

Logistic Regression



Binary Classification

- Given an instance \mathbf{x} , we must classify it to either positive (1) or negative (0) class.
 - We can use $\{1, -1\}$ instead of $\{1, 0\}$
 - (but we will use 1, 0 as it simplifies the notation in subsequent derivations)
- Binary classification can be seen as learning a function f such that $f(\mathbf{x})$ returns either 1 or 0 indicating the predicted class.

Some terms in Machine Learning

- Training dataset with N instances
 - $\{(x_1, t_1), \dots, (x_N, t_N)\}$ This can also be written as $\{(x_n, t_n)\}_{n=1}^N$
- Target label (class)
 - t : The class labels in the training dataset
 - Annotated by humans (supervised learning)
- Predicted label
 - Labels predicted by our model $f(x)$

From Naive Bayes to Logistic Regression

- Recall Naive Bayes Classifier
 - Predict:

$$\hat{Y} = \underset{y}{\operatorname{argmax}} P(\mathbf{X} \mid Y) = \underset{y}{\operatorname{argmax}} P(\mathbf{X} \mid Y)P(Y)$$

- We use independence assumption:

$$P(\mathbf{X} \mid Y) = P(X_1, X_2, \dots, X_m \mid Y)P(Y) = \prod_{i=1}^m P(X_i \mid Y)$$

- Can we model $P(Y \mid \mathbf{X})$ directly?

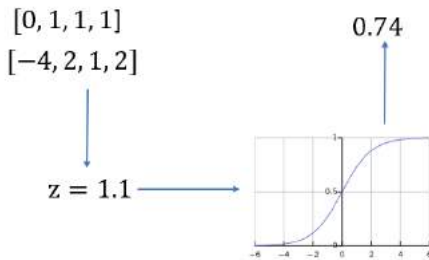
From Naive Bayes to Logistic Regression

- Can we model $P(Y \mid \mathbf{X})$ directly?
 - We use logistic regression model to achieve this
 - Given X, \mathbf{W} we compute $P(Y = 1 \mid X)$

$$\begin{array}{l} [0, 1, 1, 1] \\ [-4, 2, 1, 2] \end{array} \longrightarrow 0.74$$

From Naive Bayes to Logistic Regression

- Can we model $P(Y | \mathbf{X})$ directly?
 - We use logistic regression model to achieve this
 - Given X, \mathbf{W} we compute $P(Y = 1 | X)$



Generative vs. Discriminative Classifier



Generative

- goal of understanding how dogs look and cats look
- model can 'generate' (draw) a dog
- given a test image, choose the closest model as 'label'

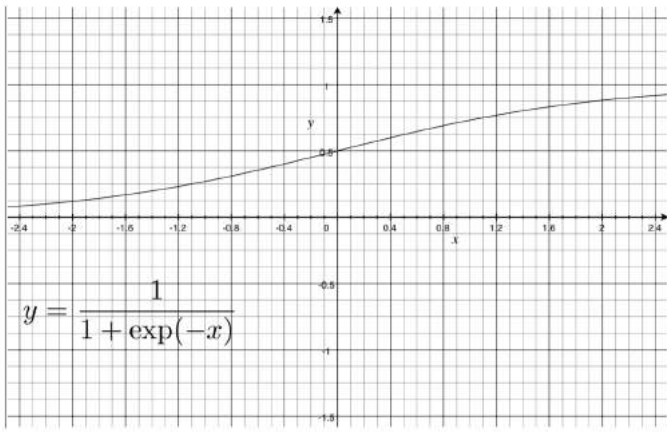
Discriminative

- trying to learn to distinguish the classes (not learning much about them)
- If one feature neatly separates the classes, the model is satisfied.
- e.g., dogs wearing collars and cat aren't

