Popular Machine Learning Methods: Idea, Practice and Math

Part 2, Chapter 2, Section 3: Logistic Regression

Yuxiao Huang

Data Science, Columbian College of Arts & Sciences
George Washington University

Fall 2024

Reference

- This set of slices was largely built on the following 7 wonderful books and a wide range of fabulous papers:

HML Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd Edition)
PML Python Machine Learning (3rd Edition)
ESL The Elements of Statistical Learning (2nd Edition)
PRML Pattern Recognition and Machine Learning
NND Neural Network Design (2nd Edition)
LFD Learning From Data
RL Reinforcement Learning: An Introduction (2nd Edition)

- For most materials covered in the slides, we will specify their corresponding books and papers for further reference.

Code Example & Case Study

- See related code examples in github repository: /p2_c2_s3_logistic_regression/code_example
- See related case studies of Kaggle Competition in github repository: /p2_c2_s3_logistic_regression/case_study

Table of Contents

1. Learning Objectives
2. Motivating Example
3. Logistic Regression
4. Sigmoid Regression
5. Sigmoid Regression in Sklearn
6. Training Sigmoid Regression
7. Softmax Regression
8. Softmax Regression in Sklearn
9. Training Softmax Regression
10. Class Imbalance: Revisit
11. Metrics for Classification
12. Appendix
13. Bibliography

Learning Objectives: Expectation

- It is expected to understand
  - the idea of Classification
  - the idea of three kinds of classification:
    - Binary Classification
    - Multiclass Classification
    - Multilabel Classification
  - the good practice for using sklearn LogisticRegression
  - the idea of Bernoulli and Categorical distribution
  - the idea of Binomial and Multinomial distribution
  - the idea of Maximum Likelihood Estimation, Kullback–Leibler Divergence and their connection
  - the idea of Binomial Cross Entropy and Multinomial Cross Entropy
  - the idea and implementation of two forms of Logistic Regression:
    - Sigmoid Regression
    - Softmax Regression
  - the idea of why both sigmoid and softmax regression suffer from class imbalance
  - the idea and good practice of three kinds of metrics for classification:
    - Confusion Matrix
    - Accuracy
    - Precision, Recall and F1-Score

Learning Objectives: Recommendation

- It is recommended to understand
  - the math of Bernoulli and Categorical distribution
  - the math of Binomial and Multinomial distribution
  - the math of Maximum Likelihood Estimation, Kullback–Leibler Divergence and their connection
  - the math of Binomial Cross Entropy and Multinomial Cross Entropy
  - the math of two forms of Logistic Regression:
    - Sigmoid Regression
    - Softmax Regression
  - the math of why both sigmoid and softmax regression suffer from class imbalance
  - the math of three kinds of metrics for classification:
    - Confusion Matrix
    - Accuracy
    - Precision, Recall and F1-Score

Kaggle Competition: Predicting Breast Cancer

Figure 1: Kaggle competition: predicting breast cancer. Picture courtesy of Kaggle.

- Breast Cancer Wisconsin (Diagnostic) dataset:
  - features: ID number + 30 variables computed from a digitized image of a fine needle aspirate (FNA) of a breast mass, describing characteristics of the cell nuclei present in the image
  - target: the diagnosis of breast cancer, Benign (B) or Malignant (M)

Kaggle Competition: Digit Recognizer

Figure 2: Kaggle competition: digit recognizer. Picture courtesy of Kaggle.

- Modified National Institute of Standards and Technology (MNIST) dataset:
  - features: flattened 28×28 (i.e., 784) pixels (taking value in $[0, 255]$) in the image of a digit
  - target: the digit in each image, taking value in $[0, 9]$

Classification

- The goal of the two Kaggle competitions is using the features to predict the target.
- The target in each dataset is a *Discrete* variable, since it can only take finite number of values:
  - Breast Cancer Wisconsin: Benign or Malignant
  - MNIST: $[0, 9]$
- We call this kind of prediction (where the target is a discrete variable) *Classification*.
- We will apply a simple classification model (a.k.a., *Classifier*), *Logistic Regression*, to the two competitions.

Three Types of Classification

- We can divide classification into three categories:
  - *Binary Classification*
    - the target has 2 classes and a sample can only belong to 1 class
    - e.g., the breast cancer diagnosis can either be benign or malignant
  - *Multiclass Classification*
    - the target has >2 two classes and a sample can only belong to 1 class
    - e.g., a digit can take any value between 0 and 9
  - *Multilabel Classification*
    - the target has ≥2 two classes and a sample can belong to any number of the classes at the same time
    - e.g., an image can contain people, dog, frisbee, sky and meadow at the same time
- Here we will focus on binary classification and multiclass classification.
- We will discuss multilabel classification in /p3_c2_s3_convolutional_neural_networks.

Logistic Regression

- Depending on the type of classification (binary or multiclass), logistic regression can be divided into two categories:
  - Sigmoid Regression for binary classification
  - Softmax Regression for multiclass classification
- The two methods are similar in that, they both: F(1|X) = p; F(0|X) = 1-p;
  - model the probability distribution of the classes
  - use Maximum Likelihood Estimation to estimate the parameters
  - pick the class with the highest probability as the predicted class
- The two methods are different in that:
  - sigmoid regression uses the Sigmoid function (hence the name) to model the probability distribution, which in the binary case is Bernoulli distribution
  - softmax regression uses the Softmax function (hence the name) to model the probability distribution, which in the multiclass case is Categorical distribution

Bernoulli Distribution

- We use *Sigmoid Regression* (a.k.a., *Binomial Logistic Regression*) for binary classification, where the probability distribution of a binary target in a sample, $p(y|\boldsymbol{\theta})$, follows a bernoulli distribution:

$$p(y|\boldsymbol{\theta}) = \begin{cases} p & \text{if} \quad y = 1, \\ 1-p & \text{if} \quad y = 0. \end{cases} \tag{1}$$

  Here:
  - y is the class of the sample, which is either 0 or 1
  - $\boldsymbol{\theta}$ is the parameter vector
- Eq. (1) says that:
  - if the class of the sample, y, is 1, then the probability of the class is p
  - if the class of the sample, y, is 0, then the probability of the class is $1-p$
  - the sum of the probabilities across the two classes is 1
- We can also write eq. (1) as

$$p(y|\boldsymbol{\theta}) = p^y(1-p)^{1-y}. \tag{2}$$

Mathematical Model

- Sigmoid regression uses a *Sigmoid* function to model the probability of class 1, p:

$$p = \frac{1}{1+e^{-n}}, \quad \text{Why do we use Sigmoid function?} \tag{3}$$

where the net input, n, is a weighted sum of features:

$$n = b + w_1 x_1 + \cdots + w_n x_n = b + \mathbf{x}\mathbf{w} = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \begin{bmatrix} b & w_1 \dots w_n \end{bmatrix}^\mathsf{T} = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \boldsymbol{\theta}. \tag{4}$$

Here:

- p is the probability of class 1 of a sample
- n is the net input of a sample
- b is the bias
- $\mathbf{w}$ is the $n \times 1$ weight vector (with n being the number of features):

$$\mathbf{w} = \begin{bmatrix} w_1 \cdots w_n \end{bmatrix}^\mathsf{T} \tag{5}$$

- $\mathbf{x}$ is the $1 \times n$ feature vector of a sample (with n being the number of features):

$$\mathbf{x} = \begin{bmatrix} x_1 \cdots x_n \end{bmatrix} \tag{6}$$

- $\boldsymbol{\theta}$ is the $(n+1) \times 1$ parameter vector:

$$\boldsymbol{\theta} = \begin{bmatrix} b & w_1 \dots w_n \end{bmatrix}^\mathsf{T} \tag{7}$$

Architecture

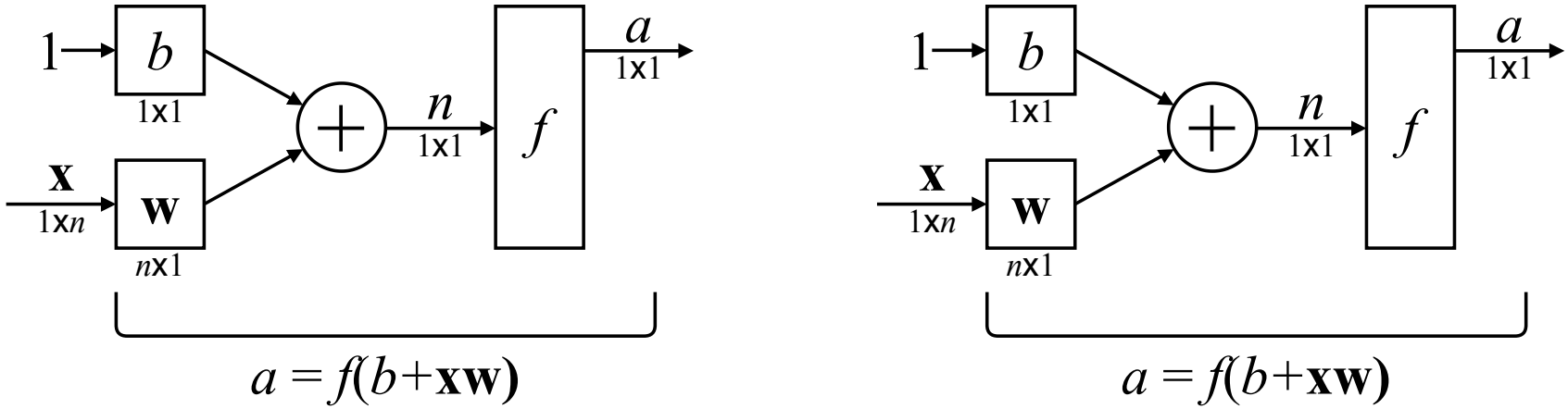


Figure 3: The architecture of linear regression (left) and sigmoid regression (right).

- As fig. 3 shows, a major part of the architecture of linear regression and sigmoid regression are exactly the same:
  - the bias b, net input n and output a for both are scalars
  - the weight $\mathbf{w}$ for both are $n \times 1$ vector
- The only difference between the two architectures is:
  - the activation function in linear regression is identity function:

$$f(n) = n \tag{8}$$

  - the activation function in sigmoid regression is sigmoid function:

$$f(n) = \frac{1}{1 + e^{-n}} \tag{9}$$

Sklearn LogisticRegression: Code Example

- See /p2_c2_s3_logistic_regression/code_example/code_example_bcw:
  1. cell 58
  2. cell 62

Good practice

- As discussed in /p2_c1_data_preprocessing, since sklearn LogisticRegression supports hyperparameter `class_weight`, it is recommended to set the hyperparameter as `'balanced'`, so as to use cost-based method to address class imbalance.

When you set the class_weight parameter to 'balanced', scikit-learn automatically adjusts the weight of each class based on its frequency in the training data.
class weight for class_i = total number of samples/(number of classes * number of samples in class_i)

Classification

- As discussed earlier, we can calculate the probability of class 1, p, using eq. (3)

$$p = \frac{1}{1 + e^{-n}}, \tag{3}$$

where the net input, n, is a weighted sum of features:

$$n = b + w_1 x_1 + \cdots + w_n x_n = b + \mathbf{x}\mathbf{w} = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \begin{bmatrix} b & w_1 \dots w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \boldsymbol{\theta}. \tag{4}$$

- Based on eqs. (3) and (4), we can use the following threshold function to predict a new sample's class, $\widehat{y}$:

$$\widehat{y} = \begin{cases} 1, & \text{if} \quad p \geq 0.5, \\ 0, & \text{if} \quad p < 0.5. \end{cases} \tag{10}$$

0.5 is the default threshold for classification; In practice, this threshold can be tuned based on practical need.

- Eq. (10) says that:
  - if the probability of class 1, p, is $\geq$0.5, then we predict the class of the new sample, $\widehat{y}$, as 1
  - if the probability of class 1, p, is <0.5 (i.e., the probability of class 0, $1 - p$, is $\geq$0.5), then we predict the class of the new sample, $\widehat{y}$, as 0
- However, unlike predicting a new sample's class, estimating the parameters, $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal$, is not straightforward.

Binomial Distribution

- As discussed earlier, the probability distribution of a binary target in one sample follows a bernoulli distribution, $p(y|\boldsymbol{\theta})$, given in eq. (2)

$$p(y|\boldsymbol{\theta}) = p^y(1-p)^{1-y}. \tag{2}$$

- Moreover, the probability distribution of a binary target across m samples follows a *Binomial* distribution:

$$\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \binom{m}{k} \prod_{i=1}^{m} \left(p^i\right)^{y^i} \left(1-p^i\right)^{1-y^i}. \tag{11}$$

Here:

- p^i is the probability of class 1 in sample i
- y^i is the class of sample i, which is either 0 or 1
- m is the number of samples
- k is the sum of y^i across m samples:

$$k = \sum_{i=1}^{m} y_i \tag{12}$$

- $\binom{m}{k}$ is the *Binomial Coefficient* (hence the name of the distribution), which is the total number of ways to choose an (unordered) subset of k elements from m elements:

$$\binom{m}{k} = \frac{n!}{k!(n-k)!} \tag{13}$$

Maximum Likelihood Estimation

- The binomial distribution is given in eq. (11)

$$\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \binom{m}{k} \prod_{i=1}^{m} \left(p^i\right)^{y^i} \left(1-p^i\right)^{1-y^i}, \tag{11}$$

where p is the probability of class 1, given in eq. (3)

$$p = \frac{1}{1+e^{-n}}, \tag{3}$$

and n is the net input, given in eq. (4)

$$n = b + w_1 x_1 + \cdots + w_n x_n = b + \mathbf{x}\mathbf{w} = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \begin{bmatrix} b & w_1 \dots w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \boldsymbol{\theta}. \tag{4}$$

- Since the binomial distribution models the distribution of the data, it is also called the *Likelihood*.
- We can estimate the parameters, $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal$, by maximizing the likelihood, $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$, given in eq. (11):

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \binom{m}{k} \prod_{i=1}^{m} \left(p^i\right)^{y^i} \left(1-p^i\right)^{1-y^i}. \tag{14}$$

- This optimization is called *Maximum Likelihood Estimation* (MLE herefafter).

