Plotting and understanding basis functions

This note encourages you to discover some things about basis functions for yourself, through some experimentation. If we can't put maths into code (at least the type of maths in this course), we don't understand it. Implementing something forces us to confront every detail. Looking at special cases in code can also help build intuition.

We start with a quick review of how to plot functions in Python, then plot some different basis functions, and plot linear regression fits using them.

1 Plotting one-dimensional functions

We can plot one-dimensional functions by evaluating them on a fine grid. Make sure you have a working python setup, and can run this code to plot the cosine function:

```
import numpy as np
import matplotlib.pyplot as plt

grid_size = 0.1
x_grid = np.arange(-10, 10, grid_size)
f_vals = np.cos(x_grid)
plt.clf()
plt.plot(x_grid, f_vals, 'b-')
plt.plot(x_grid, f_vals, 'r.')
plt.show()  # Not necessary with ipython3 --matplotlib
```

Please do run the code snippets from these notes since interacting with the notes will help you to better understand and remember the described concepts.

Our code is not really evaluating the whole function. We just evaluated it at the points shown by red dots. However, the blue line segments joining these dots closely approximate our function. If we decrease the `grid_size` further, the blue lines will become as accurate as it's possible to plot on a computer screen.

We could use the cosine function as a basis function in our model. We can also create and plot other basis functions. For example RBFs:

```
def rbf_1d(xx, cc, hh):
    return np.exp(-(xx-cc)**2 / hh**2)

plt.clf()
grid_size = 0.01
x_grid = np.arange(-10, 10, grid_size)
plt.plot(x_grid, rbf_1d(x_grid, cc=5, hh=1), '-b')
plt.plot(x_grid, rbf_1d(x_grid, cc=-2, hh=2), '-r')
plt.show()
```

The code above plots a radial basis function centred at $x=5$ in blue, and one in red that's twice as wide and centred at $x=-2$.

[The website version of this note has a question here.]

1.1 Combinations of basis functions

You might think of the function $f(\mathbf{x}) = \sum_k w_k \phi_k(\mathbf{x})$ as being made up of separate parts ϕ_k. For example, plot or sketch the function

$$f(x) = 2\phi_1(x) - \phi_2(x), \tag{1}$$

where ϕ_1 and ϕ_2 are unit bandwidth RBFs centred at -5 and $+5$. You can identify two separate bumps, each a scaled version of a basis function.

An RBF lets us push up and down a function in a region close to its centre. In low-dimensions (1d curve fitting, or fitting a surface over 2D inputs) we can evenly space a grid of RBFs and represent a large class of smooth functions. These functions can be made high or low in any region, by setting the weight associated with the RBF there. The widths of the RBFs should be set so that they overlap, or our function will consist of isolated "spikes".

In higher dimensions we can't use a grid of RBFs (why? Answer in footnote[1]). One way to set the centres of RBFs in higher dimensions is to place them on K training points picked at random.

Although the above "push up and down"-intuition has some use, it's frequently not obvious by eye what the underlying basis functions are. For example, we can create a function with monomial basis functions

$$\phi_1(x) = 1 \tag{2}$$
$$\phi_2(x) = x \tag{3}$$
$$\phi_3(x) = x^2, \tag{4}$$

and weights $\mathbf{w} = [5 \;\; 10 \;\; -1]^\top$. The resulting function has a simple form,

$$f(x) \;=\; \mathbf{w}^\top \boldsymbol{\phi}(x) \;=\; 5 + 10x - x^2 \;=\; 30 - (x-5)^2, \tag{5}$$

which, unlike any of the basis functions, peaks at $x = 5$. Similarly, a function made by combining several overlapping RBFs can have peaks in between the peaks of any of the underlying basis functions.

The following question may be difficult on first reading, but we want to give you an early sense for the sort of reasoning you will need to do in this course. Later in the course, you will face questions requiring similar kinds of reasoning, but without any hint:

[The website version of this note has a question here.]

1.2 Linear regression fits

Consider a dataset with 3 datapoints:

$$\mathbf{y} = \begin{bmatrix} 1.1 \\ 2.3 \\ 2.9 \end{bmatrix}, \qquad X = \begin{bmatrix} 0.8 \\ 1.9 \\ 3.1 \end{bmatrix}. \tag{6}$$

Try fitting different linear regression models to this dataset. As it's early in the course we provide a demonstration, which you should at least run, and check that any small changes that you can think of do what you expect:

```
import numpy as np
import matplotlib.pyplot as plt

# Set up and plot the dataset
yy = np.array([1.1, 2.3, 2.9]) # N,
X = np.array([[0.8], [1.9], [3.1]]) # N,1
plt.clf()
plt.plot(X, yy, 'x', markersize=20, mew=2)

# phi-functions to create various matrices of new features
# from an original matrix of 1D inputs.
def phi_linear(Xin):
    return np.hstack([np.ones((Xin.shape[0],1)), Xin])
```

1. Because in D-dimensions, even a grid of width 2 requires 2^D basis functions, which rapidly becomes impossible as D increases.

```
def phi_quadratic(Xin):
    return np.hstack([np.ones((Xin.shape[0],1)), Xin, Xin**2])
def fw_rbf(xx, cc):
    """fixed-width RBF in 1d"""
    return np.exp(-(xx-cc)**2 / 2.0)
def phi_rbf(Xin):
    return np.hstack([fw_rbf(Xin, 1), fw_rbf(Xin, 2), fw_rbf(Xin, 3)])

def fit_and_plot(phi_fn, X, yy):
    # phi_fn takes N, inputs and returns N,D basis function values
    w_fit = np.linalg.lstsq(phi_fn(X), yy, rcond=None)[0] # D,
    X_grid = np.arange(0, 4, 0.01)[:,None] # N,1
    f_grid = np.dot(phi_fn(X_grid), w_fit)
    plt.plot(X_grid, f_grid, linewidth=2)

fit_and_plot(phi_linear, X, yy)
fit_and_plot(phi_quadratic, X, yy)
fit_and_plot(phi_rbf, X, yy)
plt.legend(('data', 'linear fit', 'quadratic fit', 'rbf fit'))
plt.xlabel('x')
plt.ylabel('f')

plt.show()
```

It is possible to fit any three points exactly using a model with three basis functions (which means $\mathbf{f} = \mathbf{y}$). As long as the basis functions create an $N \times K$ or 3×3 feature matrix Φ that is invertible[2], we can set the weights to $\mathbf{w} = \Phi^{-1}\mathbf{y}$. We can write down lots of different models with three basis functions that will agree on the outputs for the three training inputs. However, the predictions will usually be different for new *test* inputs. Predicting the future given past data is an inherently *ill-posed* problem.

2. It turns out that for most of the basis functions we use, including polynomials or RBFs with different centres, when applied to different datapoints, the feature matrix Φ will be invertible when there are as many basis functions as datapoints. Bishop's (1995) *Neural Networks for Pattern Recognition* book defers to Micchelli (1984), *Interpolation of scattered data: Distance matrices and conditionally positive definite functions*, for the technical details.