FAQ / Crib Notes to help with Symbols in Data Mining

October 18, 2014

1 Symbols & Linear Algebra

Usually:

- Regular symbols x, y, z mean scalars such as $1, 2.5, -300$.
- Bold symbols $\mathbf{x}, \mathbf{y}$ mean vectors such as $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$, $\begin{bmatrix} 5 \\ -10 \\ 3.5 \end{bmatrix}$. By convention the vectors stand upright.
  - Bold symbol with transpose sign$\mathbf{x}'$ or $\mathbf{x}^T$, means vectors lie down, e.g., $[3, 4], [5, -10, 3.5]$.
  - A scalar is a special case of a vector (length 1 vector)
- Big letters X, Y usually mean a matrix.
  - A vector is a special case of a matrix (1 row matrix)
  - A scalar is a special case of a matrix (1 row, 1 col)
- The sum symbol $\sum_i x_i$ means add up all the x_i for every i. E.g., if $\mathbf{x} = \begin{bmatrix} 5 \\ -10 \\ 3.5 \end{bmatrix}$, then $\sum_i x_i = 5+(-10)+3.5$. (Where x_i indexes the ith value in vector $\mathbf{x}$)
- Product symbol $\prod_i x_i$ means multiply all the x_i together for every i. E.g., if $\mathbf{x} = \begin{bmatrix} 5 \\ -10 \\ 3.5 \end{bmatrix}$, then $\prod_i x_i = 5 \times -10 \times 3.5$.
- The "inner product" $\mathbf{w}^T\mathbf{x}$ of two vectors is defined as $\sum_i w_i x_i$. The result of this is a scalar.
  - E.g., This is how a classifier or regressor takes a vector of weights $\mathbf{w}$ and data $\mathbf{x}$ and produces a single output value $y = \mathbf{w}^T\mathbf{x}$.
  - (Its common to use letter $\mathbf{x}$ to describe the input data to a problem, and y to describe the predicted / target prediction.)
  - A special case of this used in regularisation is $\mathbf{w}^T\mathbf{w} = \sum_i w_i w_i = \sum_i w_i^2$. I.e., the sum of the squared values of vector $\mathbf{w}$.
- $y = f(\mathbf{x})$ normally refers to a function that takes the vector $\mathbf{x}$ as input and gives y as output.

Optimization

- Think of statements like $argmin_{\mathbf{w}}(E(\mathbf{x}, \mathbf{w}))$ as short-hand for saying:
  - "Search all the values of $\mathbf{w}$ and check what the function $E(\mathbf{x}, \mathbf{w})$ returns for each. Return the value of $\mathbf{w}$ that made $E(\mathbf{x}, \mathbf{w})$ smallest."
  - How this is implemented in practice varies, but it's a general idea behind many operations. E.g., learning regression: finding a line with minimum misprediction error to some datapoints.

- In regression, a statement like $E = \sum_i^N (y_i - f_{\mathbf{w}}(\mathbf{x}_i))^2$ is saying:
  - For every row of data i:
    - Compute the difference between every target value y_i, and its associated prediction from the regressor f when using $\mathbf{x}_i$ as input and $\mathbf{w}$ as weights.
    - Square the difference to get rid of negative/positive differences.
  - Then add all the differences up to find out the total discrepancy between the regressor and the data points
- The *identity function* $\mathbf{I}$ returns one if its parameter is true, 0 if its false.
  - So a statement like $E = \frac{1}{N} \sum_i^N \mathbf{I}(y_i = f(\mathbf{x}_i))$ is saying: "Count up all the times that the prediction $f(\mathbf{x}_i)$ was the same as the true value y_i. (I.e., compute the correct prediction rate, or the accuracy)

Norms

- Bars around a vector mean various things...
- Plain$|\mathbf{x}|$ can be ambiguous, and context dependent.
  - Sometimes it means count the number of elements in the vector, so for $\mathbf{x} = \begin{bmatrix} 5 \\ -10 \\ 3.5 \end{bmatrix}$, $|\mathbf{x}| = 3$.
  - Sometimes it means absolute value, typically when the variable in between is a scalar, e.g., $|y - f(\mathbf{x})|$ means the absolute value of the difference between data point y and prediction $f(\mathbf{x})$.
- $|\mathbf{w}|_0$ means count the non-zero elements (L0 norm)
- $|\mathbf{w}|_1$ means add up all the absolute value elements $|\mathbf{w}|_1 = \sum |w_i|$ (L1 norm)
- $|\mathbf{w}|_2$ means add up all the square elements $|\mathbf{w}|_2 = \sum w_i^2$ (L2 norm)

2 Computational Complexity

When we talk about computational complexity, we use a "big O" notation. In this context, n typically means the number of data items (database rows), and d means the number of attributes (database columns). This is important because it gives a way to compare the scalability of algorithms in general terms, and be able to estimate run-time for a given dataset in advance.

- E.g., $O(1)$ Means the cost of the algorithm/operation depends on neither the amount of rows or columns. Most scalable, rarely happens.
- E.g., $O(d)$ Means the cost of the algorithm/operation is *proportional* to the amount of columns only (not rows, as n is not present). You know that doubling the rows/features will generally double the computation time.
- E.g., $O(nd)$ Means the cost of the algorithm is proportional to the product of the number of rows and columns. Doubling the number of rows or columns will double computation time. Doubling both will quadruple computation time.
- E.g., $O(n^2)$ means the cost of the algorithm is proportional to the product of the number of rows. Doubling the rows will quadruple the computation time. While $O(d^2)$ would mean for example that halving the number of columns would quarter the computation time.

The same notation can be applied for computation time as well as memory storage requirement.