Machine Learning Worksheet 06

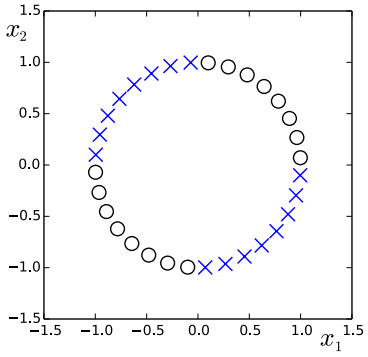
Linear Classification

1 Linear separability

Problem 1: Show that for a linearly separable data set, the maximum likelihood solution for the logistic regression model is obtained by finding a vector $\boldsymbol{w}$ whose decision boundary $\boldsymbol{w}^T\phi(\boldsymbol{x}) = 0$ separates the classes and then taking the magnitude of $\boldsymbol{w}$ to infinity. Assume that $\mathbf{w}$ contains the bias term.

Finding a separating hyperplane puts all training points on the correct side (and thus ensures posterior class probabilities > 0.5 for each training point). Taking the magnitude of $\boldsymbol{w}$ to infinity then assigns each training point a posterior class probability of 1 (this happens because of the shape of the logistic sigmoid function—it becomes a heaviside function in this particular case).

Problem 2: Which basis function $\phi(x_1, x_2)$ makes the data in the example below linearly separable (crosses in one class, circles in the other)?



$$\phi(x_1, x_2) = x_1 x_2$$

2 Basis functions

Problem 3: The decision boundary for a linear classifier on two-dimensional data crosses axis x_1 at 2 and x_2 at 5. Write down the general form of this linear classifier model with a bias term (how many parameters do you need, given the dimensions?) and calculate possible coefficients (parameters).

$$
\begin{aligned}
w_0 + w_1 x_1 + w_2 x_2 &= 0 \\
w_0 + 2w_1 &= 0 \\
w_1 &= -\frac{w_0}{2} \\
w_0 + 5w_2 &= 0 \\
w_2 &= -\frac{w_0}{5}
\end{aligned}
$$

$$
\begin{aligned}
\text{set, e.g., } w_0 &= -2 \\
w_1 &= 1 \\
w_2 &= \frac{2}{5}
\end{aligned}
$$

or anything proportional

3 Getting your hands dirty!

Problem 4: Solve the first of the four programming exercises (i.e., fill in the blanks) in the file `LinearClassification.ipynb` in piazza's "General Resources". Put the code here.

```
def pred(X, W, b):
    def sigmoid(x):
        return 1.0 / (1.0 + np.exp(-x))
    return sigmoid(np.dot(X, W.T) + b)
```

Problem 5: Solve the second of the four programming exercises (i.e., fill in the blanks) in the file `LinearClassification.ipynb` in piazza's "General Resources". Put the code here.

```
def loglikelihood(X, Z, W, b):
    return (Z[:, np.newaxis] * pred(X, W, b) + (1.0 - Z[:, np.newaxis])
     * (1.0 - pred(X, W, b))).sum(0)
```

Problem 6: Solve the third of the four programming exercises (i.e., fill in the blanks) in the file `LinearClassification.ipynb` in piazza's "General Resources". Put the code here.

```
def grad(X, Z, W, b):
```

```
dLdW = ((pred(X, W, b) - Z[:, np.newaxis]) * X).sum(0)
dLdb = (pred(X, W, b) - Z[:, np.newaxis]).sum(0)
return dLdW[np.newaxis, :], dLdb
```