Logistic Regression

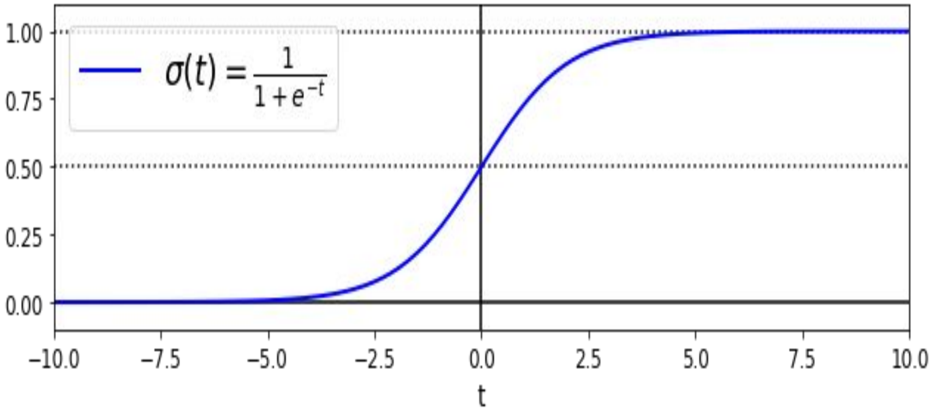
Estimating Probabilities

Training and Cost Functions

Decision Boundaries

Softmax Regression

Estimating Probabilities



Logistic regression is a linear binary classifier that outputs a logistic in the form of a sigmoid function rather than a direct result.

$\sigma(t) < 0.5$ when $t < 0$;
$\sigma(t) > 0.5$ when $t >= 0$

$t = \text{logit}(p) = \log(p / (1 - p))$
It’s the inverse of the logistic function. Also known as the log odds ratio.

Logistic Regression Model Prediction
$y = 0$ if $p < 0.5$ or $y = 1$ if $p >= 0.5$
where $p = h_\theta(x) = \sigma(x^T \theta)$

Training and Cost Function

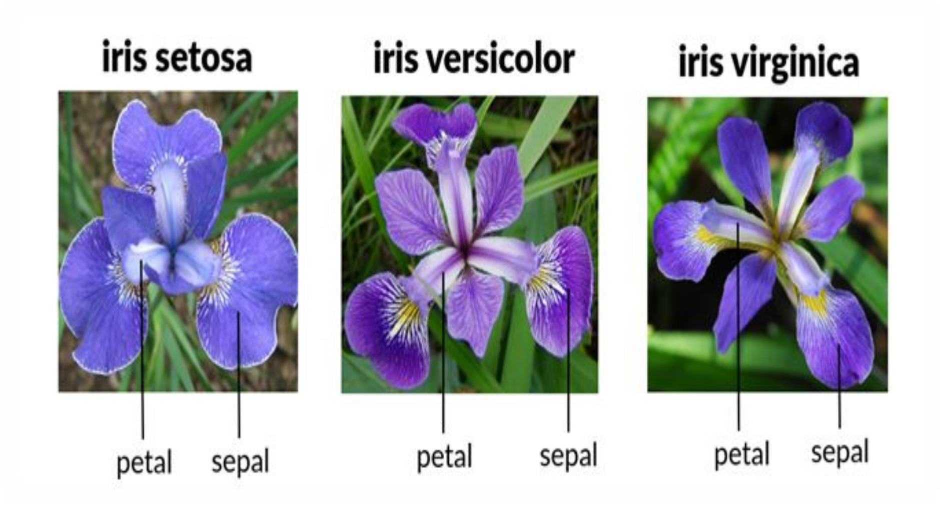
Training sets the parameter vector so that the model estimates high probabilities for positive cases ($y = 1$) and low probabilities for negative cases ($y = 0$).

Cost function for a single training instance: $c(\theta) = -\log(p)$ if $y = 1$; $-\log(1 - p)$ if $y = 0$

Since $-\log(t)$ increases as t approaches 0, the cost will be larger if the model estimates a probability close to 0 for a positive instance or close to 1 for a negative case. High costs for incorrect predictions and low costs for correct predictions.

The cost function (log loss) is convex, so optimization is guaranteed to find a global minimum with sufficient time and a properly sized learning rate.

