Difficulties of least squares

1. Outliers

Least squares, as an error metric for fitting, is sensitive to data points that are out in left field — ones that are "mistakes" rather than "errors" in the sense of uncertain values.

Outliers are points that are generated by a different process than the rest of the data.

- e.g.: in the thermal expansion experiment from last time, maybe we transposed some digits while recording one of the temperatures.

- e.g.: a computer vision application: Take the point-matching program from hw2, and imagine converting it to least squares, so that there are lots of point pairs instead of just three.

  This kind of computation is commonly used in vision to estimate motion between images taken at different times. (e.g. a robot-mounted camera trying to understand how much the robot's head has turned.) You can automatically find features in each of the images, and automatically guess correspondences, but this process is error-prone, and occasionally it will match up wrong points.

Both these applications will work poorly with plain old least squares fitting because outliers hold too much sway.

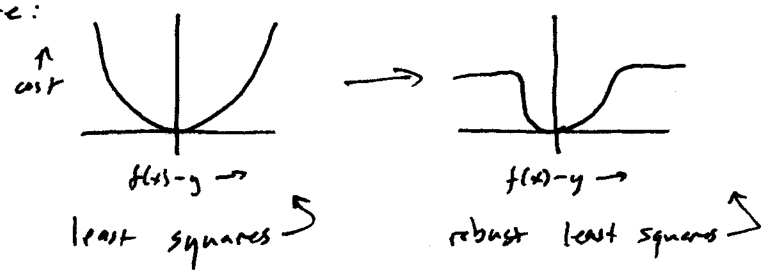
After all, think of the effect of a small change in one point's value on the error:

$$\frac{\partial}{\partial y_k}\left(\sum_i (f_i(x)-y_i)^2\right) = -2(f_k(x)-y_k) \quad \leftarrow \text{point 10 times farther away has 10 times the influence.}$$

So the least squares computation tries really hard to reduce these extreme values, at the expense of penalizing the closer-fitting points.

* demo: influence of outliers on linear regression.

[Solutions to this problem are a whole field called "robust statistics." A simple example: fit, throw out worst few points, fit again, repeat. Another example: use error metric that maxes out at some distance:



]

2. Under- and over-parameterization.

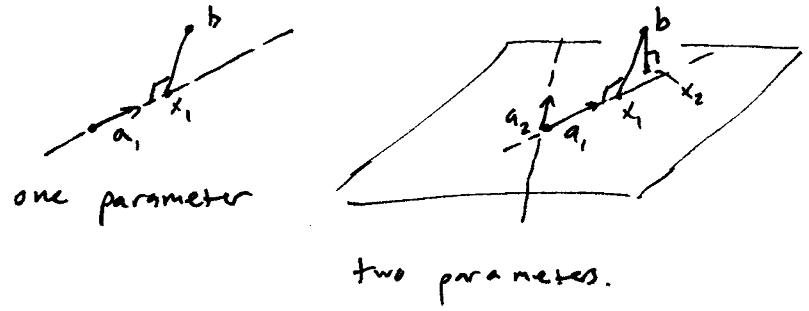
Sometimes when we fit a model to some data, it just doesn't have the expressive power to describe whats going on.

* demo: line fit to points near a parabola.

This is no surprise: its important for the model to be flexible enough. So should we always choose a really flexible model, to be sure it will fit the data?

After all, adding another parameter (another basis function, in linear least squares fitting) will always reduce the residual (or at least can't increase it).

for instance, going from 1 to 2 parameters in the 3xn geometric example:



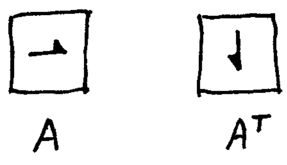
x_1 is in the plane, so its a candidate solution for the 2D problem. Therefore if $x_2 \neq x_1$ it has to be closer to b.

The problem is that too much flexibility can lead to overfitting: the model distorts ~~the~~ to accommodate the small variations in the data, producing a result that fits really well but has nothing to do with the underlying process.

* demo: points near a line, but clustered so that a 5th order polynomial goes crazy.

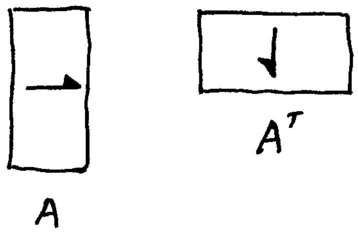
Later on we'll look at methods that can tell us exactly how sensitive the result is to small variations in the data.

Review of matrix algebra facts



$\langle A\rangle_{ij} = \langle A^T\rangle_{ji}$

Matlab : A'



$(A\,B)^T = B^T A^T$ symmetric matrix: $A = A^T$

rows of transpose $\leftrightarrow$ columns of original

---

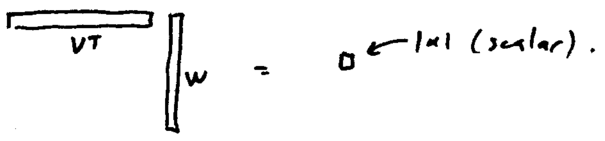
diagonal : entries a_{ii}

(k^{th} diagonal : entries $a_{i,i+k}$

---

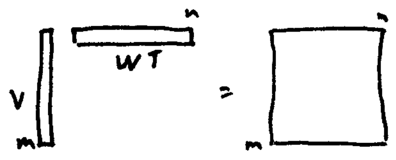
row and column vectors : When convenient, we think of a vector as an $n\times1$ or $1\times n$ matrix. In this class the default will be $n\times1$ (a column vector). If we want a row vector we'll usually write v^T.

Dot product as a row-column product: $v\cdot w = v^T w$



some authors just use this notation instead of the dot.

Outer product: $v\,w^T$



$= \begin{bmatrix} w_1 v & w_2 v & \cdots & w_n v \end{bmatrix} = \begin{bmatrix} v_1 w \\ v_2 w \\ \vdots \\ v_m w \end{bmatrix}$

$(vw^T)_{ij} = v_i w_j$

This is a rank-1 matrix — it is always singular because all the rows (or columns) are multiples of the same vector.