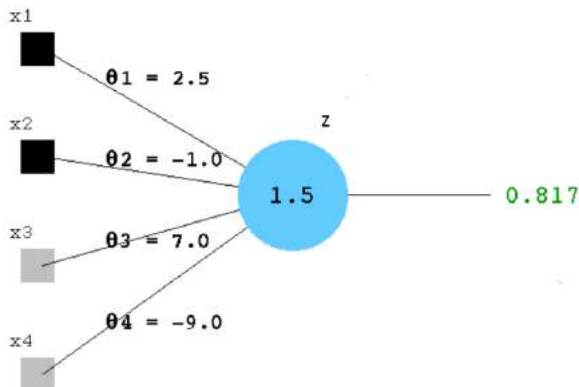
Logistic Regression

- is not a *regression* model
- is a *classification* model
- is a **Discriminative** classifier and not a **Generative** classifier
- is the basis of many advanced machine learning methods
 - neural networks, deep learning, conditional random fields, ...
- try to fit a logistic Sigmoid function to predict the class labels

Logistic Sigmoid Function



Logistic Regression Example



How to find Parameters of the Model

- In logistic regression, given \mathbf{x} , \mathbf{w} we compute $P(Y = y \mid \mathbf{x})$
- How to compute \mathbf{w} ?
- Maximum Likelihood Estimate/Principle (MLE): parameter estimation method to find parameters of the model
- The parameter values are found such that the values maximise the *likelihood* of making the observations given the parameters.

MLE Example

- Consider a bag containing 3 balls.
- Each ball is either red or blue (we have no other information)
- number of blue balls (θ) might be 0, 1, 2, or 3
- we are allowed to choose 4 balls at random with replacement.
- the random variables X_1, X_2, X_3 and X_4 are defined as follows:

$$X_i = \begin{cases} 1 & \text{if the } i\text{th chosen ball is blue} \\ 0 & \text{if the } i\text{th chosen ball is red} \end{cases}$$

MLE Example

- X_i 's are i.i.d and $X_i \sim \text{Bernoulli}(\frac{\theta}{3})$.
- let us say the following values for X_i 's are observed:

$$x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$$

- Given the above:
 - For each possible value of θ , can we compute the probability of the observed sample
 - Can we find the value of θ for which the probability of the observed sample is the largest?

MLE Example

- Since $X_i \sim \text{Bernoulli}(\frac{\theta}{3})$, we have:

$$P_{X_i}(x) = \begin{cases} \frac{\theta}{3} & \text{for } x = 1 \\ 1 - \frac{\theta}{3} & \text{for } x = 0 \end{cases}$$

- Since X_i 's are independent, the joint PMF of X_1, X_2, X_3 and X_4 , can be written as:

$$P_{X_1 X_2 X_3 X_4}(x_1, x_2, x_3, x_4) = P_{X_1}(x_1)P_{X_2}(x_2)P_{X_3}(x_3)P_{X_4}(x_4)$$

MLE Example

- Thus:

$$\begin{aligned} P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1) &= \frac{\theta}{3} \cdot \left(1 - \frac{\theta}{3}\right) \cdot \frac{\theta}{3} \cdot \frac{\theta}{3} \\ &= \left(\frac{\theta}{3}\right)^3 \left(1 - \frac{\theta}{3}\right). \end{aligned}$$

θ	$P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1; \theta)$
0	0
1	0.0247
2	0.0988
3	0

Log Likelihood

- Parameters of logistic regression model are chosen using maximum likelihood estimation (MLE) method.
- There are two steps:
 - 1 write the log-likelihood function (define cross entropy function)
 - 2 find the values of \mathbf{w} that maximise the log-likelihood function

Logistic Regression Model

- for an individual training instance (\mathbf{x}_i, t_i) :

$$P(t_i = 1 \mid \mathbf{x}_i) = y_i = \sigma(\mathbf{w}^\top \mathbf{x}_i) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x}_i)}$$

$$P(t_i = 0 \mid \mathbf{x}_i) = 1 - y_i$$

- using the equation form of the PMF of Bernoulli distribution, where $t_n \in \{0, 1\}$, the probability of a single instance can be written as:

$$P(t_i = t \mid \mathbf{x}_i) = y_i^{t_i} \{1 - y_i\}^{(1-t_i)}$$

Likelihood

- for a dataset $\{x_n, t_n\}$, where $t_n \in \{0, 1\}$, with $n = 1, \dots, N$, the likelihood function can be written as:

$$L(\mathbf{w}) = P(\mathbf{t} \mid \mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{(1-t_n)}$$

where $\mathbf{t} = (t_1, \dots, t_N)^\top$ and $y_n = p(t_n = 1 \mid x_n)$

Cross Entropy

- define a negative logarithm of the likelihood to get *cross entropy* error function $E(w)$:

$$L(\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{(1-t_n)}$$

$$LL(\mathbf{w}) = E(\mathbf{w}) = - \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

Derivation of the Gradient

- differentiating $E(w)$ w.r.t \mathbf{w} , we get the gradient $\nabla E(w)$:

$$E(\mathbf{w}) = - \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ t_n \frac{1}{y_n} \frac{\partial y_n}{\partial \mathbf{w}} + (1 - t_n) \frac{1}{1 - y_n} \left(-\frac{\partial y_n}{\partial \mathbf{w}}\right) \right\}$$

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{t_n}{y_n} - \frac{1 - t_n}{1 - y_n} \right\} \left(\frac{\partial y_n}{\partial \mathbf{w}} \right)$$

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{t_n(1 - y_n) - (1 - t_n)y_n}{y_n(1 - y_n)} \right\} \left(\frac{\partial y_n}{\partial \mathbf{w}} \right)$$

Derivation of the Gradient

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{t_n(1 - y_n) - (1 - t_n)y_n}{y_n(1 - y_n)} \right\} \left(\frac{\partial y_n}{\partial \mathbf{w}} \right)$$

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{(t_n - y_n)}{y_n(1 - y_n)} \right\} \left(\frac{\partial y_n}{\partial \mathbf{w}} \right) \quad (1)$$

Derivation of the Gradient

- differentiating y_n w.r.t \mathbf{w} :

$$\frac{\partial y_n}{\partial w} = \frac{\partial}{\partial w} \left\{ \frac{1}{1 + \exp^{-wx_n}} \right\}$$

$$\frac{\partial y_n}{\partial w} = \frac{(1 + \exp^{-wx_n}) \frac{\partial(1)}{\partial w} - 1 \cdot \frac{\partial}{\partial w}(1 + \exp^{-wx_n})}{(1 + \exp^{-wx_n})^2}$$

$$\frac{\partial y_n}{\partial w} = \frac{-(\exp^{-wx_n}) (-x_n)}{(1 + \exp^{-wx_n})^2}$$

$$\frac{\partial y_n}{\partial w} = \frac{1}{1 + \exp^{-wx_n}} \frac{\exp^{-wx_n} x_n}{1 + \exp^{-wx_n}}$$

Derivation of the Gradient

- differentiating y_n w.r.t \mathbf{w} :

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \frac{\exp^{-wx_n} x_n}{1+\exp^{-wx_n}}$$

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \frac{(1+\exp^{-wx_n})-1}{1+\exp^{-wx_n}} x_n$$

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \left(\frac{1+\exp^{-wx_n}}{1+\exp^{-wx_n}} - \frac{1}{1+\exp^{-wx_n}} \right) x_n$$

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \left(1 - \frac{1}{1+\exp^{-wx_n}} \right) x_n$$

$$\frac{\partial y_n}{\partial w} = y_n(1 - y_n)x_n \quad (2)$$

Derivation of the Gradient

- Substituting (2) in (1), we get:

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{(t_n - y_n)}{y_n(1 - y_n)} \right\} y_n(1 - y_n)x_n$$

- Simplifying further:

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N (t_n - y_n)x_n$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n)x_n \quad (3)$$

Updating the weight vector

- Generic update rule

$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \eta \nabla E(\mathbf{w})$$

- Update rule with cross-entropy error function

$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \eta (y_n - t_n) \mathbf{x}_n$$

- the contribution of the gradient from a data point n is given by the error $(y_n - t_n)$ times the feature vector \mathbf{x}_n