Kullback–Leibler Divergence

- To see why MLE makes sense, we need the definition of Kullback–Leibler Divergence.
- Let $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)$ be the real distribution of data and $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$ the estimated distribution.
- The *Kullback–Leibler Divergence* (KL-divergence hereafter) from $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)$ to $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$, $D_{KL}\Big(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\Big)$, is

$$D_{KL}\Big(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\Big) = E_{y\sim\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}\left[\log\frac{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})}\right]. \tag{15}$$

- That is, $D_{KL}\Big(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\Big)$ is the expectation (with respect to $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)$) of the logarithmic difference between $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)$ and $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$.
- Moreover, $D_{KL}\Big(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\Big)$ takes the minimal value, zero, when

$$\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*) = \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}). \tag{16}$$

- See proof of eq. (16) in Appendix (page 84).
- It turn out that, maximizing the likelihood (i.e., MLE) equates minimizing the KL-divergence (see proof of this claim in Appendix, page 85).
- Based on eq. (16), MLE indeed makes sense as it minimizes the difference between $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)$ and $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$.
- This is desirable because we want $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$ to be as close to $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)$ as possible.

Binomial Cross Entropy

- As discussed earlier, we can use MLE (eq. (14)) to estimate the parameters, $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \binom{m}{k} \prod_{i=1}^{m} \left(p^i\right)^{y^i} \left(1 - p^i\right)^{1-y^i}. \tag{14}$$

- It turns out that we can write eq. (14) as

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \left(- \sum_{i=1}^{m} \left(y^i \log\left(p^i\right) + (1 - y^i) \log\left(1 - p^i\right) \right) \right). \tag{17}$$

- See the proof of eq. (17) in Appendix (pages 86 and 88).
- The item we want to minimize in eq. (17),

$$- \sum_{i=1}^{m} \left(y^i \log\left(p^i\right) + (1 - y^i) \log\left(1 - p^i\right) \right), \tag{18}$$

is called *Binomial Cross Entropy*, which can be calculated as follows:

1. we first calculate the product of each class and its log probability
2. we then calculate the sum of the product in step 1 across both classes
3. we next calculate the sum of the sum in step 2 across all samples
4. we last calculate the additive inverse of the sum in step 3

Loss Function

- With the definition of binomial cross entropy given in eq. (18)

$$-\sum_{i=1}^{m}\left(y^i \log\left(p^i\right) + (1-y^i)\log\left(1-p^i\right)\right), \tag{18}$$

  we can define the loss function of sigmoid regression.
- It turns out that the loss function of sigmoid regression, $\mathcal{L}(\boldsymbol{\theta})$, is the average binomial cross entropy across m samples:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^i \log\left(p^i\right) + (1-y^i)\log\left(1-p^i\right)\right). \tag{19}$$

  Here:
  - m is the number of samples
  - y^i is the class of sample i, which is either 0 or 1
  - p^i is the probability of class 1 in sample i:

$$p^i = \frac{1}{1+e^{-n^i}}, \tag{20}$$

    where n^i is the net input of class 1 in sample i:

$$n^i = b + w_1 x_1^i + \cdots + w_n x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta} \tag{21}$$

- We can estimate the parameters, $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\mathsf{T}$, by minimizing the loss function in eq. (19).

Gradient Descent

- Similar to linear regression (see /p2_c2_s1_linear_regression), we can also use gradient descent to iteratively update the parameters of sigmoid regression, so as to minimize the loss function, given in eq. (19)

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^i \log\left(p^i\right) + (1-y^i)\log\left(1-p^i\right)\right). \tag{19}$$

- Similar to linear regression, we will use Mini-Batch Gradient Descent (MBGD) to update the parameters, since as discussed in /p2_c2_s1_linear_regression:
  - in theory, MBGD reduces to Batch Gradient Descent (BGD) / Stochastic Gradient Descent (SGD) when the mini-batch contains all the samples / only one sample, so that we can slightly tweak the equations for MBGD (with respect to the mini-batch size) to get the equations for BGD and SGD
  - in practice, MBGD is more popular in deep learning

MBGD: Loss

- Eq. (19) measures the average binomial cross entropy across m samples:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^i \log\left(p^i\right) + (1-y^i)\log\left(1-p^i\right)\right). \tag{19}$$

- Since MBGD updates the parameters using a mini-batch of training samples, $\mathbf{mb}^j$, the loss function, $\mathcal{L}(\boldsymbol{\theta}^j)$, can be written as:

$$\mathcal{L}(\boldsymbol{\theta}^j) = -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i \log\left(p^i\right) + (1-y^i)\log\left(1-p^i\right)\right). \tag{22}$$

  Here:
  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - y^i is the real class of sample i
  - p^i is the probability of class 1 in sample i, given in eq. (20):

$$p^i = \frac{1}{1+e^{-n^i}}, \tag{20}$$

  where n^i is the net input of class 1 in sample i, given in eq. (21):

$$n^i = b + w_1 x_1^i + \cdots + w_n x_n^i = \begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}\boldsymbol{\theta} \tag{21}$$

MBGD: Updating Rule

- As discussed in /p2_c2_s1_linear_regression, the updating rule of MBGD is

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k - \eta_k \left. \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^j}. \tag{23}$$

Here

- $\boldsymbol{\theta}_k^j$ are the parameters in epoch k after the update using mini-batch $\mathbf{mb}^j$
- $\boldsymbol{\theta}_k^{j+1}$ are the parameters in epoch k after the update using mini-batch $\mathbf{mb}_{j+1}$
- $\mathbf{g}_k^j$ is a the gradient of the loss, $\mathcal{L}(\boldsymbol{\theta}^j)$:
  - it is the first-order derivative of $\mathcal{L}(\boldsymbol{\theta}^j)$, with respect to the parameters in epoch k, updated using mini-batch $\mathbf{mb}^j$, $\boldsymbol{\theta}_k^j$
  - it is also the direction that leads to the steepest ascent of $\mathcal{L}(\boldsymbol{\theta}^j)$
- η_k is the learning rate that determines the step size in epoch k (how far we move from $\boldsymbol{\theta}_k^j$ along $\mathbf{g}_k^j$ to search for $\boldsymbol{\theta}_k^{j+1}$)

MBGD: Updating Rule

- By deriving the gradient, $\nabla\mathcal{L}(\boldsymbol{\theta}^j)^\intercal\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^j}$, eq. (23)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k - \eta_k \left.\nabla\mathcal{L}(\boldsymbol{\theta}^j)^\intercal\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^j} \tag{23}$$

  can be written as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \frac{\eta_k}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i - p^i\right)\begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}^\intercal = \boldsymbol{\theta}_k^j + \frac{\eta_k}{|\mathbf{mb}^j|}\begin{bmatrix}\mathbf{1} & \mathbf{X}^j\end{bmatrix}^\intercal(\mathbf{y}^j - \mathbf{p}^j). \tag{24}$$

  Here:
  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - y^i is the real class of sample i
  - p^i is the probability of class 1 in sample i:

$$p^i = \frac{1}{1+e^{-n^i}} \quad \text{where} \quad n^i = b + w_1x_1^i + \cdots + w_nx_n^i = \begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}\boldsymbol{\theta}^j \tag{25}$$

  - $\mathbf{y}^j$ is the $|\mathbf{mb}^j|\times 1$ real class vector of $\mathbf{mb}^j$
  - $\mathbf{p}^j$ is the $|\mathbf{mb}^j|\times 1$ probability vector of class 1 of $\mathbf{mb}^j$

$$\mathbf{p}^j = \frac{1}{1+e^{-\mathbf{n}^j}} \quad \text{where} \quad \mathbf{n}^j = b + w_1\mathbf{x}_1^j + \cdots + w_n\mathbf{x}_n^j = \begin{bmatrix}\mathbf{1} & \mathbf{X}^j\end{bmatrix}\boldsymbol{\theta}^j \tag{26}$$

  - $\mathbf{x}^i$ is the $1\times|\mathbf{mb}^j|$ feature vector of sample i, and $\mathbf{X}^j$ the $|\mathbf{mb}^j|\times n$ feature matrix of $\mathbf{mb}^j$
- See the proof of eq. (24) in Appendix (pages 89 to 98).

MBGD: The Implementation

- See /p2_c2_s3_logistic_regression/code_example/code_example_bcw:
  1. cell 58
  2. cell 63
- See /models/p2_shallow_learning:
  1. cell 5

Good practice

- It is recommended to use small batches (from 2 to 32) in MBGD [Masters and Luschi, 2018].

MBGD + Lasso: Loss

- With the MBGD loss (second item in eq. (27)) and the regularization term of lasso (third item), the loss of MBGD + lasso is the sum of the two:

$$\underbrace{\mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)}_{\text{MBGD + lasso loss}} = \underbrace{-\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i\log\left(p^i\right)+(1-y^i)\log\left(1-p^i\right)\right)}_{\text{MBGD loss}} + \underbrace{\alpha\sum_{j=1}^{n}|w_j|}_{\text{lasso term}}. \tag{27}$$

Here:

- $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
- y^i is the real class of sample i
- p^i is the probability of class 1 in sample i, given in eq. (20):

$$p^i = \frac{1}{1+e^{-n^i}}, \tag{20}$$

where n^i is the net input of class 1 in sample i, given in eq. (21):

$$n^i = b + w_1x_1^i + \cdots + w_nx_n^i = \begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}\boldsymbol{\theta} \tag{21}$$

- α is the regularization parameter

MBGD + Lasso: Updating Rule

- The updating rule of MBGD was given in eq. (23)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k - \eta_k \left. \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}, \tag{23}$$

  where the MBGD loss, $\mathcal{L}(\boldsymbol{\theta}^j)$, was given in eq. (22)

$$\mathcal{L}(\boldsymbol{\theta}^j) = -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(y^i \log\left(p^i\right) + (1 - y^i) \log\left(1 - p^i\right) \right). \tag{22}$$

- By replacing the MBGD loss in eq. (23), $\mathcal{L}(\boldsymbol{\theta}^j)$ (also the second item in eq. (27)), with MBGD + lasso loss, $\mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)$ (first item in eq. (27)), we can write the updating rule of MBGD + lasso as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}. \tag{28}$$

MBGD + Lasso: Updating Rule

- By deriving the gradient in eq. (28), $\nabla \mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)^\intercal\big|_{\boldsymbol{\theta}^j=\boldsymbol{\theta}_k^j}$, we can write eq. (28) as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \left(\frac{1}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \mathbf{p}^j) - \alpha \begin{bmatrix} 0 & \operatorname{sgn}(w_1) \cdots \operatorname{sgn}(w_n) \end{bmatrix}^\intercal \right). \tag{29}$$

  Here
  - η_k is the learning rate in epoch k
  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - $\mathbf{y}^j$ is the $|\mathbf{mb}^j| \times 1$ real class vector of $\mathbf{mb}^j$
  - $\mathbf{p}^j$ is the $|\mathbf{mb}^j| \times 1$ probability vector of class 1 of $\mathbf{mb}^j$, given in eq. (26)

$$\mathbf{p}^j = \frac{1}{1+e^{-\mathbf{n}^j}} \quad \text{where} \quad \mathbf{n}^j = b + w_1 \mathbf{x}_1^j + \cdots + w_n \mathbf{x}_n^j = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \boldsymbol{\theta}^j \tag{26}$$

  - $\mathbf{X}^j$ is the $|\mathbf{mb}^j| \times n$ feature matrix of $\mathbf{mb}^j$
  - sgn is the *Sign* function:

$$\operatorname{sgn}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \tag{30}$$

- The proof of eq. (29) is similar to that in Appendix (pages 89 to 98) and that in Appendix in /p2_c2_s1_linear_regression.

MBGD + Lasso: The Implementation

- See /models/p2_shallow_learning:
  1. cell 5

MBGD + Ridge: Loss

- With the MBGD loss (second item in eq. (31)) and the regularization term of ridge (third item), the loss of MBGD + ridge is the sum of the two:

$$\underbrace{\mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)}_{\text{MBGD + ridge loss}} = \underbrace{-\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i\log\left(p^i\right)+(1-y^i)\log\left(1-p^i\right)\right)}_{\text{MBGD loss}} + \underbrace{\frac{\alpha}{2}\sum_{j=1}^{n}w_j^2}_{\text{ridge term}}. \tag{31}$$

Here:

- $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
- y^i is the real class of sample i
- p^i is the probability of class 1 in sample i, given in eq. (20):

$$p^i = \frac{1}{1+e^{-n^i}}, \tag{20}$$

where n^i is the net input of class 1 in sample i, given in eq. (21):

$$n^i = b + w_1x_1^i + \cdots + w_nx_n^i = \begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}\boldsymbol{\theta} \tag{21}$$

- α is the regularization parameter

MBGD + Ridge: Updating Rule

- The updating rule of MBGD was given in eq. (23)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}, \tag{23}$$

where the MBGD loss, $\mathcal{L}(\boldsymbol{\theta}^j)$, was given in eq. (22)

$$\mathcal{L}(\boldsymbol{\theta}^j) = -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(y^i \log\left(p^i\right) + (1 - y^i) \log\left(1 - p^i\right) \right). \tag{22}$$

- By replacing the MBGD loss in eq. (23), $\mathcal{L}(\boldsymbol{\theta}^j)$ (also the second item in eq. (31)), with MBGD + ridge loss, $\mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)$ (first item in eq. (31)), we can write the updating rule of MBGD + lasso as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}. \tag{32}$$

MBGD + Ridge: Updating Rule

- By deriving the gradient in eq. (32), $\nabla \mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\intercal\big|_{\boldsymbol{\theta}^j=\boldsymbol{\theta}_k^j}$, we can write eq. (32) as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \left(\frac{1}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \mathbf{p}^j) - \alpha \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right). \tag{33}$$

  Here

  - η_k is the learning rate in epoch k
  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - $\mathbf{y}^j$ is the $|\mathbf{mb}^j| \times 1$ real class vector of $\mathbf{mb}^j$
  - $\mathbf{p}^j$ is the $|\mathbf{mb}^j| \times 1$ probability vector of class 1 of $\mathbf{mb}^j$, given in eq. (26)

$$\mathbf{p}^j = \frac{1}{1+e^{-\mathbf{n}^j}} \quad \text{where} \quad \mathbf{n}^j = b + w_1 \mathbf{x}_1^j + \cdots + w_n \mathbf{x}_n^j = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \boldsymbol{\theta}^j \tag{26}$$

  - $\mathbf{X}^j$ is the $|\mathbf{mb}^j| \times n$ feature matrix of $\mathbf{mb}^j$
  - α is the regularization parameter
- The proof of eq. (29) is similar to that in Appendix (pages 89 to 98) and that in Appendix in /p2_c2_s1_linear_regression.

