Machine Learning

|Softmax Regression – Implementation

Prof. Dr. Mostafa Elhosseini

https://youtube.com/ElhosseiniAcademy

Agenda

- Numerical Examples
- Iris Dataset Implementation

Example 1

- $K = 3$ #classes = 3.
- $m = 2$ # training examples.
- let's assume one-hot encoded true labels
- For the **first** example, the true class is class 1. $y^{(1)} = [1,0,0]$
- For the second example, the true class is class 3. $y^{(2)} = [0,0,1]$
- Assume the predicted probabilities for the first example
- $\hat{y}^{(1)} = [0.7,0.2,0.1]$ $\hat{y}^{(2)} = [0.1,0.3,0.6]$

Example 1

- $L^{(i)} = -\sum_{k=1}^{K} y_k{}^{(i)} \log \hat{p}_k{}^{(i)}$
- For the first Example
- $y^{(1)} = [1,0,0]$ $\hat{y}^{(1)} = [0.7,0.2,0.1]$
- $L^{(1)} = -(1.\log(0.7) + 0.\log(0.2) + 0.\log(0.1)) =$
  $-\log 0.7 = 0.3567$
- For the second Example
- $y^{(2)} = [0,0,1]$ $\hat{y}^{(2)} = [0.1,0.3,0.6]$
- $L^{(2)} = -(0.\log(0.1) + 0.\log(0.3) + 1.\log(0.6)) =$
  $-\log 0.6 = 0.5108$

Example 1

- $L^{(1)} = 0.3567$
- $L^{(2)} = 0.5108$
- Average Cross-Entropy Loss
- $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} L^{(i)} = \frac{1}{2}(0.3567 + 0.5108) = 0.43375$

Example 2

- $K = 3$ #classes = 3.
- $m = 2$ # training examples.
- let's assume one-hot encoded true labels
- For the **first** example, the true class is class 2. $y^{(1)} = [0,1,0]$
- For the **second** example, the true class is class 3. $y^{(2)} = [0,0,1]$
- Assume the predicted probabilities for the first example
- $\hat{y}^{(1)} = [0.2,0.7,0.1]$ $\hat{y}^{(2)} = [0.3,0.4,0.3]$

Example 2

- $L^{(i)} = -\sum_{k=1}^{K} y_k{}^{(i)} \log \hat{p}_k{}^{(i)}$
- For the first Example
- $y^{(1)} = [0,1,0]$ $\hat{y}^{(1)} = [0.2,0.7,0.1]$
- $L^{(1)} = -(0.\log(0.2) + 1.\log(0.7) + 0.\log(0.1)) =$
  $-\log 0.7 = 0.3567$
- For the second Example
- $y^{(2)} = [0,0,1]$ $\hat{y}^{(2)} = [0.3,0.4,0.3]$
- $L^{(2)} = -(0.\log(0.3) + 0.\log(0.4) + 1.\log(0.3)) =$
  $-\log 0.3 = 1.204$

Example 2

- $L^{(1)} = 0.3567$
- $L^{(2)} = 1.204$
- Average Cross-Entropy Loss
- $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} L^{(i)} = \frac{1}{2}(0.3567 + 1.204) = 0.78035$

Example 3

- $K = 3$ # classes.
- $n = 2$ # Features
- Learning Rate: $\alpha = 0.1$
- Single Training Example $x = [1,2]$
- the true label $y = [1,0,0]$
- $\theta_1 = [0.1,0.2]$ $\theta_2 = [0,-0.1]$ $\theta_3 = [0.1,0.1]$

Example 3

- Compute Initial Logits: Compute the logits for each class using the initial weights:
- $z_1 = \boldsymbol{\theta_1}^T \boldsymbol{x} = 0.1 \times 1 + 0.2 \times 2 = 0.5$
- $z_2 = \boldsymbol{\theta_2}^T \boldsymbol{x} = 0.0 \times 1 + (-0.1) \times 2 = -0.2$
- $z_3 = \boldsymbol{\theta_3}^T \boldsymbol{x} = 0.1 \times 1 + 0.1 \times 2 = 0.3$

