Logistic Regression 2 – Iris Case and Softmax

Michael Claudius, Associate Professor, Roskilde

31.03.2020

The Iris flower case

- **Data set with 150 Iris pictures of 3 different species (50 each)**

Figure 4-22. Flowers of three iris plant species[14]

Choose and build classifier(s)

- **Find features and make a decision. Remember simple ones first!**

- **First, a binary classification based on**
  1. **Petal width, Iris Virginica or Not Iris Virginica**
  2. **Petal width and length, decision boundaries Virginica or Not Iris Virginica**

- **Second, Multinomial classifier based on**
  1. **Petal width and length, Iris Virginica OR Iris Versicolor OR Iris Setosa**
  2. **Using Softmax**

- **Third, classification based on**
  1. **Sepal width, Iris Virginica or Not Iris Virginica**
  2. **Sepal width and length, decision boundaries Iris Virginica or Not Iris Virginica**

- **Fourth, Multinomial classifier based on**
  1. **Petal width and length , Sepal width and length**

Classifier code using Sklearn on Iris data set

- **Highlights of code**

Import data set

```
from sklearn import datasets
iris = datasets.load_iris()
list(iris.keys())
X = iris["data"][:, 3:]  # petal width
y = (iris["target"] == 2).astype(np.int)  # 1 if Iris virginica, else 0
```