MBGD + Ridge: The Implementation

- See /models/p2_shallow_learning:
  1. cell 5

MBGD + Elastic Net: Loss

- With the MBGD loss (second item in eq. (68)) and the regularization term of elastic net (third item), the loss of MBGD + elastic net is the sum of the two:

$$\underbrace{\mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)}_{\text{MBGD + elastic net loss}} = \underbrace{-\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i\log\left(p^i\right)+(1-y^i)\log\left(1-p^i\right)\right)}_{\text{MBGD loss}} + \underbrace{\alpha\gamma\sum_{j=1}^{n}|w_j|+\frac{\alpha(1-\gamma)}{2}\sum_{j=1}^{n}w_j^2}_{\text{elastic net term}}. \tag{34}$$

Here:

- $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
- y^i is the real class of sample i
- p^i is the probability of class 1 in sample i, given in eq. (20):

$$p^i = \frac{1}{1+e^{-n^i}}, \tag{20}$$

where n^i is the net input of class 1 in sample i, given in eq. (21):

$$n^i = b + w_1 x_1^i + \cdots + w_n x_n^i = \begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}\boldsymbol{\theta} \tag{21}$$

- α and γ are the regularization parameters

MBGD + Elastic Net: Updating Rule

- The updating rule of MBGD was given in eq. (23)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}, \tag{23}$$

  where the MBGD loss, $\mathcal{L}(\boldsymbol{\theta}^j)$, was given in eq. (22)

$$\mathcal{L}(\boldsymbol{\theta}^j) = -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(y^i \log\left(p^i\right) + (1 - y^i) \log\left(1 - p^i\right) \right). \tag{22}$$

- By replacing the MBGD loss in eq. (23), $\mathcal{L}(\boldsymbol{\theta}^j)$ (also the second item in eq. (68)), with MBGD + elastic net loss, $\mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)$ (first item in eq. (68)), we can write the updating rule of MBGD + elastic net as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}. \tag{35}$$

MBGD + Elastic Net: Updating Rule

- By deriving the gradient in eq. (69), $\nabla \mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)^\intercal\big|_{\boldsymbol{\theta}^j=\boldsymbol{\theta}_k^j}$, we can write eq. (69) as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \left(\frac{1}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \mathbf{p}^j) - \alpha\gamma \begin{bmatrix} 0 & \operatorname{sgn}(w_1) \cdots \operatorname{sgn}(w_n) \end{bmatrix}^\intercal - \alpha(1-\gamma) \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right). \tag{36}$$

  Here
  - η_k is the learning rate in epoch k
  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - $\mathbf{y}^j$ is the $|\mathbf{mb}^j| \times 1$ real class vector of $\mathbf{mb}^j$
  - $\mathbf{p}^j$ is the $|\mathbf{mb}^j| \times 1$ probability vector of class 1 of $\mathbf{mb}^j$, given in eq. (26)

$$\mathbf{p}^j = \frac{1}{1+e^{-\mathbf{n}^j}} \quad \text{where} \quad \mathbf{n}^j = b + w_1 \mathbf{x}_1^j + \cdots + w_n \mathbf{x}_n^j = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \boldsymbol{\theta}^j \tag{26}$$

  - $\mathbf{X}^j$ is the $|\mathbf{mb}^j| \times n$ feature matrix of $\mathbf{mb}^j$
  - sgn is the *Sign* function:

$$\operatorname{sgn}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \tag{37}$$

- The proof of eq. (29) is similar to that in Appendix (pages 89 to 98) and that in Appendix in /p2_c2_s1_linear_regression.

MBGD + Elastic Net: The Implementation

- See /models/p2_shallow_learning:
  1. cell 5

Categorical Distribution

- We use *Softmax Regression* (a.k.a., *Multinomial Logistic Regression*) for multiclass classification, where the probability distribution of a categorical target in a sample, $p(y|\boldsymbol{\theta})$, follows a categorical distribution:

$$p(y|\boldsymbol{\theta}) = \begin{cases} p_1 & \text{if} \quad y_1 = 1, \\ \vdots & \vdots \\ p_c & \text{if} \quad y_c = 1. \end{cases} \tag{38}$$

  Here:
  - c is the number of unique classes of the categorical target
  - y is the class of the sample, with:
    - $y_k = 1$ (where $1 \leq k \leq c$) says that the sample belongs to class k
    - $y_k = 0$ (where $1 \leq k \leq c$) says that the sample belongs to other classes
  - p_k (where $1 \leq k \leq c$) is the probability of the sample belonging to class k, where the sum of the probabilities across the c classes is 1:

$$\sum_{k=1}^{c} p_k = 1 \tag{39}$$

  - $\boldsymbol{\theta}$ is the parameter vector
- We can also write eq. (38) as

$$p(y|\boldsymbol{\theta}) = \prod_{k=1}^{c} p_k^{y_k}. \tag{40}$$

Mathematical Model

- Softmax regression uses a *Softmax* function to model the probability distribution of all the possible classes in a sample, $\mathbf{p}$:

$$\mathbf{p} = \frac{e^{\mathbf{n}}}{\sum e^{\mathbf{n}}}, \tag{41}$$

where the net input vector, $\mathbf{n}$, is a vector of weighted sum of features:

$$\mathbf{n} = \mathbf{b} + \mathbf{w}_1 x_1 + \cdots + \mathbf{w}_n x_n = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \begin{bmatrix} \mathbf{b} & \mathbf{w}_1 \cdots \mathbf{w}_n \end{bmatrix}^\mathsf{T} = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \boldsymbol{\theta}. \tag{42}$$

Here:

- $x_1, \ldots, x_n$ are the features of a sample
- $\mathbf{b}$ is a $1 \times c$ bias vector (where c is the number of unique classes of the target)
- $\mathbf{w}_1, \ldots, \mathbf{w}_n$ are $1 \times c$ weight vectors of $x_1, \ldots, x_n$
- $\mathbf{x}$ is a $1 \times n$ feature vector of a sample (with n being the number of features):

$$\mathbf{x} = \begin{bmatrix} x_1 \cdots x_n \end{bmatrix} \tag{43}$$

- $\boldsymbol{\theta}$ is a $(n+1) \times c$ parameter matrix:

$$\boldsymbol{\theta} = \begin{bmatrix} \mathbf{b} & \mathbf{w}_1 \cdots \mathbf{w}_n \end{bmatrix}^\mathsf{T} \tag{44}$$

Architecture

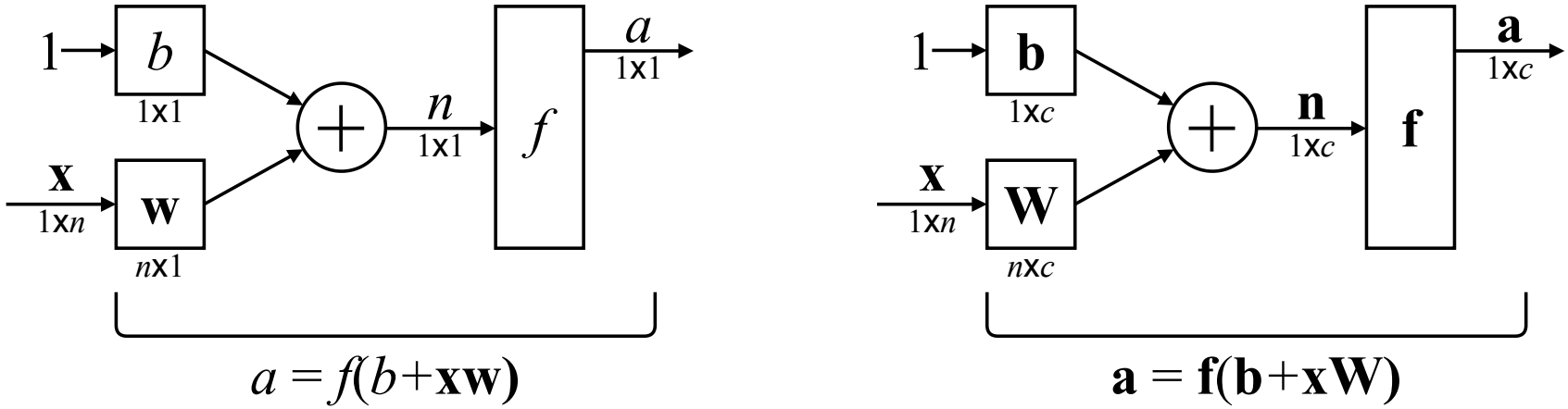


Figure 4: The architecture of sigmoid regression (left) and softmax regression (right).

- As fig. 4 shows, a major part of the architecture of sigmoid regression and softmax regression are different:
  - the bias, net input and output are:
    - scalars (b, n and a) in sigmoid regression
    - $1 \times c$ vectors ($\mathbf{b}$, $\mathbf{n}$ and $\mathbf{a}$) in softmax regression
  - the weights are:
    - scalars ($w_1, \ldots, w_n$) in sigmoid regression
    - $1 \times c$ vectors ($\mathbf{w}_1, \ldots, \mathbf{w}_n$) in softmax regression
  - the activation is:
    - sigmoid (f) in sigmoid regression
    - softmax ($\mathbf{f}$) in softmax regression

Sklearn LogisticRegression: Code Example

- See /p2_c2_s3_logistic_regression/code_example/code_example_bcw:
  1. cell 58
  2. cell 62

Good practice

- As discussed in /p2_c1_data_preprocessing, since sklearn LogisticRegression supports hyperparameter `class_weight`, it is recommended to set the hyperparameter as `'balanced'`, so as to use cost-based method to address class imbalance.

Classification

- As discussed earlier, we can calculate the probability distribution of all the possible classes, $\mathbf{p}$, using eq. (41)

$$\mathbf{p} = \frac{e^{\mathbf{n}}}{\sum e^{\mathbf{n}}}, \tag{41}$$

where the net input vector, $\mathbf{n}$, is a vector of weighted sum of features:

$$\mathbf{n} = \mathbf{b} + \mathbf{w}_1 x_1 + \cdots + \mathbf{w}_n x_n = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \begin{bmatrix} \mathbf{b} & \mathbf{w}_1 \cdots \mathbf{w}_n \end{bmatrix}^\mathsf{T} = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \boldsymbol{\theta}. \tag{42}$$

- Based on eqs. (41) and (42), we have the probability of each possible class k, p_k:

$$p_k = \frac{e^{n_k}}{\sum_{k=1}^{c} e^{n_k}}, \tag{45}$$

where the net input of class k, n_k, is a weighted sum of features:

$$n_k = b^k + w_1^k x_1 + \cdots + w_n^k x_n = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \begin{bmatrix} b^k & w_1^k \cdots w_n^k \end{bmatrix}^\mathsf{T} = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix} \boldsymbol{\theta}^k. \tag{46}$$

- Based on eqs. (45) and (46), we can use the following function to predict a new sample's class, $\widehat{y}$:

$$\widehat{y} = \arg\max_k \; p_k. \tag{47}$$

- Eq. (47) says that, the predicted class, $\widehat{y}$, is the class that has the highest probability across all the classes.
- However, unlike predicting a new sample's class, estimating the parameters, $\boldsymbol{\theta} = \begin{bmatrix} \mathbf{b} & \mathbf{w}_1 \cdots \mathbf{w}_n \end{bmatrix}^\mathsf{T}$, is not straightforward.

