Lecture 6: MATH 342W: Introduction to Data Science and Machine Learning

Sergio E. Garcia Tapia[*]

February 13, 2025 (last updated March 8, 2025)

Recap of Classification

Throughout our study, we have focused on the problem of binary classification, where there are two possible responses. That is, the response space is $\mathcal{Y} = \{0, 1\}$, and this is the context in which the *Perceptron Learning Algorithm* (PLA) and the *Support Vector Machine* (SVM) operate. A more general version of this problem allows for L different classifications, so that $\mathcal{Y} = \{c_1, \ldots, c_L\}$, for some $L \in \mathbb{N}$, a problem known as **multinomial classification**. Due to lack of time, we will not be exploring this.

K Nearest Neighbors

Given a data set $\mathbb{D}$, recall that our motivation for obtaining $g = \mathcal{A}(\mathbb{D}, \mathcal{H})$ is to make predictions. That is, if $\mathcal{X}$ is our feature space, and we see an input $\mathbf{x}_* \in \mathcal{X}$ that does not belong to $\mathbb{D}$, then we wish to predict the response as $g(\mathbf{x}_*)$.

One way to predict the response for $\mathbf{x}_*$ is to find the *closest* input $\mathbf{x}_k$ in $\mathbb{D}$ and to assign the response corresponding to $\mathbf{x}_k$ as the prediction using the input $\mathbf{x}^*$. This is called the **nearest neighbor model**. That is, we let $g(\mathbf{x}_*) = y_k$. The meaning of *closest* then becomes a hyperparameter, which is a *distance function*:

$$d : \mathcal{X} \times \mathcal{X} \to [0, \infty)$$

$$d(\mathbf{u}, \mathbf{v}) = \begin{cases} 0 & \text{if } \mathbf{u} = \mathbf{v} \\ \text{positive.} & \text{otherwise.} \end{cases}$$

If $\mathcal{X}$ is a subset of $\mathbb{R}^p$, then a reasonable default is to let $d = \|\cdot\|_2$, the vector 2-norm:

$$d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2 = \sqrt{\sum_{j=1}^{p} (u_j - v_j)^2}$$

The main difficulty with this is that it is sensitive to the scale of the units of the inputs. For example, we get vastly different values when using nanometers vs. kilometers. This problem is typically addressed by normalizing to interval data, writing $x' = \frac{x - \bar{x}}{s}$. This adds a pre-processing step on our data set $\mathbb{D}$.

[*]Based on lectures of Dr. Adam Kapelner at Queens College. See also the course GitHub page.

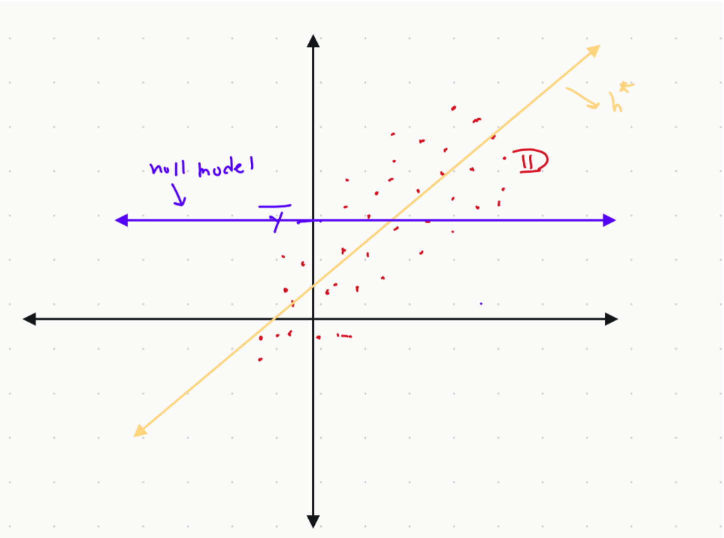

Figure 1: A scatterplot depicting a data set $\mathbb{D}$, the null model g_0, and h^*.