Decision Boundaries



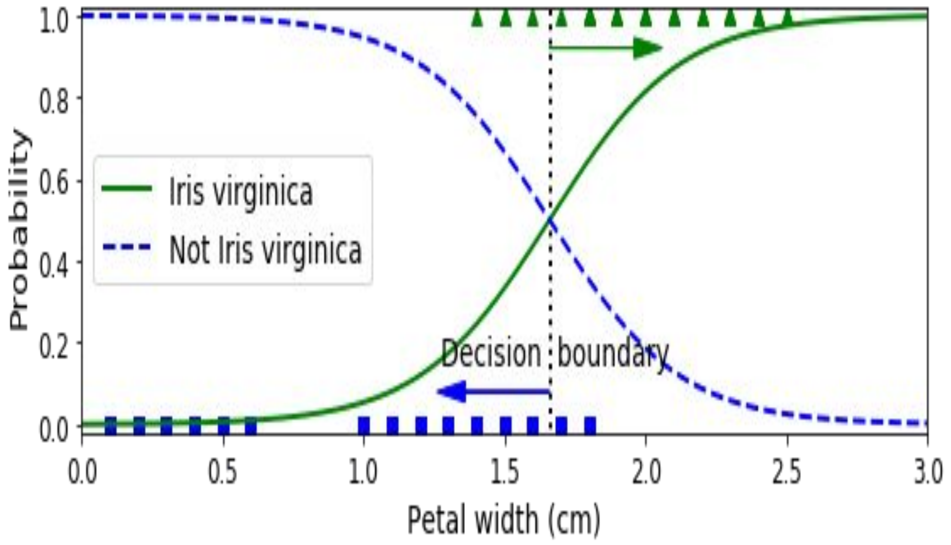
```
from sklearn import datasets
iris = datasets.load_iris()

list(iris.keys())
['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename']

X = iris["data"][:, 3:]  # petal width
y = (iris["target"] == 2).astype(np.int)  # 1 if Iris virginica, else 0

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

Decision Boundaries cont.



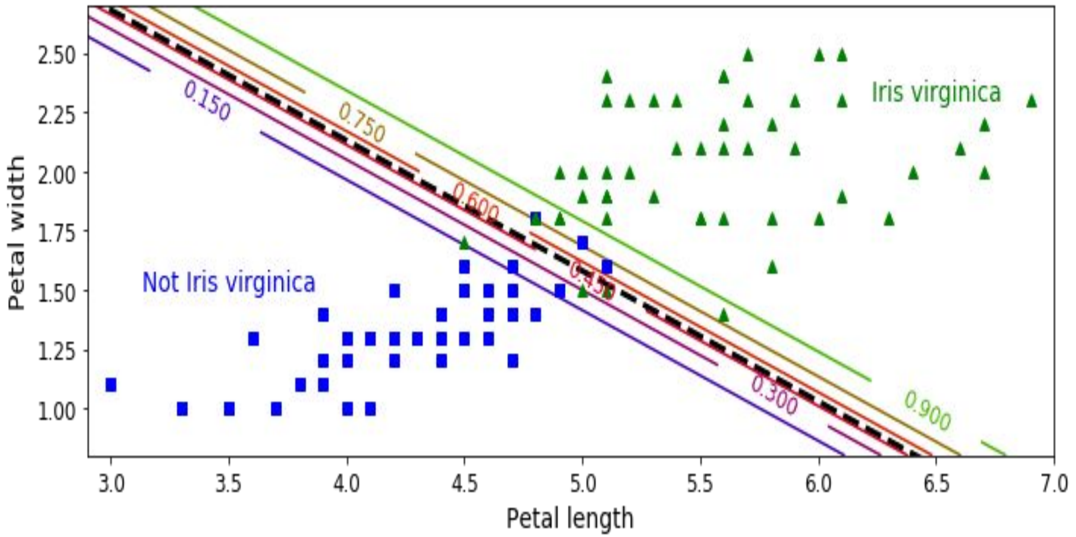
```
X_new = np.linspace(0, 3, 1000).reshape(-1, 1)
y_proba = log_reg.predict_proba(X_new)
decision_boundary = X_new[y_proba[:, 1] >=
0.5][0]
plt.plot(X_new, y_proba[:, 1], "g-", linewidth=2, label="Iris
virginica")
plt.plot(X_new, y_proba[:, 0], "b--", linewidth=2, label="Not Iris
virginica")
```

Above 2.0, classifier is highly confident that the flower is Virginica; below 1.0, it is highly confident that it is not.

Decision boundary exists at 1.6 where both probabilities are equal to 50%.