Train regression model

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(solver="lbfgs", random_state=42)
log_reg.fit(X, y)
```

Plot probability and decision boundary
A lot of hokus pokus. Next slide

- **More is given in the LogisticRegIris.ipynb program on teacher's home page**

Iris: Probability plot with a decision boundary

- **Probability function using petal width**

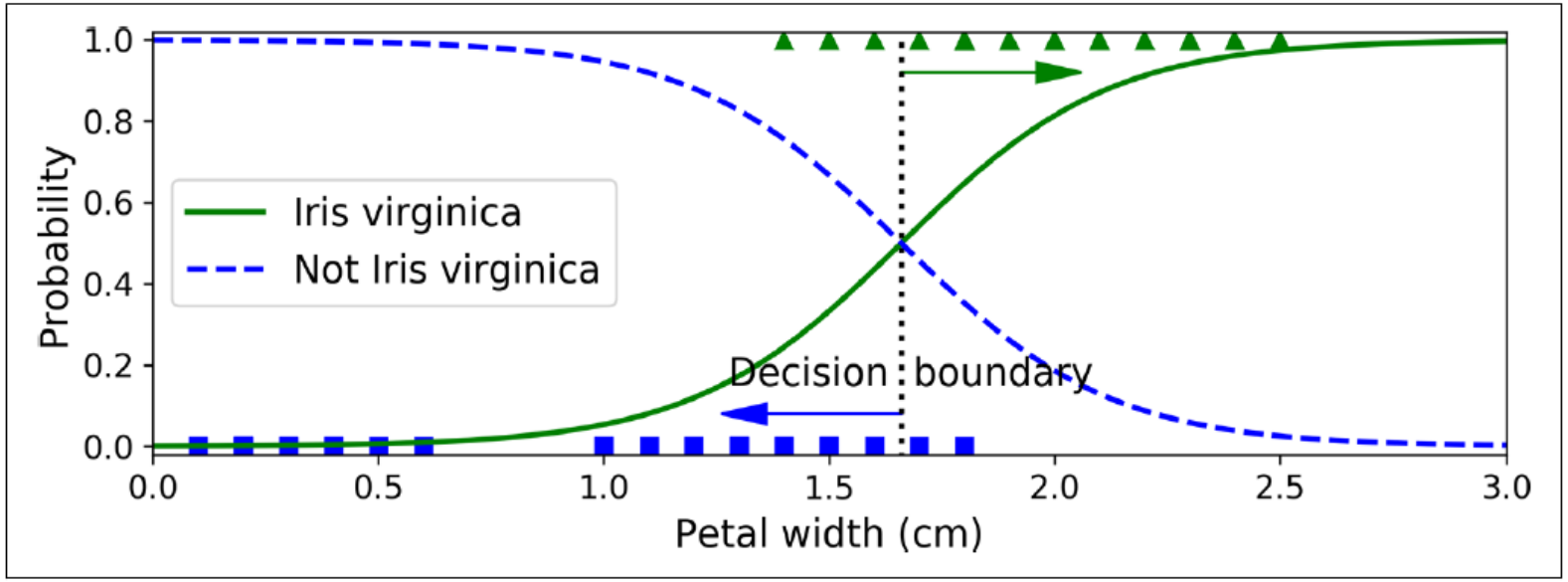


Figure 4-23. Estimated probabilities and decision boundary

Iris: Probability plot with decision boundaries

- **Probability function using petal length and petal width**

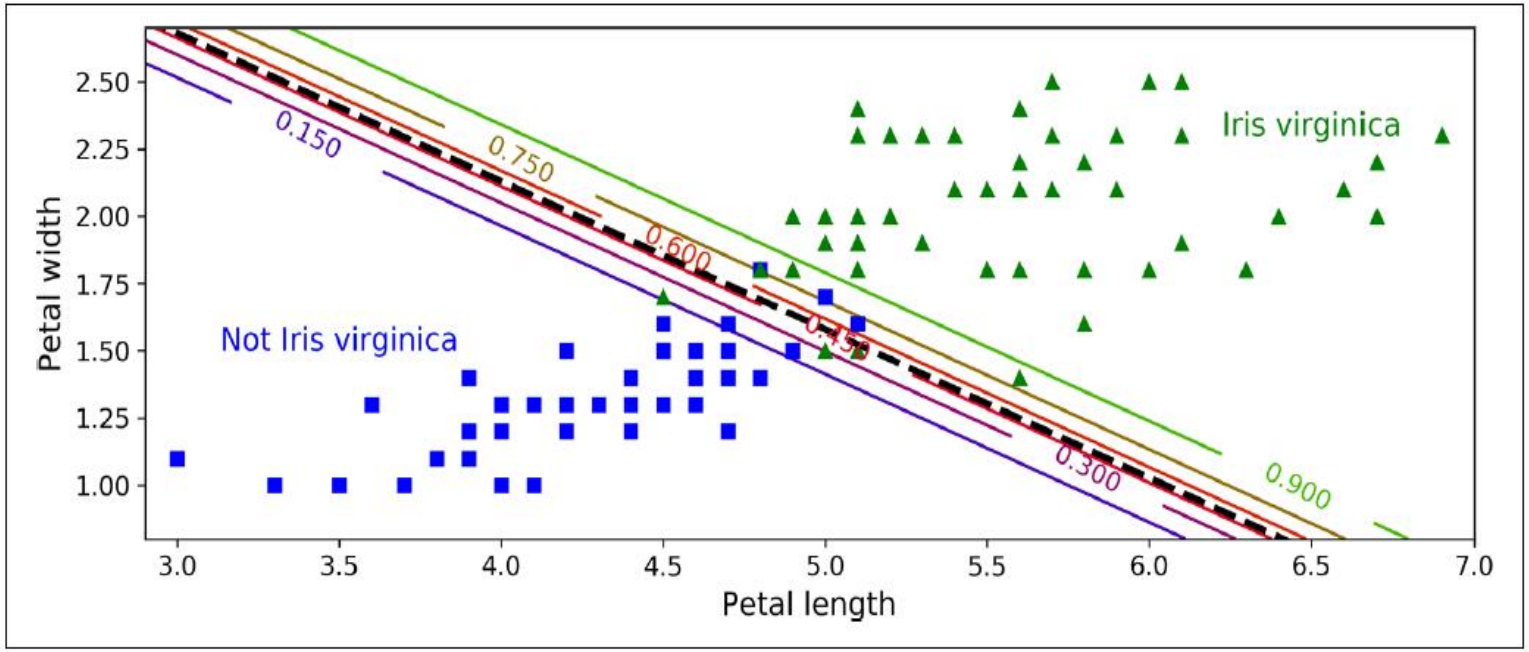


Figure 4-24. Linear decision boundary

- **Notice the linear decision boundaries (e.g. green 90%probability)**
- **What if I want more than two classes? No problem use Softmax regression**

Classifier using Softmax Regression

- **Softmax is a multinomial logistic regression classifier**
- **Support 2 or more classes; multiple classes**
- **Predict one class at a time**
- **Estimate probability for each class given an instance X**
- **Classes must exclude each other!**

Softmax Regression Steps

- **Compute a score (Softmax score) for each class**
- **Estimate the probability for each class using normalized exponential (softmax function)**
- **Predict the the class with highest probability**
- **Train the model so high probability for the target classes**
- **Minimizing the cross cost entropy function**
- **All done by applying Batch Gradient Descent**