Multinomial Distribution

- As discussed earlier, the probability distribution of a categorical target in one sample follows a categorical distribution, $p(y|\boldsymbol{\theta})$, given in eq. (40)

$$p(y|\boldsymbol{\theta}) = \prod_{k=1}^{c} p_k^{y_k}. \tag{40}$$

- Similarly, the probability distribution of a categorical target across m samples follows a *Multinomial* distribution:

$$\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \binom{n}{n_1, \ldots, n_c} \prod_{i=1}^{m} \prod_{k=1}^{c} \left(p_k^i\right)^{y_k^i}. \tag{48}$$

Here:

- p_k^i is the probability of class k in sample i
- $y_k^i = 1$ (where $1 \leq k \leq c$) says that sample i belongs to class k
- $y_k^i = 0$ (where $1 \leq k \leq c$) says that sample i belongs to other classes
- m is the number of samples
- n_k (where $1 \leq k \leq c$) is the sum of y_k^i across m samples and $n = \sum_{k=1}^{c} n_k$
- $\binom{n}{n_1,\ldots,n_c}$ is the *Multinomial Coefficient* (hence the name of the distribution), which is the number of combinations of $n_1, \ldots, n_c$ such that $\sum_{k=1}^{c} n_k = n$:

$$\binom{n}{n_1, \ldots, n_c} = \frac{n!}{\prod_{k=1}^{c} n_k!} \tag{49}$$

Maximum Likelihood Estimation

- The multinomial distribution is given in eq. (48)

$$\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \binom{n}{n_1, \ldots, n_c} \prod_{i=1}^{m} \prod_{k=1}^{c} \left(p_k^i\right)^{y_k^i}, \tag{48}$$

where p_k^i is the probability of class k in sample i,

$$p_k^i = \frac{e^{n_k^i}}{\sum_{l=1}^{c} e^{n_l^i}}, \tag{50}$$

and n_k^i is the net input of class k in sample i:

$$n_k^i = b^k + w_1^k x_1^i + \cdots + w_n^k x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \begin{bmatrix} b^k & w_1^k \cdots w_n^k \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta}^k. \tag{51}$$

- Similar to binomial distribution, since multinomial distribution models the distribution of the data, it is also called the likelihood.
- Similar to binomial distribution, we can also use MLE to estimate the parameters, $\boldsymbol{\theta} = \begin{bmatrix} \mathbf{b} & \mathbf{w}_1 \cdots \mathbf{w}_n \end{bmatrix}^\intercal$, by maximizing the likelihood, $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$, given in eq. (48):

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \binom{n}{n_1, \ldots, n_c} \prod_{i=1}^{m} \prod_{k=1}^{c} \left(p_k^i\right)^{y_k^i}. \tag{52}$$

- As discussed earlier, maximizing the likelihood makes sense since it equates minimizing the KL-divergence (see proof of this claim in Appendix, page 84).

Multinomial Cross Entropy

- As discussed earlier, we can use MLE (eq. (52)) to estimate the parameters, $\boldsymbol{\theta} = \begin{bmatrix} \mathbf{b} & \mathbf{w}_1 \cdots \mathbf{w}_n \end{bmatrix}^\intercal$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \binom{n}{n_1, \ldots, n_c} \prod_{i=1}^{m} \prod_{k=1}^{c} \left(p_k^i\right)^{y_k^i}. \tag{52}$$

- It turns out that we can write eq. (52) as

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \left(-\sum_{i=1}^{m} \sum_{k=1}^{c} y_k^i \log\left(p_k^i\right) \right). \tag{53}$$

- See the proof of eq. (53) in Appendix (pages 99 and 101).
- The item we want to minimize in eq. (53),

$$-\sum_{i=1}^{m} \sum_{k=1}^{c} y_k^i \log\left(p_k^i\right), \tag{54}$$

is called *Multinomial Cross Entropy*, which can be calculated as follows:

1. we first calculate the product of each class and its log probability
2. we then calculate the sum of the product in step 1 across all classes
3. we next calculate the sum of the sum in step 2 across all samples
4. we last calculate the additive inverse of the sum in step 3

Loss Function

- With the definition of multinomial cross entropy given in eq. (54)

$$-\sum_{i=1}^{m}\sum_{k=1}^{c} y_k^i \log\left(p_k^i\right), \tag{54}$$

  we can define the loss function of softmax regression.
- It turns out that the loss function of softmax regression, $\mathcal{L}(\boldsymbol{\theta})$, is the average multinomial cross entropy across m samples:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{c} y_k^i \log\left(p_k^i\right) \quad \text{where :} \tag{55}$$

  - m is the number of samples
  - $y_k^i = 1$ (where $1 \le k \le c$) says that sample i belongs to class k
  - $y_k^i = 0$ (where $1 \le k \le c$) says that sample i belongs to other classes
  - p_k^i is the probability of class k in sample i, given in eq. (50):

$$p_k^i = \frac{e^{n_k^i}}{\sum_{l=1}^{c} e^{n_l^i}}, \tag{50}$$

    where n_k^i is the net input of class k in sample i, given in eq. (51):

$$n_k^i = b^k + w_1^k x_1^i + \cdots + w_n^k x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta} \tag{51}$$

- We can estimate the parameters, $\boldsymbol{\theta} = \begin{bmatrix} \mathbf{b} & \mathbf{w}_1 \cdots \mathbf{w}_n \end{bmatrix}^\intercal$, by minimizing the loss function in eq. (55).

Gradient Descent

- Similar to sigmoid regression, we can also use gradient descent to iteratively update the parameters of softmax regression, so as to minimize the loss function, given in eq. (55)

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{c} y_k^i \log\left(p_k^i\right). \tag{55}$$

- Similar to sigmoid regression, we will use Mini-Batch Gradient Descent (MBGD) to update the parameters, since as discussed in /p2_c2_s1_linear_regression:
  - in theory, MBGD reduces to Batch Gradient Descent (BGD) / Stochastic Gradient Descent (SGD) when the mini-batch contains all the samples / only one sample, so that we can slightly tweak the equations for MBGD (with respect to the mini-batch size) to get the equations for BGD and SGD
  - in practice, MBGD is more popular in deep learning

MBGD: Loss

- Eq. (55) measures the average multinomial cross entropy across m samples:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{c} y_k^i \log\left(p_k^i\right). \tag{55}$$

- Since MBGD updates the parameters using a mini-batch of training samples, $\mathbf{mb}^j$, the loss function, $\mathcal{L}(\boldsymbol{\theta}^j)$, can be written as:

$$\mathcal{L}(\boldsymbol{\theta}^j) = -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\sum_{k=1}^{c} y_k^i \log\left(p_k^i\right) \quad \text{where :} \tag{56}$$

  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - $y_k^i = 1$ (where $1 \le k \le c$) says that sample i belongs to class k
  - $y_k^i = 0$ (where $1 \le k \le c$) says that sample i belongs to other classes
  - p_k^i is the probability of class k in sample i, given in eq. (50):

$$p_k^i = \frac{e^{n_k^i}}{\sum_{l=1}^{c} e^{n_l^i}}, \tag{50}$$

where n_k^i is the net input of class k in sample i, given in eq. (51):

$$n_k^i = b^k + w_1^k x_1^i + \cdots + w_n^k x_n^i = \begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}\boldsymbol{\theta} \tag{51}$$

MBGD: Updating Rule

- As discussed earlier, the updating rule of MBGD is given in eq. (23)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k - \eta_k \left. \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^j}. \tag{23}$$

Here

- $\boldsymbol{\theta}_k^j$ are the parameters in epoch k after the update using mini-batch $\mathbf{mb}^j$
- $\boldsymbol{\theta}_k^{j+1}$ are the parameters in epoch k after the update using mini-batch $\mathbf{mb}_{j+1}$
- $\mathbf{g}_k^j$ is a the gradient of the loss, $\mathcal{L}(\boldsymbol{\theta}^j)$:
  - it is the first-order derivative of $\mathcal{L}(\boldsymbol{\theta}^j)$, with respect to the parameters in epoch k, updated using mini-batch $\mathbf{mb}^j$, $\boldsymbol{\theta}_k^j$
  - it is also the direction that leads to the steepest ascent of $\mathcal{L}(\boldsymbol{\theta}^j)$
- η_k is the learning rate that determines the step size in epoch k (how far we move from $\boldsymbol{\theta}_k^j$ along $\mathbf{g}_k^j$ to search for $\boldsymbol{\theta}_k^{j+1}$)

MBGD: Updating Rule

- By deriving the gradient, $\nabla\mathcal{L}(\boldsymbol{\theta}^j)^\intercal\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^j}$, eq. (23)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k - \eta_k \left.\nabla\mathcal{L}(\boldsymbol{\theta}^j)^\intercal\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^j} \tag{23}$$

  can be written as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \frac{\eta_k}{|\mathbf{mb}^j|}\begin{bmatrix}\mathbf{1} & \mathbf{X}^j\end{bmatrix}^\intercal (\mathbf{Y}^j - \mathbf{P}^j) \quad \text{where :} \tag{57}$$

  - η_k is the learning rate in epoch k
  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - $\mathbf{Y}^j$ is the $|\mathbf{mb}^j| \times c$ indicator matrix of $\mathbf{mb}^j$
  - $\mathbf{P}^j$ is the $|\mathbf{mb}^j| \times c$ probability matrix of $\mathbf{mb}^j$

$$\mathbf{P}^j = \frac{e^{\mathbf{N}^j}}{\sum e^{\mathbf{N}^j}} \quad \text{where} \quad \mathbf{N}^j = \mathbf{1b} + \mathbf{x}_1^j\mathbf{w}_1 + \cdots + \mathbf{x}_n^j\mathbf{w}_n = \begin{bmatrix}\mathbf{1} & \mathbf{X}^j\end{bmatrix}\boldsymbol{\theta}^j \tag{58}$$

  - $\mathbf{X}^j$ is the $|\mathbf{mb}^j| \times n$ feature matrix of $\mathbf{mb}^j$

- See the proof of eq. (57) in Appendix (pages 102 and 107).

MBGD: The Implementation

- See /p2_c2_s3_logistic_regression/code_example/code_example_mnist:
  1. cell 53
  2. cell 58
- See /models/p2_shallow_learning:
  1. cell 5

Good practice

- It is recommended to use small batches (from 2 to 32) in MBGD [Masters and Luschi, 2018].

MBGD + Lasso: Loss

- With the MBGD loss (second item in eq. (59)) and the regularization term of lasso (third item), the loss of MBGD + lasso is the sum of the two:

$$\underbrace{\mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)}_{\text{MBGD + lasso loss}} = \underbrace{-\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\sum_{k=1}^{c} y_k^i \log\left(p_k^i\right)}_{\text{MBGD loss}} + \underbrace{\alpha\sum_{j=1}^{n}|w_j|}_{\text{lasso term}}. \tag{59}$$

Here:

- $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
- $y_k^i = 1$ (where $1 \le k \le c$) says that sample i belongs to class k
- $y_k^i = 0$ (where $1 \le k \le c$) says that sample i belongs to other classes
- p_k^i is the probability of class k in sample i, given in eq. (50):

$$p_k^i = \frac{e^{n_k^i}}{\sum_{l=1}^{c} e^{n_l^i}}, \tag{50}$$

where n_k^i is the net input of class k in sample i, given in eq. (51):

$$n_k^i = b^k + w_1^k x_1^i + \cdots + w_n^k x_n^i = \begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}\boldsymbol{\theta} \tag{51}$$

- α is the regularization parameter

MBGD + Lasso: Updating Rule

- The updating rule of MBGD was given in eq. (23)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k - \eta_k \left. \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}, \tag{23}$$

  where the MBGD loss, $\mathcal{L}(\boldsymbol{\theta}^j)$, was given in eq. (56)

$$\mathcal{L}(\boldsymbol{\theta}^j) = -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \sum_{k=1}^{c} y_k^i \log\left(p_k^i\right). \tag{56}$$

- By replacing the MBGD loss in eq. (23), $\mathcal{L}(\boldsymbol{\theta}^j)$ (also the second item in eq. (59)), with MBGD + lasso loss, $\mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)$ (first item in eq. (59)), we can write the updating rule of MBGD + lasso as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}. \tag{60}$$

MBGD + Lasso: Updating Rule

- By deriving the gradient in eq. (60), $\nabla \mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)^\intercal\big|_{\boldsymbol{\theta}^j=\boldsymbol{\theta}_k^j}$, we can write eq. (60) as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \left(\frac{1}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{Y}^j - \mathbf{P}^j) - \alpha \begin{bmatrix} 0 & \operatorname{sgn}(w_1) \cdots \operatorname{sgn}(w_n) \end{bmatrix}^\intercal \right). \tag{61}$$

  Here

  - η_k is the learning rate in epoch k
  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - $\mathbf{Y}^j$ is the $|\mathbf{mb}^j| \times c$ indicator matrix of $\mathbf{mb}^j$
  - $\mathbf{P}^j$ is the $|\mathbf{mb}^j| \times c$ probability matrix of $\mathbf{mb}^j$

$$\mathbf{P}^j = \frac{1}{1+e^{-\mathbf{N}^j}} \quad \text{where} \quad \mathbf{N}^j = \mathbf{1b} + \mathbf{x}_1^j \mathbf{w}_1 + \cdots + \mathbf{x}_n^j \mathbf{w}_n = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \boldsymbol{\theta}^j \tag{62}$$

  - $\mathbf{X}^j$ is the $|\mathbf{mb}^j| \times n$ feature matrix of $\mathbf{mb}^j$
  - sgn is the *Sign* function:

$$\operatorname{sgn}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \tag{63}$$

- The proof of eq. (61) is similar to that in Appendix (pages 102 and 107) and that in Appendix in /p2_c2_s1_linear_regression.

