Machine Learning W8 Tutorial

COMP30027 | Sandy Luo

Overview

Logistic Regression

Concept, code

Gradient Descent

for logistic regression

Classifier Combination

Different approaches

Logistic Regression

Q1:

What is logistic regression? How is it "logistic" and what are we "regressing"?

Logistic Regression

- Goal: **Classify** an input observation (binary classification)
  - Consider a test instance X with a vector of features [x_1,x_2, ...,x_n].
  - Output $y \in \{0, 1\}$
- Essentially:
  a. Use linear regression to predict the probability (log odds) of P(y=1|x)
  b. Transform linear regression w/ logistic regression (range in [0,1])
  c. Define a decision boundary, generally = 0.5
  d. If P(y=1|x) > 0.5, classify x as class 1. Otherwise, classify as class 0.

Q1:

What is logistic regression? How is it "logistic" and what are we "regressing"?

*The term **logistic** refers to the logistic (sigmoid) function used in the model, which is defined as*

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

where z is a linear regression of the features X and their corresponding weights β

$$z = \vec{\beta} \cdot \vec{X} = \beta_0 + \beta_1 x_1 + \ldots + \beta_D x_D$$

The logistic (or sigmoid) function has an easy–to–calculate derivative, which makes it easy to calculate the optimum parameters, and it has a range of [0, 1], which makes it suitable to represent probabilities.

Q2:

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. **Build a logistic regression classifier, which uses the counts of selected words in news articles to predict the class of the news article (fruit vs. computer).**

Q2:

e.g. document A includes 'apple' once and 'sun' five times.

Use the weights $\hat{\beta} = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4]$ = [0.2, 0.3, -2.2, 3.3, -0.2]

Recall that β_0 is the bias.

Training data:

ID	apple	ibm	lemon	sun	Class
A	1	0	1	5	fruit (1)
B	1	0	1	2	fruit (1)
C	2	0	0	1	fruit (1)
D	2	2	0	0	computer (0)
E	1	2	1	7	computer (0)

Test instance:

ID	apple	ibm	lemon	sun	Class
T	1	2	1	5	computer (0)

Q2(a):

$\hat{\beta} = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4] = [0.2, 0.3, -2.2, 3.3, -0.2]$

Explain how these parameters are used in a logistic regression model.

Q2(b):

- Here, output is either 0 (computer) or 1 (fruit)
- Four numeric features (apple, ibm, lemon, sun).

$\hat{\beta} = [\beta_0,\beta_1,\beta_2,\beta_3,\beta_4] = [0.2, 0.3, -2.2, 3.3, -0.2]$

Predict the label for the test instance.

- Logistic regression → predict probability of class by doing linear regression of the features (X) and their corresponding weights (beta)

$$P(y=1|x_1, x_2, ..., x_D) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+...+\beta_D x_D)}} = \sigma(\beta_0 + \beta_1 x_1 + ... + \beta_D x_D)$$

- β_1 to β_4 = respective importance of the features 1 to 4 for predicting y=1
  - e.g. β_2 = -2.2 → how important feature 2 ('ibm') is for predicting the fruit class
- β_0 is the bias term for the regression

Q2(b):

$\hat{\beta} = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4] = [0.2, 0.3, -2.2, 3.3, -0.2]$

Predict the label for the test instance.

Training data:

ID	apple	ibm	lemon	sun	Class
A	1	0	1	5	fruit (1)
B	1	0	1	2	fruit (1)
C	2	0	0	1	fruit (1)
D	2	2	0	0	computer (0)
E	1	2	1	7	computer (0)

Test instance:

ID	apple	ibm	lemon	sun	Class
T	1	2	1	5	computer (0)

$$P(y=1|x_1, x_2, ..., x_D) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+...+\beta_D x_D)}} = \sigma(\beta_0 + \beta_1 x_1 + ... + \beta_D x_D)$$

Q2(b):

$\hat{\beta} = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4] = [0.2, 0.3, -2.2, 3.3, -0.2]$

<u>Predict the label for the test instance.</u>

Test instance:

ID	apple	ibm	lemon	sun	Class
T	1	2	1	5	computer (0)

$$P(y=1|T) = \sigma(\beta_0 + \beta_1 t_1 + \beta_2 t_2 + \beta_3 t_3 + \beta_4 t_4)$$

$$= \sigma(0.2 + 0.3 \times 1 + (-2.2) \times 2 + 3.3 \times 1 + (-0.2) \times 5) = \sigma(-1.6)$$

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

- **0.17 < 0.5, predict class 0 (computer) → correct!**

Q2(c):

$\hat{\beta} = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4] = [0.2, 0.3, -2.2, 3.3, -0.2]$

Recall the conditional likelihood objective $-log\mathcal{L}(\beta)$ defined below, which is used as the loss function for the model. Verify that this value is lower when the model predicts the correct label for instance T and higher when the model predicts the incorrect label.

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

Test instance:

ID	apple	ibm	lemon	sun	Class
T	1	2	1	5	computer (0)

Q2(c):

ID	apple	ibm	lemon	sun	Class
T	1	2	1	5	computer (0)

$\hat{\beta} = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4]$

$= [0.2, 0.3, -2.2, 3.3, -0.2]$

Recall the conditional likelihood objective $-log\mathcal{L}(\beta)$ defined below, which is used as the loss function for the model. Verify that this value is lower when the model predicts the correct label for instance T and higher when the model predicts the incorrect label.

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

- **Compute the negative log-likelihood of the test instance:**
  a. **Assuming the true label was y = 1 (fruit), i.e., classifier made a mistake;**
  b. **Assuming that the true label was y = 0 (computer), i.e., classifier correct.**

Q2(c):

ID	apple	ibm	lemon	sun	Class
T	1	2	1	5	computer (0)

$\hat{\beta} = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4]$

$= [0.2, 0.3, -2.2, 3.3, -0.2]$

A. Assuming true label $y = 1$ and prediction $\hat{y} = 0$ (model was incorrect):

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

$$= -[(1)log(\sigma(-1.6)) + 0] = -log(0.17) = 1.77$$

B. Assuming true label $y = 0$ and prediction $\hat{y} = 0$ (model was correct):

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

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

- **As expected, the negative log likelihood is lower when the predicted label is correct than when the label is incorrect**

Q3:

$$\frac{\partial \mathcal{L}(\beta)}{\partial \beta_j} = \sum_i (y_i - \sigma(X_i; \beta)) x_{1i}$$

$$\frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \ldots + \beta_D x_D)}} = \sigma(\beta_0 + \beta_1 x_1 + \ldots + \beta_D x_D)$$

$\hat{\beta} = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4] = [0.2, 0.3, -2.2, 3.3, -0.2]$

Compute a single gradient ascent update for β_1.

Recall that for each feature *j*, weight update:

$$\beta_j \leftarrow \beta_j + \alpha \frac{\partial \mathcal{L}(\beta)}{\partial \beta_j}$$

with the conditional likelihood ***L*** computed over all training instances *i*. Do the update assuming a learning rate alpha = 0.

Training data:

ID	apple	ibm	lemon	sun	Class
A	1	0	1	5	fruit (1)
B	1	0	1	2	fruit (1)
C	2	0	0	1	fruit (1)
D	2	2	0	0	computer (0)
E	1	2	1	7	computer (0)

Test instance:

ID	apple	ibm	lemon	sun	Class
T	1	2	1	5	computer (0)

Q3:

$$\frac{\partial \mathcal{L}(\beta)}{\partial \beta_j} = \sum_i (y_i - \sigma(X_i;\beta))x_{1i}$$

$$\frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\ldots+\beta_D x_D)}} = \sigma(\beta_0+\beta_1 x_1+\ldots+\beta_D x_D)$$

$\hat{\beta} = [\beta_0,\beta_1,\beta_2,\beta_3,\beta_4] = [0.2, 0.3, -2.2, 3.3, -0.2]$

Compute a single gradient ascent update for β_1.

