Polynomial fit derivation notes
Used in *polyfit.py*

Daniel Clark

January 27, 2017

Polynomial curve-fitting

Given a real-valued input variable x, we wish to predict a real-valued target variable t.

We can fit a polynomial to an existing set of N data points, with row vector inputs $\boldsymbol{x} \equiv (x_1, ..., x_n)$ and target variables $\boldsymbol{t} \equiv (t_1, ..., t_n)$. For a given input x_n

$$t_n \approx y(x_n, \mathbf{w}) = \sum_{j=0}^{M} w_j x_n^j, \tag{1}$$

where $\mathbf{w} \equiv (w_0, ..., w_M)^\top$ and $y(x_n, \mathbf{w})$ predicts the target t_n using an $M + 1$-dimensional vector of weights, corresponding to each term of the M-order polynomial (plus the line offset w_0). *Note that $y(x_n, \mathbf{w})$ is a nonlinear function of x_n, but a linear function of the coefficients $\{w_j\}$; these type of models are known as *linear models*.

These weights can be determined by minimizing an *error function*, which measures the misfit between the approximation $y(x_n, \mathbf{w})$ and the training set for any given value of $\mathbf{w}$. A widely used error function is the sum-of-squares, measuring the sum of the distance-squared between the predicted point $y(x_n, \mathbf{w})$ and the actual target t_n, from $n = 1, ..., N$.

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \big[y(x_n, \mathbf{w}) - t_n \big]^2 \tag{2}$$

The error function can be solved in closed form to find the optimal solution, $\mathbf{w}^*$, where $E(\mathbf{w}^*)$ is minimized.

Derivation

Setting $y_n \equiv y(x_n, \mathbf{w})$, we have

$$
\begin{aligned}
E(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N}\big[y(x_n,\mathbf{w}) - t_n\big]^2 \\
&= \frac{1}{2}\sum_{n=1}^{N}\big[y_n - t_n\big]^2 \\
&= \frac{1}{2}\sum_{n=1}^{N}\big[y_n^2 - 2y_n t_n + t_n^2\big] \\
&= \frac{1}{2}\sum_{n=1}^{N} y_n^2 - \sum_{n=1}^{N} y_n t_n + \frac{1}{2}\sum_{n=1}^{N} t_n^2 \\
&= \frac{1}{2}\sum_{n=1}^{N} y(x_n,\mathbf{w})y(x_n,\mathbf{w}) - \sum_{n=1}^{N} y(x_n,\mathbf{w})t_n + \frac{1}{2}\sum_{n=1}^{N} t_n^2
\end{aligned}
$$

Since $E(\mathbf{w})$ is a quadratic function of $\mathbf{w}$, its global minimum is found by setting its derivative with respect to $\mathbf{w}$ to 0. First, let‘s take the partial derivative of $y(x_n, \mathbf{w})$ with respect to any component of $\mathbf{w}$, w_i

$$
\begin{aligned}
\frac{\partial}{\partial w_i} y(x_n,\mathbf{w}) &= \frac{\partial}{\partial w_i}\sum_{j=0}^{M} w_j x^j \\
&= \frac{\partial}{\partial w_i}\Big(... + w_{i-1}x^{i-1} + w_i x^i + w_{i+1}x^{i+1} + ...\Big) \\
&= ... + \frac{\partial}{\partial w_i} w_{i-1}x^{i-1} + \frac{\partial}{\partial w_i} w_i x^i + \frac{\partial}{\partial w_i} w_{i+1}x^{i+1} + ... \\
&= ... + 0 + x^i + 0 + ... \\
\frac{\partial}{\partial w_i} y_n = \frac{\partial}{\partial w_i} y(x_n,\mathbf{w}) &= x^i
\end{aligned}
$$

And we can take the derivative of $y(x_n, \mathbf{w})y(x_n, \mathbf{w})$ using the product rule, where

$$
\begin{aligned}
\frac{\partial}{\partial w_i}\big[y(x_n,\mathbf{w})y(x_n,\mathbf{w})\big] &= \frac{\partial}{\partial w_i} y(x_n,\mathbf{w}) \cdot y(x_n,\mathbf{w}) + y(x_n,\mathbf{w}) \cdot \frac{\partial}{\partial w_i} y(x_n,\mathbf{w}) \\
&= 2\frac{\partial}{\partial w_i} y(x_n,\mathbf{w}) \cdot y(x_n,\mathbf{w}) \\
&= 2x^i y(x_n,\mathbf{w})
\end{aligned}
$$