MBGD + Lasso: The Implementation

- See /models/p2_shallow_learning:
  1. cell 5

MBGD + Ridge: Loss

- With the MBGD loss (second item in eq. (64)) and the regularization term of ridge (third item), the loss of MBGD + ridge is the sum of the two:

$$\underbrace{\mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)}_{\text{MBGD + ridge loss}} = \underbrace{-\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\sum_{k=1}^{c} y_k^i \log\left(p_k^i\right)}_{\text{MBGD loss}} + \underbrace{\frac{\alpha}{2}\sum_{j=1}^{n} w_j^2}_{\text{ridge term}}. \tag{64}$$

Here:

- $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
- $y_k^i = 1$ (where $1 \le k \le c$) says that sample i belongs to class k
- $y_k^i = 0$ (where $1 \le k \le c$) says that sample i belongs to other classes
- p_k^i is the probability of class k in sample i, given in eq. (50):

$$p_k^i = \frac{1}{1+e^{-n_k^i}}, \tag{50}$$

where n_k^i is the net input of class k in sample i, given in eq. (51):

$$n_k^i = b^k + w_1^k x_1^i + \cdots + w_n^k x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta} \tag{51}$$

- α is the regularization parameter

MBGD + Ridge: Updating Rule

- The updating rule of MBGD was given in eq. (23)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}, \tag{23}$$

where the MBGD loss, $\mathcal{L}(\boldsymbol{\theta}^j)$, was given in eq. (56)

$$\mathcal{L}(\boldsymbol{\theta}^j) = -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \sum_{k=1}^{c} y_k^i \log\left(p_k^i\right). \tag{56}$$

- By replacing the MBGD loss in eq. (23), $\mathcal{L}(\boldsymbol{\theta}^j)$ (also the second item in eq. (64)), with MBGD + ridge loss, $\mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)$ (first item in eq. (64)), we can write the updating rule of MBGD + lasso as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}. \tag{65}$$

MBGD + Ridge: Updating Rule

- By deriving the gradient in eq. (65), $\nabla \mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\intercal\big|_{\boldsymbol{\theta}^j=\boldsymbol{\theta}_k^j}$, we can write eq. (65) as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \left(\frac{1}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{Y}^j - \mathbf{P}^j) - \alpha \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right). \tag{66}$$

  Here
  - η_k is the learning rate in epoch k
  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - $\mathbf{Y}^j$ is the $|\mathbf{mb}^j| \times c$ indicator matrix of $\mathbf{mb}^j$
  - $\mathbf{P}^j$ is the $|\mathbf{mb}^j| \times c$ probability matrix of $\mathbf{mb}^j$

$$\mathbf{P}^j = \frac{1}{1+e^{-\mathbf{N}^j}} \quad \text{where} \quad \mathbf{N}^j = \mathbf{1b} + \mathbf{x}_1^j \mathbf{w}_1 + \cdots + \mathbf{x}_n^j \mathbf{w}_n = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \boldsymbol{\theta}^j \tag{67}$$

  - $\mathbf{X}^j$ is the $|\mathbf{mb}^j| \times n$ feature matrix of $\mathbf{mb}^j$
- The proof of eq. (61) is similar to that in Appendix (pages 102 and 107) and that in Appendix in /p2_c2_s1_linear_regression.

MBGD + Ridge: The Implementation

- See /models/p2_shallow_learning:
  1. cell 5

MBGD + Elastic Net: Loss

- With the MBGD loss (second item in eq. (68)) and the regularization term of elastic net (third item), the loss of MBGD + elastic net is the sum of the two:

$$\underbrace{\mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)}_{\text{MBGD + elastic net loss}} = \underbrace{-\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\sum_{k=1}^{c} y_k^i \log\left(p_k^i\right)}_{\text{MBGD loss}} + \underbrace{\alpha\gamma\sum_{j=1}^{n}|w_j| + \frac{\alpha(1-\gamma)}{2}\sum_{j=1}^{n} w_j^2}_{\text{elastic net term}}. \tag{68}$$

Here:

- $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
- $y_k^i = 1$ (where $1 \le k \le c$) says that sample i belongs to class k
- $y_k^i = 0$ (where $1 \le k \le c$) says that sample i belongs to other classes
- p_k^i is the probability of class k in sample i, given in eq. (50):

$$p_k^i = \frac{1}{1+e^{-n_k^i}}, \tag{50}$$

where n_k^i is the net input of class k in sample i, given in eq. (51):

$$n_k^i = b^k + w_1^k x_1^i + \cdots + w_n^k x_n^i = \begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}\boldsymbol{\theta} \tag{51}$$

- α and γ are the regularization parameters

MBGD + Elastic Net: Updating Rule

- The updating rule of MBGD was given in eq. (23)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}, \tag{23}$$

  where the MBGD loss, $\mathcal{L}(\boldsymbol{\theta}^j)$, was given in eq. (56)

$$\mathcal{L}(\boldsymbol{\theta}^j) = -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \sum_{k=1}^{c} y_k^i \log\left(p_k^i\right). \tag{56}$$

- By replacing the MBGD loss in eq. (23), $\mathcal{L}(\boldsymbol{\theta}^j)$ (also the second item in eq. (68)), with MBGD + elastic net loss, $\mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)$ (first item in eq. (68)), we can write the updating rule of MBGD + elastic net as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}. \tag{69}$$

MBGD + Elastic Net: Updating Rule

- By deriving the gradient in eq. (69), $\nabla \mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)^\intercal\big|_{\boldsymbol{\theta}^j=\boldsymbol{\theta}_k^j}$, we can write eq. (69) as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \left(\frac{1}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{Y}^j - \mathbf{P}^j) - \alpha\gamma \begin{bmatrix} 0 & \operatorname{sgn}(w_1) \cdots \operatorname{sgn}(w_n) \end{bmatrix}^\intercal - \alpha(1-\gamma) \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right). \tag{70}$$

  Here
  - η_k is the learning rate in epoch k
  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - $\mathbf{Y}^j$ is the $|\mathbf{mb}^j| \times c$ indicator matrix of $\mathbf{mb}^j$
  - $\mathbf{P}^j$ is the $|\mathbf{mb}^j| \times c$ probability matrix of $\mathbf{mb}^j$

$$\mathbf{P}^j = \frac{1}{1+e^{-\mathbf{N}^j}} \quad \text{where} \quad \mathbf{N}^j = \mathbf{1b} + \mathbf{x}_1^j \mathbf{w}_1 + \cdots + \mathbf{x}_n^j \mathbf{w}_n = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \boldsymbol{\theta}^j \tag{71}$$

  - $\mathbf{X}^j$ is the $|\mathbf{mb}^j| \times n$ feature matrix of $\mathbf{mb}^j$
  - sgn is the *Sign* function:

$$\operatorname{sgn}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \tag{72}$$

- The proof of eq. (61) is similar to that in Appendix (pages 102 to 107) and that in Appendix in /p2_c2_s1_linear_regression.

MBGD + Elastic Net: The Implementation

- See /models/p2_shallow_learning:
  1. cell 5

Class Imbalance: Revisit

Table 1: The first 5 features and target (diagnosis) of Breast Cancer Wisconsin dataset.

id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
842302	M	17.99	10.38	122.80	1001.0
842517	M	20.57	17.77	132.90	1326.0
84300903	M	19.69	21.25	130.00	1203.0
84348301	M	11.42	20.38	77.58	386.1
84358402	M	20.29	14.34	135.10	1297.0

- As discussed in /p2_c1_data_preprocessing, a key challenge in classification is *Class Imbalance*, where the number of samples for one class is higher than the number for the other classes.
- We usually call the class with more samples the *Majority Class*, whereas the class with fewer samples the *Minority Class*.
- Take the target, diagnosis, in table 1 for example:
  - majority class: benign (with 357 or 63% samples)
  - minority class: malignant (with 212 or 37% samples)

The Problem of Class Imbalance

- The problem of class imbalance is that, when training a model from imbalanced data such as Breast Cancer Wisconsin (table 1), the model will have a bias against the minority class (malignant):
  - that is, even before looking at any clinical or pathological status of a patient, the model already decides that it is less likely (or, in the extreme case, impossible) for the patient to have malignant cancer cells
  - however, wrongly predicting cancer cells as benign when they are actually malignant could lead to lethal consequences
- Now let us use the updating rule of sigmoid and softmax regression derived in Appendix (pages 89 to 98 and pages 102 to 107) to explain where the bias against the minority class comes from.

The Bias in Sigmoid Regression

- The updating rule of sigmoid regression is shown in eq. (24)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \frac{\eta_k}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \mathbf{p}^j). \tag{24}$$

  Here:

  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - $\mathbf{y}^j$ is the $|\mathbf{mb}^j| \times 1$ real class vector of $\mathbf{mb}^j$
  - $\mathbf{p}^j$ is the $|\mathbf{mb}^j| \times 1$ probability vector of class 1 of $\mathbf{mb}^j$, given in eq. (26)

$$\mathbf{p}^j = \frac{1}{1 + e^{-\mathbf{n}^j}} \quad \text{where} \quad \mathbf{n}^j = b + w_1 \mathbf{x}_1^j + \cdots + w_n \mathbf{x}_n^j = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \boldsymbol{\theta}^j \tag{26}$$

  - $\mathbf{X}^j$ is the $|\mathbf{mb}^j| \times n$ feature matrix of $\mathbf{mb}^j$

- Since $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 \dots w_n \end{bmatrix}^\intercal$, then based on eq. (24), we have

$$b_k^{j+1} = b_k^j + \frac{\eta_k}{|\mathbf{mb}^j|} \mathbf{1}^\intercal (\mathbf{y}^j - \mathbf{p}^j). \tag{73}$$

The Bias in Sigmoid Regression

- Based on eq. (73)

$$b_k^{j+1} = b_k^j + \frac{\eta_k}{|\mathbf{mb}^j|}\mathbf{1}^\intercal(\mathbf{y}^j - \mathbf{p}^j), \tag{73}$$

if class 1 is the majority class:

1. it is more likely that the samples in mini-batch $\mathbf{mb}^j$ belong to class 1 (i.e., it is more likely that $\mathbf{y}^j$ is 1)
2. since the probability of class 1, $\mathbf{p}^j \in (0, 1)$, it is more likely that $\mathbf{y}^j - \mathbf{p}^j$ is positive
3. based on eq. (73), it is more likely that b will increase
4. based on eq. (26)

$$\mathbf{p}^j = \frac{1}{1 + e^{-\mathbf{n}^j}} \quad \text{where} \quad \mathbf{n}^j = b + w_1\mathbf{x}_1^j + \cdots + w_n\mathbf{x}_n^j = \begin{bmatrix}\mathbf{1} & \mathbf{X}^j\end{bmatrix}\boldsymbol{\theta}^j, \tag{26}$$

when b increases

   1. $\mathbf{n}$ (the net input of class 1) increases
   2. $\mathbf{p}$ (the probability of class 1) increases
   3. $\mathbf{1} - \mathbf{p}$ (the probability of class 0) decreases

5. as a result, the bias against class 0 (the minority class) arises

The Bias in Sigmoid Regression

- Based on eq. (73)

$$b_k^{j+1} = b_k^j + \frac{\eta_k}{|\mathbf{mb}^j|}\mathbf{1}^\intercal(\mathbf{y}^j - \mathbf{p}^j), \tag{73}$$

if class 1 is the minority class:

1. it is less likely that the samples in mini-batch $\mathbf{mb}^j$ belong to class 1 (i.e., it is less likely that $\mathbf{y}^j$ is 1)
2. since the probability of class 1, $\mathbf{p}^j \in (0, 1)$, it is less likely that $\mathbf{y}^j - \mathbf{p}^j$ is positive
3. based on eq. (73), it is less likely that b will increase
4. based on eq. (26)

$$\mathbf{p}^j = \frac{1}{1 + e^{-\mathbf{n}^j}} \quad \text{where} \quad \mathbf{n}^j = b + w_1\mathbf{x}_1^j + \cdots + w_n\mathbf{x}_n^j = \begin{bmatrix}\mathbf{1} & \mathbf{X}^j\end{bmatrix}\boldsymbol{\theta}^j, \tag{26}$$

when b decreases

   1. $\mathbf{n}$ (the net input of class 1) decreases
   2. $\mathbf{p}$ (the probability of class 1) decreases
   3. $\mathbf{1} - \mathbf{p}$ (the probability of class 0) increases

5. as a result, the bias against class 1 (the minority class) arises

The Bias in Softmax Regression

- The updating rule of softmax regression is shown in eq. (57)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \frac{\eta_k}{|\mathbf{mb}^j|}\begin{bmatrix}\mathbf{1} & \mathbf{X}^j\end{bmatrix}^\intercal (\mathbf{Y}^j - \mathbf{P}^j) \quad \text{where :} \tag{57}$$

  Here

  - η_k is the learning rate in epoch k
  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - $\mathbf{Y}^j$ is the $|\mathbf{mb}^j| \times c$ indicator matrix of $\mathbf{mb}^j$
  - $\mathbf{P}^j$ is the $|\mathbf{mb}^j| \times c$ probability matrix of $\mathbf{mb}^j$, given in eq. (71)

$$\mathbf{P}^j = \frac{e^{\mathbf{N}^j}}{\sum e^{\mathbf{N}^j}} \quad \text{where} \quad \mathbf{N}^j = \mathbf{1}\mathbf{b} + \mathbf{x}_1^j\mathbf{w}_1 + \cdots + \mathbf{x}_n^j\mathbf{w}_n = \begin{bmatrix}\mathbf{1} & \mathbf{X}^j\end{bmatrix}\boldsymbol{\theta}^j \tag{71}$$

  - $\mathbf{X}^j$ is the $|\mathbf{mb}^j| \times n$ feature matrix of $\mathbf{mb}^j$

- Since $\boldsymbol{\theta} = \begin{bmatrix}\mathbf{b} & \mathbf{w}_1 \ldots \mathbf{w}_n\end{bmatrix}^\intercal$, then based on eq. (57), we have

$$\mathbf{b}_k^{j+1} = \mathbf{b}_k^j + \frac{\eta_k}{|\mathbf{mb}^j|}\mathbf{1}^\intercal(\mathbf{Y}^j - \mathbf{P}^j). \tag{74}$$

The Bias in Sigmoid Regression

- Based on eq. (74)

$$\mathbf{b}_k^{j+1} = \mathbf{b}_k^j + \frac{\eta_k}{|\mathbf{mb}^j|}\mathbf{1}^\intercal(\mathbf{Y}^j - \mathbf{P}^j), \tag{73}$$

if class l is the majority class:

1. it is more likely that the samples in mini-batch $\mathbf{mb}^j$ belong to class l (i.e., it is more likely that $\mathbf{y}_l^j$ is 1)
2. since the probability of class l, $\mathbf{p}_l^j \in (0, 1)$, it is more likely that $\mathbf{y}_l^j - \mathbf{p}_l^j$ is positive
3. based on eq. (73), it is more likely that b_l will increase
4. based on eq. (71)

$$\mathbf{P}^j = \frac{e^{\mathbf{N}^j}}{\sum e^{\mathbf{N}^j}} \quad \text{where} \quad \mathbf{N}^j = \mathbf{1}\mathbf{b} + \mathbf{x}_1^j\mathbf{w}_1 + \cdots + \mathbf{x}_n^j\mathbf{w}_n = \begin{bmatrix}\mathbf{1} & \mathbf{X}^j\end{bmatrix}\boldsymbol{\theta}^j \tag{71}$$

when b_l increases

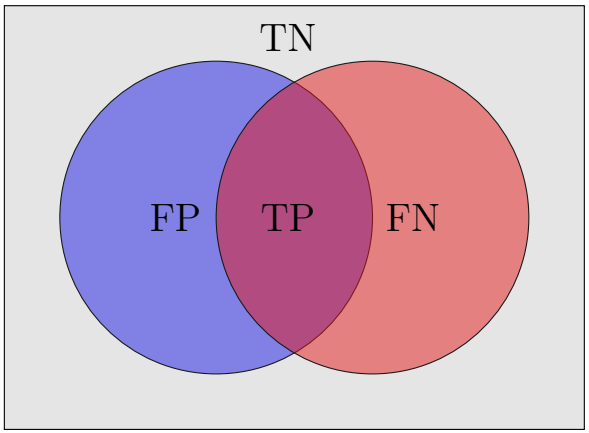
   1. n_l (the net input of class l) increases
   2. p_l (the probability of class l) increases
   3. $1 - p_l$ (the probability of the other classes) decreases

5. as a result, the bias against the other classes (the minority classes) arises

Metrics for Classification

- Unlike regression which uses Mean Squared Error to measure their performance, classification uses a different set of metrics.
- Here we will focus on three kinds of metrics:
  - Confusion Matrix
  - Accuracy
  - Precision, Recall and F1-score
- For illustration purposes, we will use binary classification (with classes Positive and Negative) to explain these measurements.