Another issue is this: what if one of our sample points behaves strangely? For example, suppose that our data set contains information about credit history for people and whether they defaulted on a loan they took out. Though unlikely, one of our pairs on $(\mathbf{x}_i, y_i)$ in $\mathbb{D}$ might correspond to a person whose observation $\mathbf{x}_i$ describes them as having a perfect credit history, but whose response y_i was to default on their loan. What happens when a person $\mathbf{x}_*$ comes in with a perfect credit score, or more specifically, with features very similar to $\mathbf{x}_i$? The approach we've described would determine that $\mathbf{x}_i$ is the closest point to $\mathbf{x}_*$, and would assign $g(\mathbf{x}_*) = y_i$. This is problematic, for example, if our data set contains many other points similar to $\mathbf{x}_i$ that had a different response, and perhaps one of those responses would have been more appropriate for $g(\mathbf{x}_*)$. To address this difficulty, we take not the closest $\mathbf{x}_i$, but the closest k points $\mathbf{u}_1, \ldots, \mathbf{u}_k$ (this is some permutation of the $\mathbf{x}$'s in $\mathbb{D}$), and take the mode:

$$g(\mathbf{x}_*) = \text{Mode}[g(\mathbf{u}_1), \ldots, g(\mathbf{u}_k)]$$

This approach is called the **k-nearest neighbors** (KNN) model. This concludes (for now) our discussion of classification modeling.

Regression Modeling

Suppose the response space is an interval or it's a "numeric" response, meaning $\mathcal{Y} \subseteq \mathbb{R}$. Models with such a response space are called **"regression" models** (for historical reasons only). If $\mathcal{X} = \mathbb{R}$, then $\mathbb{D}$ is a set of points in the plane, which can be depicted with a *scatterplot* (see Figure 1).

If we do not know or see any features, the *null model* in this context is the *average*:

$$g_0 = \bar{y} \tag{1}$$

Recall the null model is meant to serve as a reference for performance. How can we beat it? A simple starting point is to let the set of candidate functions be the set of hyperplanes:

$$\mathcal{H} = \{\mathbf{w} \cdot \mathbf{x} = 0 : \mathbf{w} \in \mathbb{R}^{p+1}, \mathbf{x} \in \{1\} \times \mathbb{R}^p\},$$

where $\mathbf{x}$ has been extended to length $p+1$ by pre-pending a 1 entry. If the data set $\mathbb{D}$ does not suggest a linear pattern, we may see a high amount of misspecification error. Nevertheless, we will study this approach. The best candidate function h^* is given by

$$\begin{aligned} h^*(\mathbf{x}) &= \beta \cdot \mathbf{x} + \mathcal{E} \\ &= \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \mathcal{E} \end{aligned}$$

The vector $\beta \in \mathbb{R}^{p+1}$ consists of the optimal weights that define a hyperplane that performs best with respect to the data set $\mathbb{D}$. The quantity $\mathcal{E}$ is the error or noise. However, our algorithm will produce g, not h^*. In binary classification, we quantified the error as misclassification error. Here, we have a few choices:

$$\begin{aligned} SSE &:= \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad \text{(Sum of Squared Errors)} \\ SAE &:= \sum_{i=1}^{n} |e_i| = \sum_{i=1}^{n} |y_i - \hat{y}_i| \quad \text{(Sum of Absolute Errors)} \end{aligned}$$

The SSE is the default for practical reasons, as we will see. An algorithm that attempts to find the optimal $\mathbf{w} \in \mathbb{R}^{p+1}$, call it $\mathbf{b}$, that defines a hyperplane will attempt to minimize the SSE:

$$\mathcal{A} : \mathbf{b} = \underset{\mathbf{w} \in \mathbb{R}^{p+1}}{\operatorname{argmin}} \{SSE(\mathbf{w})\}$$

It looks difficult, but this is why we chose SSE; this has an analytical solution. We will focus on the case where $p = 1$, and we will derive the general case in the future.