- The logits are $z = [0.5, -0.2, 0.3]$

Example 3

- Apply Softmax to Compute Probabilities $\hat{p}_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$
- $e^{z_1} = e^{0.5} = 1.6487$ $\quad e^{z_2} = e^{-0.2} = 0.8187$
- $e^{z_3} = e^{0.3} = 1.3499$
- $\sum_{j=1}^{3} e^{z_j} = 1.6487 + 0.8187 + 1.3499 = 3.8173$
- $\hat{p}_1 = \frac{1.6487}{3.8173} = 0.4319$ $\quad \hat{p}_2 = 0.2144$ $\quad \hat{p}_3 = 0.3537$

Example 3

- Compute Initial Cross-Entropy Loss $L = -\sum_{k=1}^{K} y_k \log \hat{p}_k$
- $L = -(1 \cdot \log(0.4319) + 0 \cdot \log(0.2144) + 0 \cdot \log(0.3537)) = 0.8392$
- Compute the gradient of the loss with respect to each weight:
- $\frac{\partial L}{\partial \boldsymbol{\theta}_{1,j}} = (\hat{p}_1 - y_1)x_j = (0.4319 - 1)x_j = -0.5681x_j$
- $\frac{\partial L}{\partial \boldsymbol{\theta}_{2,j}} = (\hat{p}_2 - y_2)x_j = (0.2144 - 0)x_j = 0.2144x_j$
- $\frac{\partial L}{\partial \boldsymbol{\theta}_{3,j}} = (\hat{p}_3 - y_3)x_j = (0.3537 - 0)x_j = 0.3537x_j$

Example 3

- Compute the gradients for each feature
- For class 1:
- $\frac{\partial L}{\partial \boldsymbol{\theta}_{1,1}} = -0.5681 \times 1 = -0.5681$ $\frac{\partial L}{\partial \boldsymbol{\theta}_{1,2}} = -0.5681 \times 2 = -1.1362$
- For class 2:
- $\frac{\partial L}{\partial \boldsymbol{\theta}_{2,1}} = 0.2144 \times 1 = 0.2144$ $\frac{\partial L}{\partial \boldsymbol{\theta}_{2,2}} = 0.2144 \times 2 = 0.4288$
- For class 3:
- $\frac{\partial L}{\partial \boldsymbol{\theta}_{3,1}} = 0.3537 \times 1 = 0.3537$ $\frac{\partial L}{\partial \boldsymbol{\theta}_{3,2}} = 0.3537 \times 2 = 0.7074$

Example 3

- Update Weights
- For class 1:
- $\theta_{1,1} := 0.1 + 0.1 \times 0.5681 = 0.15681$
- $\theta_{1,2} := 0.2 + 0.1 \times 1.1362 = 0.31362$
- For class 2:
- $\theta_{2,1} = 0 - 0.1 \times 0.2144 = -0.02144$
- $\theta_{2,2} = -0.1 - 0.1 \times 0.4288 = -0.14288$
- For class 3:
- $\theta_{3,1} = 0.1 - 0.1 \times 0.3537 = 0.06463$
- $\theta_{3,2} = 0.1 - 0.1 \times 0.7074 = 0.02926$

Example 3

- Compute New Logits with Updated Weights:
- $z_1 = 0.78305$
- $z_2 = -0.3072$
- $z_3 = 0.12315$

Example 3

- Apply Softmax to Compute New Probabilities $\hat{p}_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$
- $e^{z_1} = 2.1886$ $e^{z_2} = 0.7357$ $e^{z_3} = 1.1309$
- $\sum_{j=1}^{3} e^{z_j} = 4.0552$
- $\hat{p}_1 = 0.5397$ $\hat{p}_2 = 0.1814$ $\hat{p}_3 = 0.2789$

Example 3

- Compute New Cross-Entropy Loss $L = -\sum_{k=1}^{K} y_k \log \hat{p}_k$
- $L = -(1 \cdot \log(0.5397) + 0 \cdot \log(0.1814) + 0 \cdot \log(0.2789))$
- $L = -\log(0.5397) \approx 0.6173$
- Initial Cross-Entropy Loss: $L \approx 0.8392$
- New Cross-Entropy Loss After One Update: $L \approx 0.6173$
- As we can see, the cross-entropy loss has decreased from 0.8392 to 0.6173 after just one iteration of gradient descent.