```
>>> log_reg.predict([[1.7], [1.5]])
array([1, 0])
```

Decision Boundaries cont.



Classifying both petal width and length

Note that the boundary is linear

Dashed line at 50% represents the decision boundary

All flowers above the green line have over a 90% chance of being classified Virginica., while those below the blue line have less than a 15%.

Softmax Regression

Logistic regression model generalized to handle multiple classes directly without needing to train multiple binary classifiers. Also referred to as Multinominal Logistic Regression. (Recall the one-vs-one and one-vs-all approaches from chapter 3).

Should only be used for mutually exclusive classes because it only identifies one class at a time - it's multinominal, not multioutput.

Given an instance x, softmax regression computes a score s(x) for each class k and then applies the softmax function to the scores to calculate the probability p of the instance belonging to each class. Each class has its own dedicated parameter vector.

The function then computes the exponential of every score and normalizes them by dividing by the sum of all of the exponents.

It then predicts the class with the highest estimated probability (highest score) using the argmax operator, which returns the value of a variable that maximizes a function.

Cross Entropy

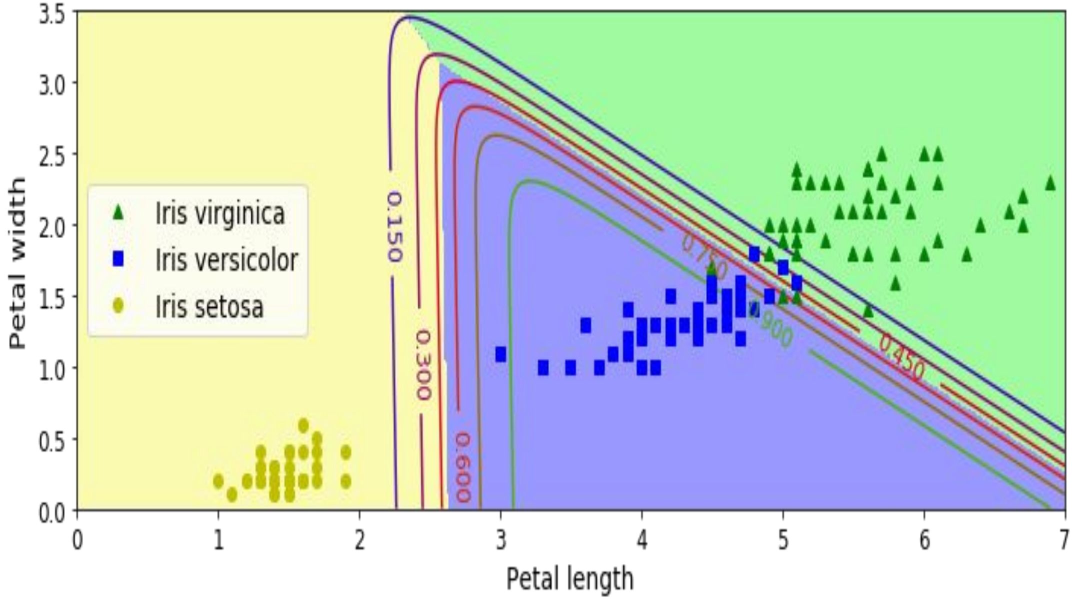
Ideally, a model should estimate high probability for a target class and low probabilities for the other classes.

Minimizing the cross entropy cost function, frequently used to measure how well estimated class probabilities match the target classes, penalizes a model when it gives a low estimate for a target class.

Cross entropy is commonly used to measure how well a set of estimated class probabilities match the target classes.

Using the cross entropy gradient vector of the cost function, the gradient vector for every class can be computed. Gradient descent or another optimization algorithm can then be applied to find the parameter matrix that minimizes the cost function.

Softmax Regression cont.



```
X = iris["data"][:, (2, 3)]  # petal
length, petal width
y = iris["target"]

softmax_reg =
LogisticRegression(multi_class="multinom
ial",solver="lbfgs", C=10,
random_state=42)
softmax_reg.fit(X, y)

>>>softmax_reg.predict([[5, 2]])
array([2])
>>>softmax_reg.predict_proba([[5, 2]])
array([[6.38014896e-07, 5.74929995e-02,
9.42506362e-01]])
```

Decision boundaries between any two classes are linear.

The point where all decision boundaries meet, all classes have an equal estimated probability of 33%.