Ordinary Least Squares Model (OLS)

If $p = 1$, then $\mathbf{b} \in \mathbb{R}^2$, and

$$\hat{y} = g(x) = \mathbf{b} \cdot \mathbf{x} = \mathbf{b}^\top \mathbf{x} = \begin{bmatrix} b_0 & b_1 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} = b_0 + b_1 x$$

where b_0 is the y-intercept and b_1 is the slope. The objective function that we are trying to minimize is

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i)^2$$

This is a function of two variables w_0 and w_1, whose optimal choice will be $w_0 = b_0$ and $w_1 = b_1$ (recall that we know x_i and y_i, since they make up $\mathbb{D}$). Since w_0 and w_1 are unknowns, SSE is a function of w_0 and w_1. To minimize the SSE, we take the partial derivative with respect to each independent variable and set it to zero:

$$\begin{aligned}
\frac{\partial}{\partial w_0}[SSE] &= \frac{\partial}{\partial w_0}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2 \\
&= \sum_{i=1}^{n}\frac{\partial}{\partial w_0}(y_i - w_0 - w_1 x_i)^2 && \text{(by the linearity of the derivative)} \\
&= -2\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i) && \text{(by the Chain Rule)} \\
&= -2\left[\sum_{i=1}^{n} y_i - n w_0 - w_1 \sum_{i=1}^{n} x_i\right]
\end{aligned}$$

First we'll simplify notation by using the average of the x's and y's:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \quad \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

Now using this, we set the partial derivative to zero to find an extremum. We will denote b_0 and b_1 as the values of the parameters w_0 and w_1 that allows the SSE to achieve its extremum value:

$$\begin{aligned}
0 &= \frac{\partial}{\partial w_0}[SSE_0] \\
0 &= -2\left[\sum_{i=1}^{n} y_i - n b_0 - b_1 \sum_{i=1}^{n} x_i\right] \\
0 &= -2(n\bar{y} - n b_0 - b_1 n \bar{x}) \\
b_0 &= \bar{y} - b_1 \bar{x} \qquad (2)
\end{aligned}$$

Since we have two variables b_0 and b_1, we need another equation, which we get by taking the partial derivative with respect to w_1. Once again, applying the Chain Rule and using the same notational simplification from earlier:

$$\begin{aligned}
\frac{\partial}{\partial w_1}[SSE] &= \frac{\partial}{\partial w_1}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2 \\
&= \sum_{i=1}^{n} -2x_i(y_i - w_0 - w_1 x_i) \\
&= -2\left[\sum_{i=1}^{n} x_i y_i - n w_0 \bar{x} - w_1 \sum_{i=1}^{n} x_i^2\right]
\end{aligned}$$

Setting the partial derivative to 0 yields and using Equation (2):

$$\begin{aligned}
0 &= \frac{\partial}{\partial w_1}[SSE] \\
0 &= -2\left[\sum_{i=1}^{n} x_i y_i - n b_0 \bar{x} - b_1 \sum_{i=1}^{n} x_i^2\right] \\
0 &= \sum_{i=1}^{n} x_i y_i - n b_0 \bar{x} - b_1 \sum_{i=1}^{n} x_i^2 \\
0 &= \sum_{i=1}^{n} x_i y_i - n \bar{x}\bar{y} + n b_1 \bar{x}^2 - b_1 \sum_{i=1}^{n} x_i^2 && \text{(using Equation 2)} \\
b_1\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right) &= \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} \\
b_1 &= \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} \qquad (3)
\end{aligned}$$

We can substitute this into Equation (2) to get an expression for b_0. However, before that, we will change into a notation that is more compact, using language and notation from probability theory and statistics. Let

$$\rho := \mathrm{Corr}[X,Y] := \frac{\mathrm{Cov}[X,Y]}{\mathrm{SD}[X]\cdot\mathrm{SD}[Y]} = \frac{\mathrm{Cov}[X,Y]}{\sqrt{\mathrm{Var}[X]\cdot\mathrm{Var}[Y]}} := \frac{\mathrm{E}[(X-\mu_X)(Y-\mu_Y)]}{\sqrt{\mathrm{Var}[X]\cdot\mathrm{Var}[Y]}}$$

where Corr stands for correlation, Cov stands for covariance, SD stands for standard deviation, Var stands for variance, and E stands for expectation. The estimate for ρ is

$$\begin{aligned}
r &:= \frac{S_{xy}}{S_x S_y}, \\
S_{xy} &:= \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y}), \\
S_x^2 &:= \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2
\end{aligned}$$