Confusion Matrix



Actual Class \ Predicted Class	$\widehat{\mathbf{P}}$	$\widehat{\mathbf{N}}$
P	True Positive (TP)	False Negative (FN)
N	False Positive (FP)	True Negative (TN)

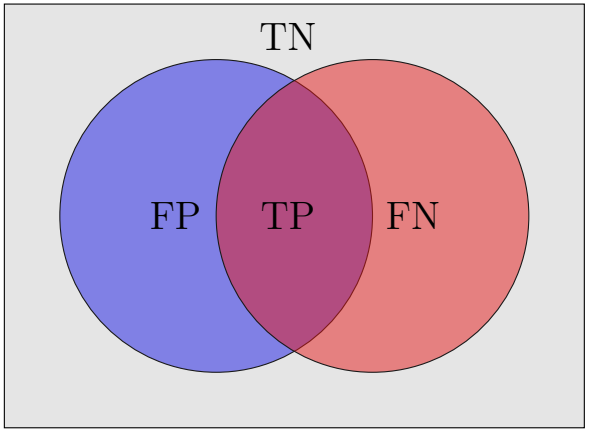
Figure 5: Venn diagram and confusion matrix showing the TP, FP, TN and FN.

- Based on the TP, FP, TN and FN in fig. 5, we have

$$\mathbf{P} = \mathrm{TP} + \mathrm{FN}, \quad \mathbf{N} = \mathrm{TN} + \mathrm{FP}, \quad \widehat{\mathbf{P}} = \mathrm{TP} + \mathrm{FP}, \quad \widehat{\mathbf{N}} = \mathrm{TN} + \mathrm{FN}. \tag{75}$$

- Given the confusion matrix, we can obtain the remaining two kinds of metrics:
  - accuracy
  - precision, recall and F1-score

Accuracy



		Predicted Class $\widehat{\mathbf{P}}$	Predicted Class $\widehat{\mathbf{N}}$
Actual Class	**P**	True Positive (TP)	False Negative (FN)
Actual Class	**N**	False Positive (FP)	True Negative (TN)

Figure 5: Venn diagram and confusion matrix showing the TP, FP, TN and FN.

- Based on the TP, FP, TN and FN in fig. 5, *Accuracy*, acc, is defined as follows:

$$acc = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{76}$$

The Problem of Accuracy

- It is recommended not to use accuracy to measure the performance of a classifier, particularly when the data is imbalanced.
- This can be seen from the definition of accuaracy, acc, given in eq. (76)

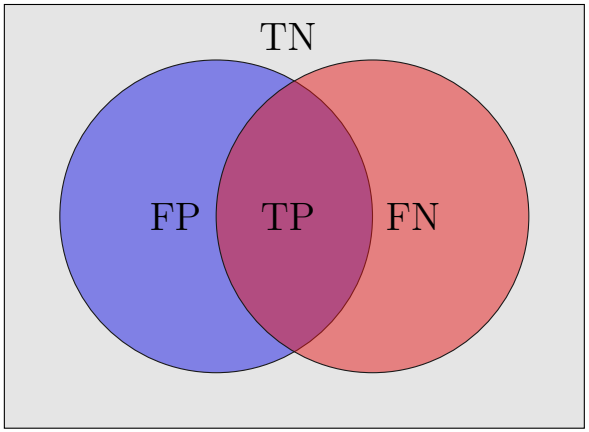
$$acc = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{76}$$

- Concretely, for imbalanced data where the number of the majority class is much higher than the number of the minority class:
  - a useless classifier that predicts every sample as the majority class will have almost perfect accuracy (i.e., close to 1)
  - this is because in eq. (76) TP (if positive is the majority class) or TN (if negative is the majority class) will be much larger than the other items or, in other words, the weight of the majority class will be much higher than that of the minority class
  - this results in an accuracy close to 1

Good practice

- It is recommended not to use accuracy to measure the performance of a classifier, particularly when the data is imbalanced.

Precision (Class-Specific)



		Predicted Class $\widehat{\mathbf{P}}$	Predicted Class $\widehat{\mathbf{N}}$
Actual Class	**P**	True Positive (TP)	False Negative (FN)
	N	False Positive (FP)	True Negative (TN)

Figure 5: Venn diagram and confusion matrix showing the TP, FP, TN and FN.

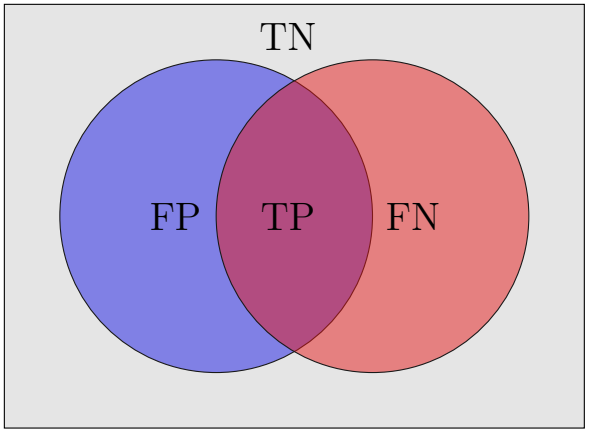
- Based on the TP, FP, TN and FN in fig. 5:
  - *Precision of the Positive Class*, p_1, is defined as follows:

$$p_1 = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{77}$$

  - *Precision of the Negative Class*, p_0, is defined as follows:

$$p_0 = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}} \tag{78}$$

Precision (Micro / Macro)



		Predicted Class $\widehat{\mathbf{P}}$	Predicted Class $\widehat{\mathbf{N}}$
Actual Class	**P**	True Positive (TP)	False Negative (FN)
	N	False Positive (FP)	True Negative (TN)

Figure 5: Venn diagram and confusion matrix showing the TP, FP, TN and FN.

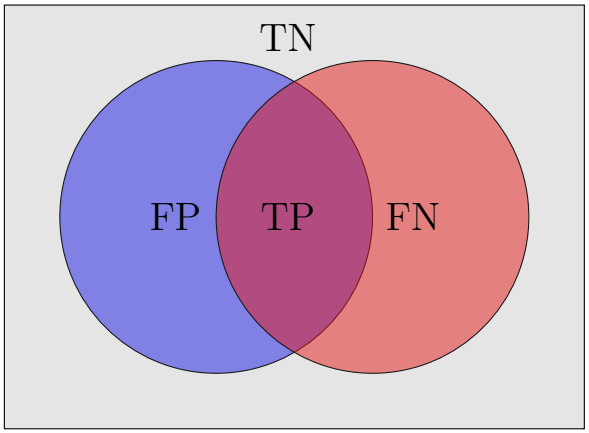
- Based on the TP, FP, TN and FN in fig. 5:
  - *Precision across the two classes (Micro)*, p_{micro}, is defined as follows:

$$p_{\text{micro}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{79}$$

    which is the same as accuracy, defined in eq. (76)
  - *Precision across the two classes (Macro)*, p_{macro}, is defined as follows:

$$p_{\text{macro}} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FP}} + \frac{\text{TN}}{\text{TN} + \text{FN}}\right) \tag{80}$$

Recall (Class-Specific)



		Predicted Class $\widehat{\mathbf{P}}$	Predicted Class $\widehat{\mathbf{N}}$
Actual Class	**P**	True Positive (TP)	False Negative (FN)
	N	False Positive (FP)	True Negative (TN)

Figure 5: Venn diagram and confusion matrix showing the TP, FP, TN and FN.

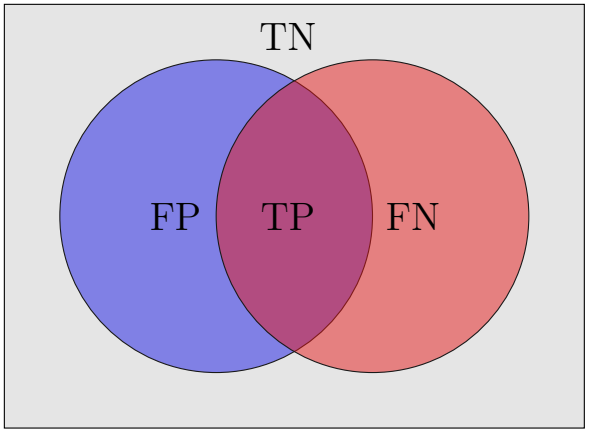
- Based on the TP, FP, TN and FN in fig. 5:
  - *Recall of the Positive Class*, r_1, is defined as follows:

$$r_1 = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{81}$$

  - *Recall of the Negative Class*, r_0, is defined as follows:

$$r_0 = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{82}$$

Recall (Micro / Macro)



		Predicted Class $\widehat{\mathbf{P}}$	Predicted Class $\widehat{\mathbf{N}}$
Actual Class	**P**	True Positive (TP)	False Negative (FN)
Actual Class	**N**	False Positive (FP)	True Negative (TN)

Figure 5: Venn diagram and confusion matrix showing the TP, FP, TN and FN.

- Based on the TP, FP, TN and FN in fig. 5:
  - *Recall across the two classes (Micro)*, r_{micro}, is defined as follows:

$$r_{\text{micro}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{83}$$

    which is the same as acc and p_{micro}, defined in eqs. (76) and (79)
  - *Recall across the two classes (Macro)*, r_{macro}, is defined as follows:

$$r_{\text{macro}} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}}\right) \tag{84}$$

F1-Score (Class-Specific)

- *F1-score of the Positive Class*, f_1, is defined as follows:

$$f_1 = \frac{2p_1 r_1}{p_1 + r_1}, \tag{85}$$

where p_1 and r_1 are the precision and recall of the positive class, given in eqs. (77) and (81):

$$p_1 = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \tag{77}$$

$$r_1 = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{81}$$

- *F1-score of the Negative Class*, f_0, is defined as follows:

$$f_0 = \frac{2p_0 r_0}{p_0 + r_0}, \tag{86}$$

where p_0 and r_0 are the precision and recall of the negative class, given in eqs. (78) and (82):

$$p_0 = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}}, \tag{80}$$

$$r_0 = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}. \tag{82}$$

F1-Score (Micro / Macro)

- *F1-score across the two Classes (Micro)*, f_{micro}, is defined as follows:

$$f_{\text{micro}} = \frac{2p_{\text{micro}}r_{\text{micro}}}{p_{\text{micro}} + r_{\text{micro}}}, \tag{87}$$

where p_{micro} and r_{micro} are the precision and recall across the two classes (micro), given in eqs. (79) and (83):

$$p_{\text{micro}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{79}$$

$$r_{\text{micro}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}. \tag{83}$$

- *F1-score across the two Classes (Macro)*, f_{macro}, is defined as follows:

$$f_{\text{macro}} = \frac{2p_{\text{macro}}r_{\text{macro}}}{p_{\text{macro}} + r_{\text{macro}}}, \tag{88}$$

where p_{macro} and r_{macro} are the precision and recall across the two classes (macro), given in eqs. (80) and (84):

$$p_{\text{macro}} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FP}} + \frac{\text{TN}}{\text{TN} + \text{FN}}\right), \tag{80}$$

$$r_{\text{macro}} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}}\right). \tag{84}$$

Accuracy, Precision, Recall and F1-Score

- It is recommended to report the class-specific precision, recall and F1-score, rather than the accuracy, particularly when the data is imbalanced, since:
  - class-specific metrics allow us to see the classification error in each class
  - accuracy is a summarized metric across all the classes, and gives higher weights to the majority class (leading to a compromised measurement of the classifier)
- If we have to report a summarized score across all the classes, we usually favor macro over micro, particularly when the data is imbalanced, since:
  - micro reduces to accuracy, which gives higher weights to the majority class (leading to a compromised measurement of the classifier)
  - macro gives the same weight to each class, which helps addressing the problem of accuracy and micro

Good practice

- It is recommended to report the class-specific precision, recall and F1-score, rather than the accuracy, particularly when the data is imbalanced.
- If we have to report a summarized score across all the classes, we usually favor macro over micro, particularly when the data is imbalanced.