- **All in all, similar steps as linear regression as well as logistic regression for binary class**

- **BUT more complicated mathematical formulas and functions**

Softmax functions

- **This I will skip ☺☺ I don't want to be killed by students**

Equation 4-19. Softmax score for class k

$$s_k(\mathbf{x}) = \mathbf{x}^\intercal \boldsymbol{\theta}^{(k)}$$

Equation 4-20. Softmax function

$$\hat{p}_k = \sigma(\mathbf{s}(\mathbf{x}))_k = \frac{\exp\left(s_k(\mathbf{x})\right)}{\sum_{j=1}^{K} \exp\left(s_j(\mathbf{x})\right)}$$

Equation 4-21. Softmax Regression classifier prediction

$$\hat{y} = \underset{k}{\operatorname{argmax}}\ \sigma(\mathbf{s}(\mathbf{x}))_k = \underset{k}{\operatorname{argmax}}\ s_k(\mathbf{x}) = \underset{k}{\operatorname{argmax}}\ \left(\left(\boldsymbol{\theta}^{(k)}\right)^\intercal \mathbf{x}\right)$$

Equation 4-22. Cross entropy cost function

$$J(\boldsymbol{\Theta}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)$$

In this equation:

- $y_k^{(i)}$ is the target probability that the i^{th} instance belongs to class k. In general, it is either equal to 1 or 0, depending on whether the instance belongs to the class or not.

Equation 4-23. Cross entropy gradient vector for class k

$$\nabla_{\boldsymbol{\theta}^{(k)}} J(\boldsymbol{\Theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{p}_k^{(i)} - y_k^{(i)}\right)\mathbf{x}^{(i)}$$

Classifier code using Softmax on Iris data set

- **Highlights of code**

Import data set