where r is the **sample correlation**. Here, S_{xy} is an estimate for the covariance, and S_x^2 is an estimate of variance. If we expand $S_{x,y}$, we get

$$\begin{aligned}
S_x^2 &= \frac{\sum_{i=1}^{n} x_i y_i - \bar{y}\sum_{i=1}^{n} x_i - \bar{x}\sum_{i=1}^{n} y_i + \bar{x}\bar{y}}{n-1} \\
&= \frac{\sum_{i=1}^{n} x_i y_i - \bar{y}\bar{x} - \bar{x}\bar{y} + \bar{x}\bar{y}}{n-1} \\
&= \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{n-1}
\end{aligned}$$

This looks like the numerator of b_1. Next, expanding S_x^2:

$$S_x^2 = \frac{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}{n-1}$$

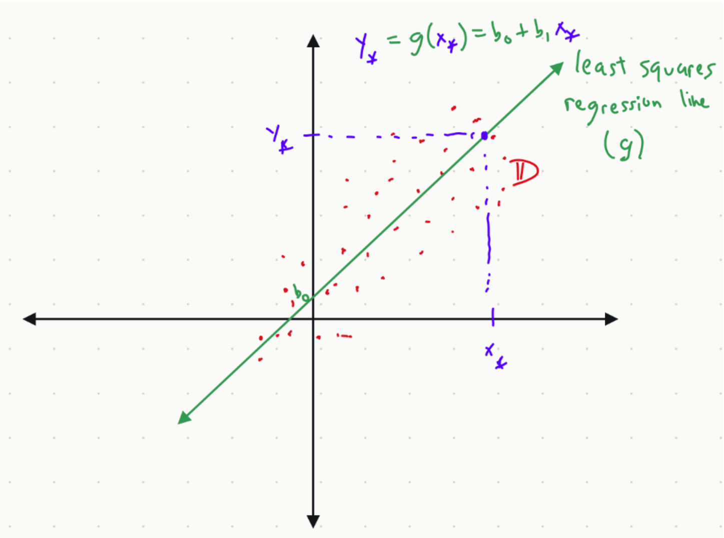
$Y_* = g(x_*) = b_0 + b_1 x_*$
least squares regression line (g)
Y_*
$\mathbb{D}$
b_0
x_*

Figure 2: The least squares regression line for a data set $\mathbb{D}$.

which looks like the denominator of b_1. Now:

$$\begin{aligned}
b_1 &= \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} \\
&= \frac{(n-1)S_{xy}}{(n-1)S_x^2} = \frac{S_{xy}}{S_x^2} = \frac{rS_xS_y}{S_x^2} \\
&= r\frac{S_y}{S_x}
\end{aligned}$$

All of these formulas are valid and equivalent, though some are more pervasive than others. In short, we have the **simple/univariate least squares regression model**:

$$b_0 = \bar{y} - r\frac{S_y}{S_x}\bar{x} \tag{4}$$

$$b_1 = r\frac{S_y}{S_x} \tag{5}$$

The significance is that $g(x_*) = b_0 + b_1 x_*$, which is an analytic solution (unlike our previous algorithms that required using an optimization from a package such as `optimx` in R). The parameters b_0 and b_1 define the y-intercept and slope, respectively, of the **least squares regression line** (think of a line of best fit, see Figure 2.)

Assessing the Quality of the Regression Model (Performance Metrics)

To see how well this model performs, we have the following options:

(1) SSE: The units of this quantity are response units squared. A disadvantage is that squared units are generally not intuitive (for example, what would squared dollars describe?). Also, SSE is sensitive to n; as n increases, the SSE also increases with n.

(2) MSE: One way to handle the scaling issue of SSE is to use the *mean squared error*:

$$MSE := \frac{1}{n-2} SSE$$

Now this no longer scales with n, but it stills suffers from a lack of interpretability due to having squared units.

(3) $RMSE$: To get back to the response units, we can take the squared root:

$$RMSE := \sqrt{MSE}$$

This is the *root mean squared error*. This addresses both the scaling issue and the units.