Precision, Recall and F1-Score: Code

- See /utilities/p2_shallow_learning:
  1. cell 10

Proof of KL-divergence Minimum: Page 19

- KL-divergence from $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)$ to $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$, $D_{KL}\Big(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\Big)$, is given in eq. (15)

$$D_{KL}\Big(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\Big) = E_{y\sim\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}\left[\log\frac{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})}\right]. \tag{15}$$

- When y is discrete, we can write eq. (15) as

$$D_{KL}\Big(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\Big) = \sum_y \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)\log\frac{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})}. \tag{89}$$

- Since for each $x \geq 0$ we have

$$-\log x \geq 1 - x, \tag{90}$$

where the equality holds when $x = 1$, we can write eq. (89) as

$$\begin{aligned} D_{KL}\Big(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\Big) &= \sum_y \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)\log\frac{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})}, \\ &= -\sum_y \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)\log\frac{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})}{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}, \\ &\geq -\sum_y \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)\left(1 - \frac{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})}{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}\right) = -\sum_y \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*) - \sum_y \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = 0, \end{aligned} \tag{91}$$

where the equality holds (i.e., KL-divergence takes the minimum value, 0) when $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*) = \mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$, which proves the claim in eq. (16) on page 19.

Proof of Minimizing KL-divergence Equates MLE: Page 19

- KL-divergence from $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)$ to $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$, $D_{KL}\left(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\right)$, is given in eq. (15)

$$D_{KL}\left(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\right) = E_{y\sim\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}\left[\log\frac{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})}\right]. \tag{15}$$

- We can write eq. (15) as

$$\begin{aligned} D_{KL}\left(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\right) &= E_{y\sim\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}\left[\log\frac{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}{\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})}\right], \\ &= E_{y\sim\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}\left[\log\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)\right] - E_{y\sim\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}\left[\log\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\right]. \end{aligned} \tag{92}$$

- Since the second item in eq. (92), $E_{y\sim\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}\left[\log\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)\right]$, is constant, minimizing the KL-divergence (first item in eq. (92)), $D_{KL}\left(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\right)$, equates maximizing the third item in the equation, $E_{y\sim\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}\left[\log\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\right]$ and, in turn, maximizing the likelihood, $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$:

$$\arg\min_{\boldsymbol{\theta}} D_{KL}\left(\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)||\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\right) = \arg\max_{\boldsymbol{\theta}} E_{y\sim\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}^*)}\left[\log\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})\right] = \arg\max_{\boldsymbol{\theta}}\mathbf{p}(\mathbf{y}|\boldsymbol{\theta}), \tag{93}$$

which proves the claim on page 19.

Proof of Binomial Cross Entropy: Page 20

- As discussed earlier, we can use MLE (eq. (14)) to estimate the parameters, $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \binom{m}{k} \prod_{i=1}^{m} \left(p^i\right)^{y^i} \left(1 - p^i\right)^{1-y^i}. \tag{14}$$

- Since $\binom{m}{k}$ is a constant with respect to $\boldsymbol{\theta}$, we can write eq. (14) as

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} \left(p^i\right)^{y^i} \left(1 - p^i\right)^{1-y^i}. \tag{94}$$

- Since maximizing $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$ equates maximizing $\log \mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$, we can write eq. (94) as

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \log \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log \left(\prod_{i=1}^{m} \left(p^i\right)^{y^i} \left(1 - p^i\right)^{1-y^i} \right). \tag{95}$$

Proof of Binomial Cross Entropy: Page 20

- Based on the following law of logarithms:

$$\log ab = \log a + \log b, \tag{96}$$

we can write eq. (95) as

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \log \left(\prod_{i=1}^{m} \left(p^i\right)^{y^i} \left(1-p^i\right)^{1-y^i} \right), \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log \left(\left(p^i\right)^{y^i} \left(1-p^i\right)^{1-y^i} \right), \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \left(\log \left(p^i\right)^{y^i} + \log \left(1-p^i\right)^{1-y^i} \right).
\end{aligned} \tag{97}$$

Proof of Binomial Cross Entropy: Page 20

- Based on the following law of logarithms:

$$\log a^b = b \log a, \tag{98}$$

we can write eq. (97) as

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \left(\log \left(p^i \right)^{y^i} + \log \left(1 - p^i \right)^{1-y^i} \right), \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \left(y^i \log \left(p^i \right) + (1 - y^i) \log \left(1 - p^i \right) \right), \\
&= \arg\min_{\boldsymbol{\theta}} \left(- \sum_{i=1}^{m} \left(y^i \log \left(p^i \right) + (1 - y^i) \log \left(1 - p^i \right) \right) \right),
\end{aligned} \tag{99}$$

which proves the claim in eq. (17) on page 20.

Proof of the Updating Rule: Page 25

- We can write the gradient (i.e., first-order derivative) of the loss function with respect to mini-batch $\mathbf{mb}^j$, $\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal$, as

$$\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal = \left[\frac{\partial}{\partial b} \mathcal{L}(\boldsymbol{\theta}^j) \quad \frac{\partial}{\partial w_1} \mathcal{L}(\boldsymbol{\theta}^j) \cdots \frac{\partial}{\partial w_n} \mathcal{L}(\boldsymbol{\theta}^j) \right]^\intercal, \tag{100}$$

where the loss function, $\mathcal{L}(\boldsymbol{\theta}^j)$, is given in eq. (22)

$$\mathcal{L}(\boldsymbol{\theta}^j) = -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(y^i \log\left(p^i\right) + (1 - y^i) \log\left(1 - p^i\right) \right). \tag{22}$$

Proof of the Updating Rule: Page 25

- Based on eq. (22), we can write $\frac{\partial}{\partial b}\mathcal{L}(\boldsymbol{\theta}^j)$ as

$$\begin{aligned}\frac{\partial}{\partial b}\mathcal{L}(\boldsymbol{\theta}^j) &= -\frac{\partial}{\partial b}\left(\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i\log\left(p^i\right)+(1-y^i)\log\left(1-p^i\right)\right)\right),\\ &= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i\frac{\partial\log\left(p^i\right)}{\partial b}+(1-y^i)\frac{\partial\log\left(1-p^i\right)}{\partial b}\right).\end{aligned} \tag{101}$$

- We now derive the two partial derivatives in eq. (101), $\frac{\partial\log(p^i)}{\partial b}$ and $\frac{\partial\log(1-p^i)}{\partial b}$.
- Based on the derivative rule of Logarithms, we can write the two partial derivatives as

$$\frac{\partial\log\left(p^i\right)}{\partial b} = \frac{1}{p^i}\frac{\partial p^i}{\partial b}, \tag{102}$$

$$\frac{\partial\log\left(1-p^i\right)}{\partial b} = \frac{1}{1-p^i}\frac{\partial}{\partial b}\left(1-p^i\right) = -\frac{1}{1-p^i}\frac{\partial p^i}{\partial b}. \tag{103}$$

- We now derive the common item in eqs. (102) and (103), $\frac{\partial p^i}{\partial b}$.

Proof of the Updating Rule: Page 25

- Based on eq. (25)

$$p^i = \frac{1}{1+e^{-n^i}} \quad \text{where} \quad n^i = b + w_1 x_1^i + \cdots + w_n x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta}^j, \tag{25}$$

we can write $\frac{\partial p^i}{\partial b}$ in eqs. (102) and (103) as

$$\begin{aligned} \frac{\partial \left(p^i\right)}{\partial b} &= \frac{\partial \frac{1}{1+e^{-n^i}}}{\partial b}, \\ &= -\frac{1}{\left(1+e^{-n^i}\right)^2} \frac{\partial}{\partial b}\left(1+e^{-n^i}\right), \\ &= \frac{e^{-n^i}}{\left(1+e^{-n^i}\right)^2} \frac{\partial n^i}{\partial b}. \end{aligned} \tag{104}$$

Proof of the Updating Rule: Page 25

- Based on eq. (25), we can write eq. (104) as

$$
\begin{aligned}
\frac{\partial\left(p^i\right)}{\partial b} &= \frac{e^{-n^i}}{\left(1+e^{-n^i}\right)^2}\frac{\partial n^i}{\partial b}, \\
&= \frac{e^{-n^i}}{\left(1+e^{-n^i}\right)^2}\frac{\partial}{\partial b}\left(b+w_1x_1^i+\cdots+w_nx_n^i\right), \\
&= \frac{e^{-n^i}}{\left(1+e^{-n^i}\right)^2}, \\
&= \left(\frac{1+e^{-n^i}}{\left(1+e^{-n^i}\right)^2}-\frac{1}{\left(1+e^{-n^i}\right)^2}\right), \\
&= \left(p^i-\left(p^i\right)^2\right), \\
&= p^i\left(1-p^i\right).
\end{aligned}
\tag{105}
$$

Proof of the Updating Rule: Page 25

- By substituting eq. (105) into eqs. (102) and (103), we have

$$\frac{\partial \log\left(p^i\right)}{\partial b} = \frac{1}{p^i}\frac{\partial p^i}{\partial b} = \frac{1}{p^i}p^i\left(1-p^i\right) = 1-p^i, \tag{106}$$

$$\frac{\partial \log\left(1-p^i\right)}{\partial b} = -\frac{1}{1-p^i}\frac{\partial p^i}{\partial b} = -\frac{1}{1-p^i}\left(1-p^i\right)p^i = -p^i. \tag{107}$$

- By substituting eqs. (106) and (107) into eq. (101), we have

$$\begin{aligned}
\frac{\partial}{\partial b}\mathcal{L}(\boldsymbol{\theta}^j) &= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i\frac{\partial \log\left(p^i\right)}{\partial b} + (1-y^i)\frac{\partial \log\left(1-p^i\right)}{\partial b}\right), \\
&= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i\left(1-p^i\right) + (1-y^i)\left(-p^i\right)\right), \\
&= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i - y^ip^i - p^i + y^ip^i\right), \\
&= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i - p^i\right).
\end{aligned} \tag{108}$$

Proof of the Updating Rule: Page 25

- Based on eq. (22), we can write $\frac{\partial}{\partial w_j}\mathcal{L}(\boldsymbol{\theta}^j)$ as

$$\begin{aligned}\frac{\partial}{\partial w_j}\mathcal{L}(\boldsymbol{\theta}^j) &= -\frac{\partial}{\partial w_j}\left(\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i\log\left(p^i\right)+(1-y^i)\log\left(1-p^i\right)\right)\right),\\ &= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i\frac{\partial\log\left(p^i\right)}{\partial w_j}+(1-y^i)\frac{\partial\log\left(1-p^i\right)}{\partial w_j}\right).\end{aligned} \tag{109}$$

- We now derive the two partial derivatives in eq. (109), $\frac{\partial\log(p^i)}{\partial w_j}$ and $\frac{\partial\log(1-p^i)}{\partial w_j}$.
- Based on the derivative rule of Logarithms, the two partial derivaties can be written as

$$\frac{\partial\log\left(p^i\right)}{\partial w_j} = \frac{1}{p^i}\frac{\partial p^i}{\partial w_j}, \tag{110}$$

$$\frac{\partial\log\left(1-p^i\right)}{\partial w_j} = \frac{1}{1-p^i}\frac{\partial}{\partial w_j}\left(1-p^i\right) = -\frac{1}{1-p^i}\frac{\partial p^i}{\partial w_j}. \tag{111}$$

- We now derive the common item in eqs. (110) and (111), $\frac{\partial p^i}{\partial w_j}$.

Proof of the Updating Rule: Page 25

- Based on eq. (25)

$$p^i = \frac{1}{1+e^{-n^i}} \quad \text{where} \quad n^i = b + w_1 x_1^i + \cdots + w_n x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta}^j, \tag{25}$$

we can write $\frac{\partial p^i}{\partial w_j}$ in eqs. (110) and (111) as

$$\begin{aligned} \frac{\partial \left(p^i\right)}{\partial w_j} &= \frac{\partial \frac{1}{1+e^{-n^i}}}{\partial w_j}, \\ &= -\frac{1}{\left(1+e^{-n^i}\right)^2} \frac{\partial}{\partial w_j}\left(1+e^{-n^i}\right), \\ &= \frac{e^{-n^i}}{\left(1+e^{-n^i}\right)^2} \frac{\partial n^i}{\partial w_j}. \end{aligned} \tag{112}$$

Proof of the Updating Rule: Page 25

- Based on eq. (25), we can write eq. (112) as

$$
\begin{aligned}
\frac{\partial\left(p^i\right)}{\partial w_j} &= \frac{e^{-n^i}}{\left(1+e^{-n^i}\right)^2}\frac{\partial n^i}{\partial w_j}, \\
&= \frac{e^{-n^i}}{\left(1+e^{-n^i}\right)^2}\frac{\partial}{\partial w_j}\left(b+w_1x_1^i+\cdots+w_nx_n^i\right), \\
&= \left(\frac{e^{-n^i}}{\left(1+e^{-n^i}\right)^2}\right)x_j^i, \\
&= \left(\frac{1+e^{-n^i}}{\left(1+e^{-n^i}\right)^2}-\frac{1}{\left(1+e^{-n^i}\right)^2}\right)x_j^i, \\
&= \left(p^i-\left(p^i\right)^2\right)x_j^i, \\
&= p^i\left(1-p^i\right)x_j^i.
\end{aligned}
\tag{113}
$$

Proof of the Updating Rule: Page 25

- By substituting eq. (113) into eqs. (110) and (111), we have

$$\frac{\partial \log\left(p^i\right)}{\partial w_j} = \frac{1}{p^i}\frac{\partial p^i}{\partial w_j} = \frac{1}{p^i}p^i\left(1-p^i\right)x_j^i = \left(1-p^i\right)x_j^i, \tag{114}$$

$$\frac{\partial \log\left(1-p^i\right)}{\partial w_j} = -\frac{1}{1-p^i}\frac{\partial p^i}{\partial w_j} = -\frac{1}{1-p^i}\left(1-p^i\right)p^i x_j^i = -p^i x_j^i. \tag{115}$$