Iris plants

- Use softmax regression to classify the iris plants into all three classes
- Scikit-Learn's LogisticRegression classifier uses **softmax regression** automatically when you train it on **more than two classes** (assuming you use solver="**lbfgs**", which is the default).
- L-BFGS (**L**imited-memory **B**royden–**F**letcher–**G**oldfarb–**S**hanno) is an optimization algorithm
- It also applies $\ell 2$ regularization by default, which you can control using the hyperparameter C

Iris plants

```
X = iris.data[["petal length (cm)", "petal width (cm)"]].values
y = iris["target"]
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

softmax_reg = LogisticRegression(C=30, random_state=42)
softmax_reg.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(C=30, random_state=42)
```

- ask your model to tell you what type of iris (iris with petals that are 5 cm long and 2 cm wide) it is

```
softmax_reg.predict([[5, 2]])
```

Iris plants

- It will answer Iris virginica (class 2) with 96% probability (or Iris versicolor with 4% probability):

```
array([2])
```

```
softmax_reg.predict_proba([[5, 2]]).round(2)
```

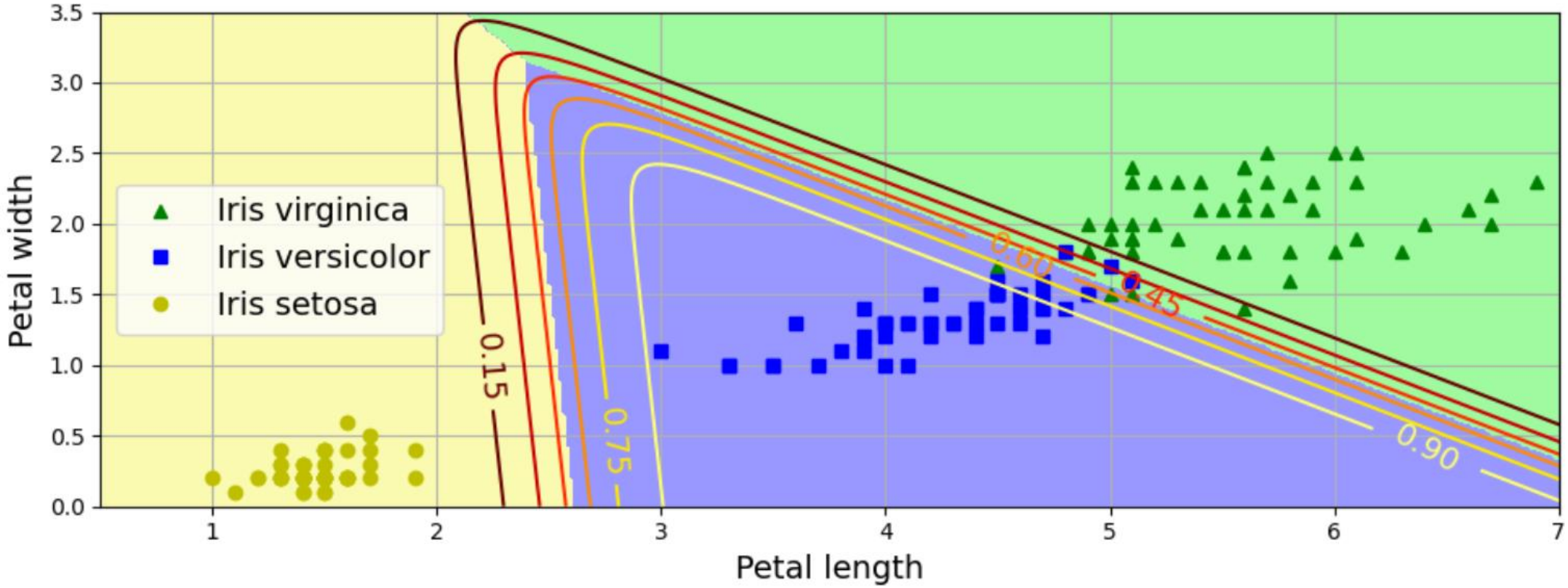
```
array([[0.  , 0.04, 0.96]])
```

Iris plants

Decision boundaries:

- Decision boundaries between any two classes are linear
- The probabilities for the Iris versicolor class, represented by the curved lines (e.g., the line labeled with 0.30 represents the 30% probability boundary). Notice that the model can predict a class that has an estimated probability below 50%. For example, at the point where all decision boundaries meet, all classes have an equal estimated probability of 33%.

Iris plants



- ~~Decision boundaries~~