```
from sklearn import datasets
iris = datasets.load_iris()
list(iris.keys())
X = iris["data"][:, (2,3)]  # petal length , petal width
y = iris["target"] #all 3 classes involved
```

Train regression model

```
from sklearn.linear_model import LogisticRegression
soft_reg = LogisticRegression(class="multinomial, "solver="lbfgs", C=10)
soft_reg.fit(X, y)
```

```
soft_reg.predict([5,2])
array([[1.55606703e-04, 9.51675722e-01, 4.81686717e-02]])
```

- **Plot probability and decision boundaries**
  A lot of hokus pokus. Next slide

- **More is given in the LogisticRegIris.ipynb program on teacher's home page**

Iris: Probability plot with decision boundaries

- **Probability function using petal length and petal width and 3 classes**

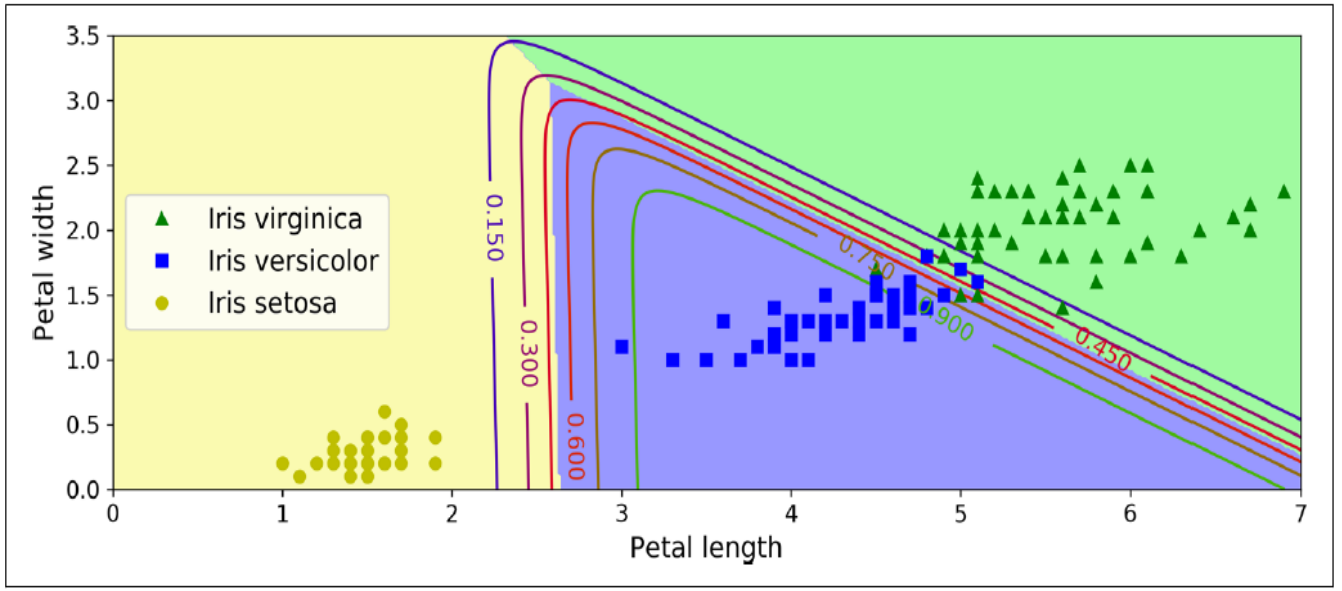


Figure 4-25. Softmax Regression decision boundaries

- **Notice the linear decision boundaries (e.g. green 90%probability) for Iris Versicolor**

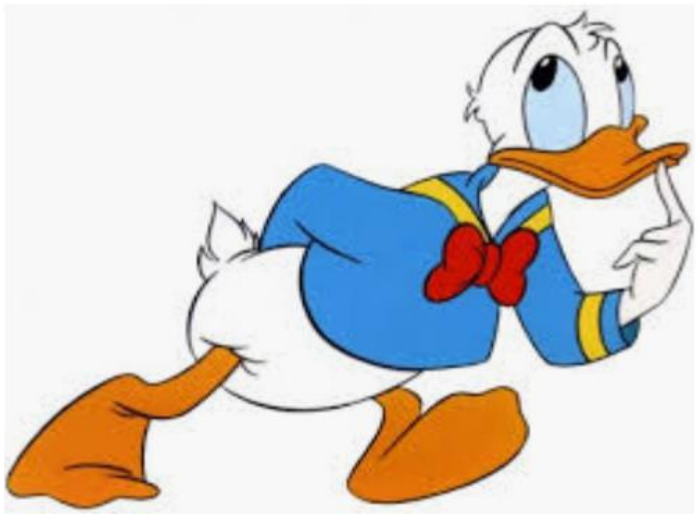
The Program Code

- **It is time for a short review of the code**
- **I will highlight a few points**

  - Logistic Regression Iris Program

Exercise

- **It is time for discussion, coding and solving the Iris case yourself**
  - Logistic Regression Iris Exercise
    Logistic Regression Iris Program



- **Or if you need rest an old timer:** Donald Duck on Math hunting 1959