- By substituting eqs. (114) and (115) into eq. (109), we have

$$\begin{aligned}
\frac{\partial}{\partial w_j}\mathcal{L}(\boldsymbol{\theta}^j) &= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i\frac{\partial \log\left(p^i\right)}{\partial w_j} + (1-y^i)\frac{\partial \log\left(1-p^i\right)}{\partial w_j}\right), \\
&= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i\left(1-p^i\right) + (1-y^i)\left(-p^i\right)\right)x_j^i, \\
&= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i - y^i p^i - p^i + y^i p^i\right)x_j^i, \\
&= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i - p^i\right)x_j^i.
\end{aligned} \tag{116}$$

Proof of the Updating Rule: Page 25

- Based on eqs. (108) and (116), we can write eq. (100) as

$$\begin{aligned}\nabla\mathcal{L}(\boldsymbol{\theta}^j)^\intercal &= \left[\frac{\partial}{\partial b}\mathcal{L}(\boldsymbol{\theta}^j) \quad \frac{\partial}{\partial w_1}\mathcal{L}(\boldsymbol{\theta}^j)\cdots\frac{\partial}{\partial w_n}\mathcal{L}(\boldsymbol{\theta}^j)\right]^\intercal, \\ &= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i - p^i\right)\left[1 \quad x_1^i\cdots x_n^i\right]^\intercal, \\ &= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i - p^i\right)\left[1 \quad \mathbf{x}^i\right]^\intercal.\end{aligned} \tag{117}$$

- By substituting eq. (117) into eq. (23), we have

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \frac{\eta_k}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i - p^i\right)\left[1 \quad \mathbf{x}^i\right]^\intercal = \boldsymbol{\theta}_k^j + \frac{\eta_k}{|\mathbf{mb}^j|}\left[\mathbf{1} \quad \mathbf{X}^j\right]^\intercal(\mathbf{y}^j - \mathbf{p}^j), \tag{118}$$

which proves the claim in eq. (24) on page 25. □

Proof of Multinomial Cross Entropy: Page 46

- As discussed earlier, we can use MLE (eq. (52)) to estimate the parameters, $\boldsymbol{\theta} = \begin{bmatrix} \mathbf{b} & \mathbf{w}_1 \cdots \mathbf{w}_n \end{bmatrix}^\mathsf{T}$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \binom{n}{n_1, \ldots, n_c} \prod_{i=1}^{m} \prod_{k=1}^{c} \left(p_k^i\right)^{y_k^i}. \tag{52}$$

- Since $\binom{n}{n_1,\ldots,n_c}$ is a constant with respect to $\boldsymbol{\theta}$, we can write eq. (52) as

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} \prod_{k=1}^{c} \left(p_k^i\right)^{y_k^i}. \tag{119}$$

- Since maximizing $\mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$ equates maximizing $\log \mathbf{p}(\mathbf{y}|\boldsymbol{\theta})$, we can write eq. (119) as

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \log \mathbf{p}(\mathbf{y}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log \left(\prod_{i=1}^{m} \prod_{k=1}^{c} \left(p_k^i\right)^{y_k^i} \right). \tag{120}$$

Proof of Multinomial Cross Entropy: Page 46

- Based on the law of logarithms in eq. (96):

$$\log ab = \log a + \log b, \tag{96}$$

we can write eq. (120) as

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \log \left(\prod_{i=1}^{m} \prod_{k=1}^{c} \left(p_k^i \right)^{y_k^i} \right), \\ &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \sum_{k=1}^{c} \log \left(\left(p_k^i \right)^{y_k^i} \right). \end{aligned} \tag{121}$$

Proof of Multinomial Cross Entropy: Page 46

- Based on the law of logarithms in eq. (98):

$$\log a^b = b \log a, \tag{98}$$

we can write eq. (121) as

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \sum_{k=1}^{c} \log \left(\left(p_k^i \right)^{y_k^i} \right), \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \sum_{k=1}^{c} y_k^i \log \left(p_k^i \right), \\
&= \arg\min_{\boldsymbol{\theta}} \left(- \sum_{i=1}^{m} \sum_{k=1}^{c} y_k^i \log \left(p_k^i \right) \right),
\end{aligned} \tag{122}$$

which proves the claim in eq. (53) on page 46.

Proof of the Updating Rule: Page 51

- The gradient (i.e., first-order derivative) of the loss function with respect to mini-batch $\mathbf{mb}^j$ and class k, $\nabla \mathcal{L}^k(\boldsymbol{\theta}^j)^\intercal$, can be written as

$$\nabla \mathcal{L}^k(\boldsymbol{\theta}^j)^\intercal = \left[\frac{\partial}{\partial b^k} \mathcal{L}(\boldsymbol{\theta}^j) \quad \frac{\partial}{\partial w_1^k} \mathcal{L}(\boldsymbol{\theta}^j) \cdots \frac{\partial}{\partial w_n^k} \mathcal{L}(\boldsymbol{\theta}^j) \right]^\intercal, \tag{123}$$

  where the loss function, $\mathcal{L}(\boldsymbol{\theta}^j)$, is given in eq. (56)

$$\mathcal{L}(\boldsymbol{\theta}^j) = -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \sum_{l=1}^{c} y_l^i \log\left(p_l^i\right). \tag{56}$$

Proof of the Updating Rule: Page 51

- Based on eq. (56), we can write $\frac{\partial}{\partial b^k}\mathcal{L}(\boldsymbol{\theta}^j)$ as

$$\begin{aligned}\frac{\partial}{\partial b^k}\mathcal{L}(\boldsymbol{\theta}^j) &= -\frac{\partial}{\partial b^k}\left(\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\sum_{l=1}^{c} y_l^i \log\left(p_l^i\right)\right),\\ &= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\sum_{l=1}^{c} y_l^i \frac{\partial \log\left(p_l^i\right)}{\partial b^k}, \qquad (124)\\ &= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(\sum_{l\neq k} y_l^i \frac{\partial \log\left(p_l^i\right)}{\partial b^k} + y_k^i \frac{\partial \log\left(p_k^i\right)}{\partial b^k}\right).\end{aligned}$$

- Based on the derivative rule of Logarithms, we can write eq. (124) as

$$\frac{\partial}{\partial b^k}\mathcal{L}(\boldsymbol{\theta}^j) = -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(\sum_{l\neq k}\frac{y_l^i}{p_l^i}\frac{\partial\left(p_l^i\right)}{\partial b^k} + \frac{y_k^i}{p_k^i}\frac{\partial\left(p_k^i\right)}{\partial b^k}\right). \qquad (125)$$

Proof of the Updating Rule: Page 51

- Based on eqs. (50) and (51)

$$p_q^i = \frac{e^{n_q^i}}{\sum_{m=1}^c e^{n_m^i}}, \quad \text{where} \quad n_q^i = b^q + w_1^q x_1^i + \cdots + w_n^q x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta}_q^j, \tag{50,51}$$

we can write eq. (125) as

$$\begin{aligned}
\frac{\partial}{\partial b^k} \mathcal{L}(\boldsymbol{\theta}^j) &= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(\sum_{l \neq k} \frac{y_l^i}{p_l^i} \frac{\partial \left(p_l^i \right)}{\partial b^k} + \frac{y_k^i}{p_k^i} \frac{\partial \left(p_k^i \right)}{\partial b^k} \right), \\
&= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(\sum_{l \neq k} \frac{y_l^i}{p_l^i} \frac{\partial}{\partial b^k} \frac{e^{n_l^i}}{\sum_{m=1}^c e^{n_m^i}} + \frac{y_k^i}{p_k^i} \frac{\partial}{\partial b^k} \frac{e^{n_k^i}}{\sum_{m=1}^c e^{n_m^i}} \right), \\
&= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(-\sum_{l \neq k} \frac{y_l^i}{p_l^i} \frac{e^{n_l^i} e^{n_k^i}}{\left(\sum_{m=1}^c e^{n_m^i} \right)^2} + \frac{y_k^i}{p_k^i} \left(\frac{e^{n_k^i}}{\sum_{m=1}^c e^{n_m^i}} - \left(\frac{e^{n_k^i}}{\sum_{m=1}^c e^{n_m^i}} \right)^2 \right) \right) \frac{\partial n_k^i}{\partial b^k}, \\
&= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(-\sum_{l \neq k} \frac{y_l^i}{p_l^i} p_l^i p_k^i + \frac{y_k^i}{p_k^i} \left(p_k^i - \left(p_k^i \right)^2 \right) \right), \\
&= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(-\sum_{l \neq k} y_l^i p_k^i + y_k^i \left(1 - p_k^i \right) \right), \\
&= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(y_k^i - p_k^i \left(\sum_{l \neq k} y_l^i + y_k^i \right) \right), \\
&= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(y_k^i - p_k^i \right).
\end{aligned} \tag{126}$$

Proof of the Updating Rule: Page 51

- Based on eq. (56), we can write $\frac{\partial}{\partial w_j^k}\mathcal{L}(\boldsymbol{\theta}^j)$ as

$$\begin{aligned}\frac{\partial}{\partial w_j^k}\mathcal{L}(\boldsymbol{\theta}^j) &= -\frac{\partial}{\partial w_j^k}\left(\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\sum_{l=1}^{c} y_l^i \log\left(p_l^i\right)\right),\\ &= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\sum_{l=1}^{c} y_l^i \frac{\partial \log\left(p_l^i\right)}{\partial w_j^k}, \qquad (127)\\ &= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(\sum_{l\neq k} y_l^i \frac{\partial \log\left(p_l^i\right)}{\partial w_j^k} + y_k^i \frac{\partial \log\left(p_k^i\right)}{\partial w_j^k}\right).\end{aligned}$$

- Based on the derivative rule of Logarithms, we can write eq. (127) as

$$\frac{\partial}{\partial w_j^k}\mathcal{L}(\boldsymbol{\theta}^j) = -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(\sum_{l\neq k}\frac{y_l^i}{p_l^i}\frac{\partial\left(p_l^i\right)}{\partial w_j^k} + \frac{y_k^i}{p_k^i}\frac{\partial\left(p_k^i\right)}{\partial w_j^k}\right). \qquad (128)$$

Proof of the Updating Rule: Page 51

- Based on eqs. (50) and (51)

$$p_q^i = \frac{e^{n_q^i}}{\sum_{m=1}^c e^{n_m^i}}, \quad \text{where} \quad n_q^i = b^q + w_1^q x_1^i + \cdots + w_n^q x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta}_q^j, \tag{50,51}$$

we can write eq. (128) as

$$\begin{aligned}
\frac{\partial}{\partial w_j^k} \mathcal{L}(\boldsymbol{\theta}^j) &= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(\sum_{l \neq k} \frac{y_l^i}{p_l^i} \frac{\partial \left(p_l^i \right)}{\partial w_j^k} + \frac{y_k^i}{p_k^i} \frac{\partial \left(p_k^i \right)}{\partial w_j^k} \right), \\
&= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(\sum_{l \neq k} \frac{y_l^i}{p_l^i} \frac{\partial}{\partial w_j^k} \frac{e^{n_l^i}}{\sum_{m=1}^c e^{n_m^i}} + \frac{y_k^i}{p_k^i} \frac{\partial}{\partial w_j^k} \frac{e^{n_k^i}}{\sum_{m=1}^c e^{n_m^i}} \right), \\
&= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(-\sum_{l \neq k} \frac{y_l^i}{p_l^i} \frac{e^{n_l^i} e^{n_k^i}}{\left(\sum_{m=1}^c e^{n_m^i} \right)^2} + \frac{y_k^i}{p_k^i} \left(\frac{e^{n_k^i}}{\sum_{m=1}^c e^{n_m^i}} - \left(\frac{e^{n_k^i}}{\sum_{m=1}^c e^{n_m^i}} \right)^2 \right) \right) \frac{\partial n_k^i}{\partial w_j^k}, \\
&= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(-\sum_{l \neq k} \frac{y_l^i}{p_l^i} p_l^i p_k^i + \frac{y_k^i}{p_k^i} \left(p_k^i - \left(p_k^i \right)^2 \right) \right) x_j^i, \\
&= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(-\sum_{l \neq k} y_l^i p_k^i + y_k^i \left(1 - p_k^i \right) \right) x_j^i, \\
&= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(y_k^i - p_k^i \left(\sum_{l \neq k} y_l^i + y_k^i \right) \right) x_j^i, \\
&= -\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(y_k^i - p_k^i \right) x_j^i.
\end{aligned} \tag{129}$$

Proof of the Updating Rule: Page 51

- Based on eqs. (126) and (129), we can write eq. (123) as

$$\begin{aligned}\nabla \mathcal{L}^k(\boldsymbol{\theta}^j)^\intercal &= \left[\frac{\partial}{\partial b^k}\mathcal{L}(\boldsymbol{\theta}^j) \quad \frac{\partial}{\partial w_1^k}\mathcal{L}(\boldsymbol{\theta}^j) \cdots \frac{\partial}{\partial w_n^k}\mathcal{L}(\boldsymbol{\theta}^j)\right]^\intercal, \\ &= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y_k^i - p_k^i\right)\left[1 \quad x_1^i \cdots x_n^i\right]^\intercal, \\ &= -\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y_k^i - p_k^i\right)\left[1 \quad \mathbf{x}^i\right]^\intercal.\end{aligned} \tag{130}$$

- By substituting eq. (130) into eq. (23), we have

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \frac{\eta_k}{|\mathbf{mb}^j|}\left[\mathbf{1} \quad \mathbf{X}^j\right]^\intercal(\mathbf{Y}^j - \mathbf{P}^j), \tag{131}$$

which proves the claim in eq. (57) on page 51. □

Bibliography I

Masters, D. and Luschi, C. (2018).
Revisiting Small Batch Training for Deep Neural Networks.
arXiv preprint arXiv:1804.07612.