Lesson 2: Regression & Classification

What is regression?

Mapping continuous input to discrete/continuous outputs

- The term regression became overloaded over time. Initially it described how children of very tall people and children of very short people *tended both to be a bit closer average* height than their parents. i.e. The "function" `child_height(parent_height)` looks like a line and has slope less than 1.
- Now it is used to describe the functional form to approximate a dataset.

Linear regression

Use a line to fit the dataset to minimize the error.

Tool: calculus

Error/cost function: squared error is good due to continuous and it tends to bring the thing back to the mean.

Polynomial Regression

$$f(x) = c_0 + c_1 x + c_2 x^2 + ... + c_k x^k$$

- k = 0: constant * (Best fit is the mean) *
- k = 1: line
- k = 2: parabola
- *note: k should be less than the number of data points, otherwise unconstrained*
- Larger k, more degree of freedom, less error, better fitting, but more overfitting.

When we have an overfit candidate function, though it produces no error on the training set, it will not generalize.

It's possible to fit affine hyperplanes to data. If we transform the input vectors with squared or cubed terms, etc. we can also fit any higher dimensional polynomial with the same technique.

Performing linear regression

Suppose we have a training set $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$ where x_i are in R^n and y_i are in R

$$\begin{pmatrix} 1 & x_1 & x_1^2 & ... & x_1^k \\ 1 & x_2 & x_2^2 & ... & x_2^k \\ 1 & x_3 & x_3^2 & ... & x_3^k \\ & & ... & & \\ 1 & x_n & x_1^2 & ... & x_n^k \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ ... \\ c_k \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ ... \\ y_n \end{pmatrix}$$

Then we try to solve for W in:

$$\mathbf{XW} \approx \mathbf{Y}$$
$$\mathbf{W} \approx (\mathbf{X^T X})^{-1} \mathbf{X^T Y}$$

For details: http://en.wikipedia.org/wiki/Linear_regression

This formula minimizes the squared error (proven by differentiating $L(w) = ||Y - wX||^2$).

Errors

All data has noise/errors: * sensor error, * malicious agents, * transcription error, * unmodeled influences, etc.

We should not overfit to training data in order not to model the errors.

We want to train until we minimuze the mean squared error (MSE) on the training data.

$$MSE = \frac{1}{n} \sum_i (w \cdot x_i - y_i)^2$$

Next we want to compute the error on various testing sets to make sure that the model is not so complicated that it overfits and doesn't generalize.

Cross validation

To check that data generalizes well, hold aside a subset of the training data as **testing data**, training on the remaining set and test against the testing data. **Choose our inductive bias (picking model or model complexity)** (Usage of the crossval: select better model / tune hyperparameters) such that when the model is trained on the training set it's error on the testing set is still minimal.

An assumption: i.i.d, independent and identical distirbution.

Take each fold as the test set and average the error rate, which then becomes a good representation of the performance.

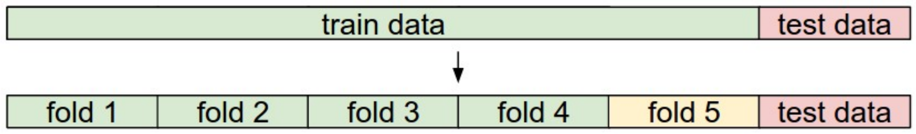

Figure 1: cross validation demo

Other input spaces

- salar input, continuous
- vector input, continuous
- discrete input, vector of scalar

In the real world the values of each feature normally don't have a natural order. Convert discrete values to Boolean vectors may be a good choice often.