the current parameters as specified above, and a learning rate $\eta = 0.1$.

θ_1 is the model parameter (respective importance) of the features 1 (apple) in our model. Using Gradient Descent method over iterations, we want to find the best parameters of the model (the parameters that minimise the loss (error) of our model).

Step 1, we need to compute the $\sigma(x_i; \theta)$ for all our training instances.

$$\sigma(x_A; \theta) = \sigma(0.2 + (0.3 \times 1 + (-2.2) \times 0 + 3.3 \times 1 + (-0.2) \times 5)) = 0.94$$

$$\sigma(x_B; \theta) = \sigma(0.2 + (0.3 \times 1 + (-2.2) \times 0 + 3.3 \times 1 + (-0.2) \times 2)) = 0.97$$

$$\sigma(x_C; \theta) = \sigma(0.2 + (0.3 \times 2 + (-2.2) \times 0 + 3.3 \times 0 + (-0.2) \times 1)) = 0.65$$

$$\sigma(x_D; \theta) = \sigma(0.2 + (0.3 \times 2 + (-2.2) \times 2 + 3.3 \times 0 + (-0.2) \times 0)) = 0.03$$

$$\sigma(x_E; \theta) = \sigma(0.2 + (0.3 \times 1 + (-2.2) \times 2 + 3.3 \times 1 + (-0.2) \times 7)) = 0.12$$

Step 2, now we can compute the parameter update for θ_1. We are using the following formula:

$$\theta_1 = \theta_1 - \eta \sum_{i \in \{A,B,C,D,E\}} (\sigma(x_i; \theta) - y_i)\, x_{1i}$$

$$\theta_1 = 0.3 - 0.1 \sum_{i \in \{A,B,C,D,E\}} (\sigma(x_i; \theta) - y_i)\, x_{1i}$$

$$\theta_1 = 0.3 - 0.1\,[((\sigma(x_A; \theta) - y_A).x_{1A}) + ((\sigma(x_B; \theta) - y_B).x_{1B}) + ((\sigma(x_C; \theta) - y_C).x_{1C}) + ((\sigma(x_D; \theta) - y_D).x_{1D}) + ((\sigma(x_E; \theta) - y_E).x_{1E})]$$

$$= 0.3 - 0.1\,[((0.94 - 1) \times 1) + ((0.97 - 1) \times 1) + ((0.65 - 1) \times 2) + ((0.03 - 0) \times 2) + ((0.12 - 0) \times 1)]$$

$$= 0.3 - 0.1\big((-0.06) + (-0.03) + (-0.70) + 0.06 + 0.12\big) = 0.3 - 0.1\,(-0.61)$$
$$= 0.3 + 0.061 = 0.361$$

The new value for θ_1 is 3.061. Given the feedback from the current model mistakes (over on our training data), the *importance* of feature `Apple` has slightly increased. We can do the same thing for other parameters θ_2, θ_3 and θ_4. We can continue the iterations until we find the "optimum" parameters.

NOTE: In the full Gradient Descent algorithm, we first compute the sigma value for all parameters (step 1 above), and then update **all parameters at once** (step 2 above).

5. [OPTIONAL] What is the relation between "odds" and "probability"?

A probability is the **ratio of number of successes to the total possible outcomes**. For example in an experiment involving drawing a ball from a bag containing 8 balls (5 of them `red` and the rest `blue`), the probability of choosing a `red` ball is 5/8.

Odds on the other hand is the **ratio of the probability of success to the probability of failure**. Returning to our example, if we want to compute the odds of drawing a `red` ball, probability of *success* (draw `red`) over probability of *failure* (not draw `red`) is $\frac{\frac{5}{8}}{\frac{3}{8}} = \frac{5}{3} = 1.7$

Note that the odds consider two possible outcomes (success vs. failure) so that we can write the denominator *p(failure) = 1 - p(success)*.

$$Odds = \frac{P(x)}{1 - P(x)}$$

So in our example the odds of choosing the 'red' can also calculated as follow:

$$Odds('red') = \frac{P('red')}{1 - P('red')} = \frac{\frac{5}{8}}{1 - \frac{5}{8}} = \frac{\frac{5}{8}}{\frac{3}{8}} = \frac{5}{3} = 1.7$$

6. [OPTIONAL] (a) What is **Regression**? How is it similar to **Classification**, and how is it different?

Regression and classification are both categorized under the same umbrella of *supervised* machine learning. Both share the same concept of utilizing known datasets (labelled training datasets) to make predictions.

Our target attribute (class) is *nominal* in classification, but *numeric* (continuous) in Regression. Consequently, in regression we can't exhaustively assess the likelihood of each class.

(b) Come up with one typical classification task, and one typical regression task. Specify the range of valid values of y (results) and possible valid values for x (attributes).

Regression example: Predicting house prices. Possible attributes can be x= {location, size, age, interest rates, ...}. *y* is a real value (price) and strictly positive.

Classification example: Sentiment classification of star movie reviews. x= {set of words, author ID, review *lengths, ...}*. *y* can have five classes {Weak (1), Not bad (2), Good (3), Great (4), Masterpiece (5)}.