ID	apple	ibm	lemon	sun	Class
A	1	0	1	5	fruit (1)
B	1	0	1	2	fruit (1)
C	2	0	0	1	fruit (1)
D	2	2	0	0	computer (0)
E	1	2	1	7	computer (0)

Step 1: calculate sigma for all training instances

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

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

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

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

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

Q3:

$$\frac{\partial \mathcal{L}(\beta)}{\partial \beta_j} = \sum_i (y_i - \sigma(X_i;\beta))x_{1i}$$

$$\frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\ldots+\beta_D x_D)}} = \sigma(\beta_0+\beta_1 x_1+\ldots+\beta_D x_D)$$

ID	apple	ibm	lemon	sun	Class
A	1	0	1	5	fruit (1)
B	1	0	1	2	fruit (1)
C	2	0	0	1	fruit (1)
D	2	2	0	0	computer (0)
E	1	2	1	7	computer (0)

$\hat{\beta} = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4] = [0.2, 0.3, -2.2, 3.3, -0.2]$

Compute a single gradient ascent update for β_1.

Step 1: calculate update to beta_1

$$\beta_1 = \beta_1 + \alpha \sum_{i \in \{A,B,C,D,E\}} (y_i - \sigma(X_i;\beta))x_{1i}$$

$$\beta_1 = 0.3 + 0.1[(y_A - \sigma(X_A;\beta))x_{1A} + (y_B - \sigma(X_B;\beta))x_{1B} + (y_C - \sigma(X_C;\beta))x_{1C} + (y_D - \sigma(X_D;\beta))x_{1D} + (y_E - \sigma(X_E;\beta))x_{1E}]$$

$$\beta_1 = 0.3 + 0.1[((1-0.94)\times 1) + ((1-0.97)\times 1) + ((1-0.65)\times 2) + ((0-0.03)\times 2) + ((0-0.12)\times 1)]$$

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

Classifier Combination

Q4:

Describe how to build a random forest for a given dataset.

Q4:

Describe how to build a <u>random forest</u> for a given dataset.

Random forest: Ensemble of multiple decision trees, classification via voting

1. <u>Feature selection</u>: For each tree in the forest, randomly select a subset (k) of features (M) to use for training. This helps to reduce overfitting and improve the generalisation performance of the model.
2. <u>Decision tree construction</u>: Using the selected features, construct a decision tree by randomly selecting N training instances with replacement similar to bagging.
3. <u>Random forest construction</u>: Combine the decision trees into a random forest by taking the majority vote of the individual trees.

Q4(a):

What is the benefit of bagging in random forests?

Q4(a):

What is the benefit of bagging in random forests?

- Bootstrap aggregating
- Intuition:
  - More data → better performance (lower variance)
  - Construct new datasets through random sampling & replacement from the training set

- Benefit: Bagging helps build uncorrelated decision trees

Q4(b):

What is the impact of the number of trees in a random forest?

Q4(b):

What is the impact of the number of trees in a random forest?

- More trees (assuming the trees are uncorrelated) reduce variance.
  - Good!

- More trees → need to build more trees → need to train more trees
  - Take more time to train and classify
  - The complexity of the algorithm grows linearly with the number of trees.
  - Bad, depending on computational resources...

Q4(c):

What will happen if the random number of features chosen for splitting nodes in a random forest is very large?

Q4(c):

What will happen if the random number of features chosen for splitting nodes in a random forest is very large?

- More features → More similar trees:
  - Subset at each split becomes less diverse → trees choose similar features to split on → more correlated trees
- More correlation → Less ensemble benefit:
  - Random Forest relies on averaging many independent trees to reduce variance
  - If trees are highly correlated, averaging doesn't reduce variance much → the ensemble becomes less powerful → accuracy drops

Q5:

Under what circumstances we prefer stacking to boosting and bagging?

1. **Boosting:** Tune base classifiers to focus on the hard-to-classify instances
    - Reduce bias by iteratively adjusting the weights of misclassified data points
    - Yet, if base models too simple, can ***overfit*** on misclassified points of previous models
2. **Bagging:** Construct new datasets through random sampling & replacement
    - Reduce overfitting by reducing variance
    - Yet, if base models too complex, can ***overfit*** on their respective subsets of data, leading to an overall overfitting of the bagging ensemble
3. **Stacking:** Train a meta-classifier over the outputs of base classifiers
    - Combines the strengths of both bagging and boosting while mitigating weaknesses
    - Can reduce both bias and variance by stacking models trained with different algorithms
    - Yet, more complex to implement & tune → requires more ***computational resources***

→ Stacking good when data is complex & want to consider predictions from heterogeneous models