Finally, taking $\frac{\partial}{\partial w_i}E(\mathbf{w})$ and setting it equal to 0 (where we are using the sum rule, $(f + g)' = f' + g'$, to take the derivatives on the inside of the summations),

$$
\begin{aligned}
\frac{\partial}{\partial w_i} E(x_n, \mathbf{w}^*) = 0 = \frac{1}{2} \sum_{n=1}^{N} \frac{\partial}{\partial w_i} \left[y(x_n, \mathbf{w}^*) y(x_n, \mathbf{w}^*) \right] - \sum_{n=1}^{N} \frac{\partial}{\partial w_i} y(x_n, \mathbf{w}^*) t_n + \frac{1}{2} \sum_{n=1}^{N} \frac{\partial}{\partial w_i} t_n^2 \\
0 = \frac{1}{2} \sum_{n=1}^{N} 2x_n^i y(x_n, \mathbf{w}^*) - \sum_{n=1}^{N} x_n^i t_n + 0 \\
= \sum_{n=1}^{N} x_n^i y(x_n, \mathbf{w}^*) - \sum_{n=1}^{N} x_n^i t_n \\
\sum_{n=1}^{N} x_n^i t_n = \sum_{n=1}^{N} x_n^i \sum_{j=0}^{M} w_j^* x_n^j \\
\sum_{n=1}^{N} x_n^i t_n = \sum_{n=1}^{N} \sum_{j=0}^{M} w_j^* (x_n)^{i+j} \\
\sum_{n=1}^{N} x_n^i t_n = \sum_{j=0}^{M} w_j^* \sum_{n=1}^{N} (x_n)^{i+j}
\end{aligned}
$$

We can re-arrange the above and represent the equations with vectors and matrices in the form of $\mathbf{Aw} = \mathbf{b}$, where

$$
\begin{aligned}
\sum_{j=0}^{M} w_j^* \underbrace{\sum_{n=1}^{N} (x_n)^{i+j}}_{a_{ij}} = \underbrace{\sum_{n=1}^{N} x_n^i t_n}_{b_i} \\
\sum_{j=0}^{M} a_{ij} w_j^* = b_i \\
\text{where } i \text{ are the vectors' index, from } 0..M \rightarrow \mathbf{Aw}^* = \mathbf{b} \\
\rightarrow \mathbf{w}^* = \mathbf{A}^{-1}\mathbf{b}
\end{aligned}
$$

To solve this efficiently, we can use matrix algebra to create $\mathbf{A}$ and $\mathbf{b}$ using a $N \times M + 1$ matrix $\mathbf{X}$, where

$$
\mathbf{X} = \begin{bmatrix} x_1^0 & x_1^1 & ... & x_1^M \\ x_2^0 & x_2^1 & ... & x_2^M \\ \vdots & \vdots & ... & \vdots \\ x_N^0 & x_N^1 & ... & x_N^M \end{bmatrix}, \boldsymbol{t} = \begin{bmatrix} t_1 & ... & t_N \end{bmatrix}
$$
$$
\text{and } \mathbf{A} = \mathbf{X}^\top \mathbf{X}, \ \mathbf{b} = \boldsymbol{t}\mathbf{X}
$$

.

There's also a problem when choosing the degree of the polynomial, M. A high M can lead to over-fitting; we want to achieve a good *generalization.*

One way of testing the model is by comparison of $E(\mathbf{w}^*)$ across model parameters (e.g. size of M). A good way to incorparte different-sized datasets is via root-mean-squared error (RMS). With $E(\mathbf{w}^*) \equiv \frac{1}{2}\sum_{n=1}^{N}[y(x_n, \mathbf{w}^*) - t_n]^2$,

$$
E_{RMS} = \sqrt{2E(\mathbf{w}^*)/N} \tag{3}
$$