3 Fundamental Algorithms (part 1 of 5)

Here are popular supervised learning algorithms trained on \(\{(x_i, y_i)\}_{i=1}^{N}\) (where \(x = [x_1, \ldots, x_D]\)), each useful directly or as a part of a more complex algorithm.

Linear regression

- **Linear regression** models \(y \in \mathbb{R}\) as a linear combination of the features in \(x\): \(\hat{y} \leftarrow f_{w,b}(x) = wx + b\). Draw the case where \(D = 1\), that is, \(x\) is 1D:
  - \((x_i, y_i)\): \(i^{th}\) example
  - \(\hat{y} = wx + b\): regression line
    * \(w\): slope
    * \(b\): intercept
  - \(\hat{y}_i = wx_i + b\): predicted \(y\) for \(x = x_i\)
  - \(\hat{y}_i - y_i\): difference

Now we consider any \(D \geq 1\) (simple linear regression or multiple linear regression).

- Find optimal \([w^*, b^*]\) that minimize the objective function \(\frac{1}{N} \sum_{i=1}^{N} [f_{w,b}(x_i) - y_i]^2\) (also called the mean squared error (MSE)). Alternatives to MSE include:
  - mean error: \(\frac{1}{N} \sum_{i=1}^{N} |f_{w,b}(x_i) - y_i|\), as any line through centroid of two points works equally well
  - mean absolute error, \(\frac{1}{N} \sum_{i=1}^{N} |f_{w,b}(x_i) - y_i|\): useful, but derivative

We predict \(y\) from \(x\), so minimize vertical difference in the “least squares” sense.

- \([f_{w,b}(x_i) - y_i]^2\), a **loss function**, penalizes difference of example \(i\). We minimize a **cost function** given by the average loss (MSE, above) of all penalties from applying the model to the training data.

- Minimize MSE by setting gradient (vector of partial derivatives) to \(\frac{\partial}{\partial w}\). Use matrix notation:

\[
\begin{align*}
X &= 
\begin{bmatrix}
1 & x_{11} & \cdots & x_{1j} & \cdots & x_{1D} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
1 & x_{i1} & \cdots & x_{ij} & \cdots & x_{iD} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
1 & x_{N1} & \cdots & x_{Nj} & \cdots & x_{ND}
\end{bmatrix}

&= 
\begin{bmatrix}
x_{10} & x_{11} & \cdots & x_{1j} & \cdots & x_{1D} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
x_{i0} & x_{i1} & \cdots & x_{ij} & \cdots & x_{iD} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
x_{N0} & x_{N1} & \cdots & x_{Nj} & \cdots & x_{ND}
\end{bmatrix}
\end{align*}
\]
This design matrix $X$ is the training examples (without their labels $\{y_i\}$) preceded by a column of ones for convenience: we understand $x_{i0} \equiv \ldots$ for $i = 1, \ldots, N$.

$$w = \begin{bmatrix} w_0 = b \\ w_1 \\ \vdots \\ w_i \\ \vdots \\ w_D \end{bmatrix}_{(1+D) \times 1}$$

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_N \end{bmatrix}_{N \times 1}$$

and

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_i \\ \vdots \\ \hat{y}_N \end{bmatrix}_{N \times 1}$$

The system of equations, $\hat{y}_i = wx_i + b$ for $i = 1, \ldots, N$ is expressed as $\hat{y} = Xw$. We minimize

$$MSE = \frac{1}{N} \sum_{i=1}^{N} [f_{w,b}(x_i) - y_i]^2$$

$$= \frac{1}{N} \sum_{i=1}^{N} [\hat{y}_i - y_i]^2$$

which is also equal to $\frac{1}{N} ||\hat{y} - y||^2 = \frac{1}{N} ||Xw - y||^2$

$$= \frac{1}{N} \sum_{i=1}^{N} \left( y_i - \sum_{j=0}^{D} x_{ij}w_j \right)^2$$

by setting to zero the partial derivative with respect to $w_k$ for each $k = 0, \ldots, D$:

$$\frac{\partial}{\partial w_k} (MSE) = \frac{1}{N} \sum_{i=1}^{N} 2 \left( y_i - \sum_{j=0}^{D} x_{ij}w_j \right) (-x_{ik})$$

$$= 0$$

$$\implies \sum_{i=1}^{N} y_ix_{ik} = \sum_{i=1}^{N} \sum_{j=0}^{D} w_jx_{ij}x_{ik}$$

$$\implies \sum_{i=1}^{N} X^T_{ki}y_i = \sum_{j=0}^{D} \left( \sum_{i=1}^{N} X^T_{ki}x_{ij} \right) w_j$$

$$\implies [X^Ty]_k = \sum_{j=0}^{D} (X^TX)_{kj}w_j$$

$$= [(X^TX)w]_k$$

$^1$This is an abuse of notation: I moved $b$ into $w$ as $w_0$ and added an $x_{0}(i) = 1$ element to each $x$ feature vector.
This is true for each $k$, so we can write $X^T y = (X^T X) w$. To solve for $w$, multiply both sides on the left by $(X^T X)^{-1}$: $w = (X^T X)^{-1} X^T y$.

e.g. Use $w = (X^T X)^{-1} X^T y$ to find the regression line for the points $(1, 1), (2, 3), (3, 2)$.

\begin{itemize}
  \item \textit{\textsuperscript{2}What did one regression coefficient say to the other?}
  \item \textit{\textsuperscript{3}Scikit-learn does not invert the matrix, instead using a faster, more stable algorithm that solves $X^T y = (X^T X) w$.}
\end{itemize}
Python

- `from sklearn import linear_model` loads the `linear_model` module
- `model = linear_model.LinearRegression()` gives the model
- `model.fit(X, y)` fits the model to array $X_{N \times D}$ and $y_{N \times 1}$
- `model.coef_` gives $w^*$ and `model.intercept_` gives $b^*$
- `model.predict(X)` gives predictions for examples in $X$
- `model.score(X, y)` gives $R^2$, the coefficient of determination from statistics, that is the proportion (in $[0, 1]$) of variability in $y$ accounted for by $X$ via the linear model.

To learn more:


---