Logistic Regression Algorithm

- Given a set of training instances $\{(x_1, t_1), \dots, (x_N, t_N)\}$, learning rate (η) and iterations T :
- Initialise weight vector $\mathbf{w} = \mathbf{0}$
- For j in $1, \dots, T$
 - for n in $1, \dots, N$
 - if $\text{pred}(\mathbf{x}_i \neq t_i)$ #misclassification
 - $\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \eta(y_n - t_n)\mathbf{x}_n$
- Return the final weight vector \mathbf{w}

Prediction Function (*pred*)

- Given the weight vector \mathbf{w} , returns the class label for an instance \mathbf{x}
 - if $\mathbf{w}^T \mathbf{x} > 0$:
 - predicted label = +1 # positive class
 - else:
 - predicted label = 0 # negative class

Online vs. Batch

- Online vs. Batch Logistic Regression
 - The algorithm we discussed in the previous slides is an *online algorithm* because it considers only one instance at a time and updates the weight vector
 - Referred to as the **Stochastic Gradient Descent (SGD)** update
 - In the batch version, we will compute the cross-entropy error over the *entire* training dataset and then update the weight vector
 - Popular optimisation algorithm for the batch learning of logistic regression is the Limited Memory BFGS (L-BFGS) algorithm
- Batch version is slow compared to the SGD version. But shows slightly improved accuracies in many cases
- SGD version can require multiple iterations over the dataset before it converges (if ever)
- SGD is a technique that is frequently used with large scale machine learning tasks (even when the objective function is non-convex)

Regularisation

- Regularisation
 - Reducing overfitting in a model by constraining it (reducing the complexity/no. of parameters)
 - For classifiers that use a weight vector, regularisation can be done by minimising the norm (length) of the weight vector.
 - Several popular regularisation methods exist
 - L2 regularisation (ridge regression or Tikhonov regularisation)
 - L1 regularisation (Lasso regression)
 - L1+L2 regularisation (mixed regularisation)

L2 Regularisation

- Let us denote the Loss of classifying a dataset D using a model represented by a weight vector \mathbf{w} by $L(D, \mathbf{w})$ and we would like to impose L2 regularisation on \mathbf{w} .
- The overall objective to minimise can then be written as follows (here λ is called the regularisation coefficient and is set via cross-validation)

$$J(D, \mathbf{w}) = L(D, \mathbf{w}) + \lambda ||\mathbf{w}||_2^2$$

- The gradient of the overall objective simply becomes the addition of the loss-gradient and the scaled weight vector \mathbf{w} .

$$\frac{\partial J(D, \mathbf{w})}{\partial \mathbf{w}} = \frac{\partial L(D, \mathbf{w})}{\partial \mathbf{w}} + 2\lambda \mathbf{w}$$

Examples

- Note that SGD update for minimising a loss multiplies the loss gradient by a negative learning rate (η). Therefore, the L2 regularised update rules will have a $-2\eta\lambda\mathbf{w}$ term as shown in the following examples
- L2 regularised Perceptron update (for a misclassified instance we do)

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + t\mathbf{x} - 2\lambda\mathbf{w}^{(k)}$$

- L2 regularised logistic regression

$$\begin{aligned}\mathbf{w}^{(k+1)} &= \mathbf{w}^{(k)} - \eta((y - t)\mathbf{x} + 2\lambda\mathbf{w}^{(k)}) \\ &= (1 - 2\lambda\eta)\mathbf{w}^{(k)} - \eta(y - t)\mathbf{x}\end{aligned}$$

How to set λ

- Split your training dataset into training and validation parts (eg. 80%-20%)
- Try different values for λ (typically in the logarithmic scale). Train a different classification model for each λ and select the value that gives the best performance (eg. accuracy) on the validation data.
- $\lambda = 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^1, 10^2, 10^3, 10^4, 10^5$

References

- Bishop (Pattern Recognition and Machine Learning)
Section 4.3.2
- Software
 - Scikit-learn (Python)
 - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html