DATASCI 207

Nedelina Teneva, PhD
nteneva@berkeley.edu

School of Information, UC Berkeley

Announcements

- HW1 is graded
    - Issues with the grades on Gradescope (working on fixing it)
- Group selection: 2 weeks from now
- **Suggestions on what material you'd like to see in more depth are welcome!**
- Break out group work for part of today

ML Models - Tools and Frameworks

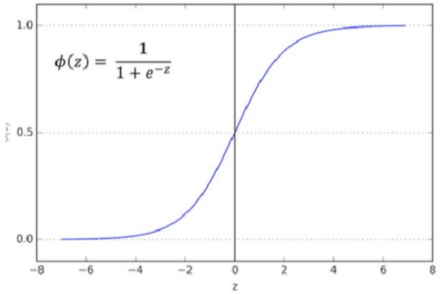
- We'll use at least 3 ways …
  - From scratch
    - Focus on scientific programming using using python, numpy and lin. algebra primitives
  - Sklearn
    - Apply off the shelf algorithms (regression, clustering algorithms, etc.)
  - Tensorflow (or Pytorch)
    - Training neural based models, relies on deep learning specific primitives
- Other frameworks and tools are possible depending on the data modality or data scale
  - E.g. when working with graph data one may use pytorch geometric
  - E.g. when working at scale, we may use AWS Sagemaker for model training and hosting

Classification vs Regression

Linear Regression $\hat{y} = xw^T + b$

Logistic Regression $\hat{y} = \dfrac{1}{1 + e^{-(xw^T + b)}}$

Apply a sigmoid to the linear regression model



Logistic Regression Recap

We're not going to use MSE for logistic regression. Instead, we'll use the *logistic loss*, also called *binary cross-entropy* (KL divergence between empirical and predicted distribution).

$$LogLoss = \frac{1}{m}\sum_i -y_i \log(\hat{y}_i) - (1 - y_i)\log(1 - \hat{y}_i)$$

Despite this new loss function, it turns out that the gradient computation is the same as it was for MSE with linear regression (a happy coincidence ...)

$$\nabla J(W) = \frac{1}{m}(h_W(X) - Y)X$$

Let's write the code for a single gradient descent step:

```
# Run gradient descent
m, n = X.shape  # m = number of examples; n = number of features (including bias)
learning_rate = 0.1

preds = sigmoid(np.dot(X, W))

loss = (-Y * np.log(preds) - (1 - Y) * np.log(1 - preds)).mean()

gradient = np.dot((preds - Y), X) / m
W = W - learning_rate * gradient

print('predictions:', preds)
print('loss:', loss)
print('gradient:', gradient)
print('weights:', W)
```

Regularization

Generally seeks to reduce any unlikely coefficients (e.g high values)

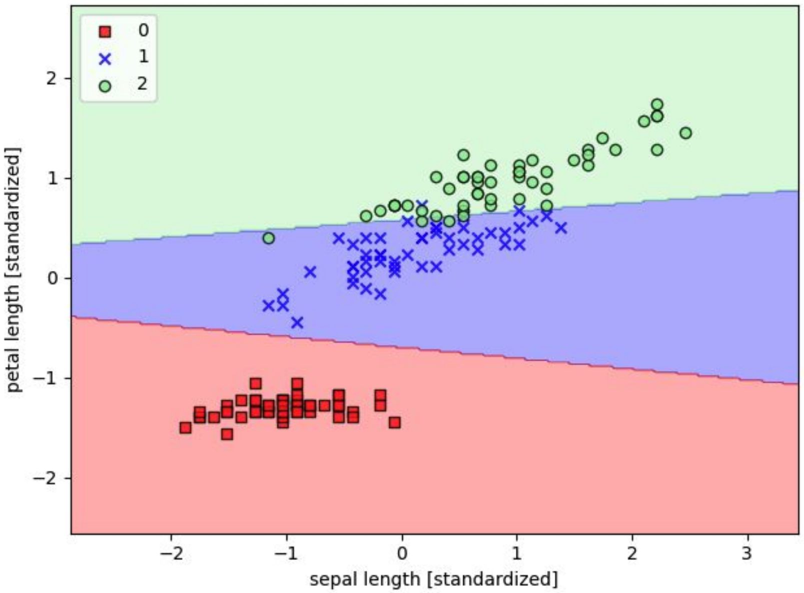
Demo with code: https://github.com/MIDS-W207/nteneva/blob/main/live_sessions_current/week4/Gradient_Descent_for_Logistic_Regression.ipynb

The Decision Boundary

- What is a decision boundary?

Decision Boundary

- How do we compute it?




```
def plot_decision_regions (X, y, classifier, plot_test,
resolution=0.02):
    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[: len(np.unique(y))])

    # plot the decision surface (a discretized mesh)
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha= 0.3, cmap=cmap)
    plt.xlim(xx1. min(), xx1. max())
    plt.ylim(xx2. min(), xx2. max())

    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0],
                    y=X[y == cl, 1],
                    alpha= 0.8,
                    c=colors[idx],
                    marker=markers[idx],
                    label=cl,
                    edgecolor= 'black')
```

Code in https://github.com/MIDS-W207/nteneva/blob/main/live_sessions_current/week4/Decision_Boundaries.ipynb

Metrics

	Class 1 (+) Predicted	Class 2(-) Predicted
Class 1 (+) Actual	TP	FN
Class 2 (-) Actual	FP	TN

class 1 (+); class 2 (-)

Accuracy (Classification Rate):
(TP + TN) / (TP + TN + FP + FN)
(# correctly classified examples/ # examples)

Recall: TP/ (TP + FN)
(# correctly classified pos. examples/# pos. examples)

Precision: TP/ (TP + FP)
correctly classified + examples/ # of predicted + examples

See: https://ibug.doc.ic.ac.uk/media/uploads/documents/ml-lecture3-2014.pdf

Break out session exercise

https://github.com/MIDS-W207/nteneva/blob/main/live_sessions_current/week4/LogisticRegresssionExercise.ipynb