Lab 1A: Data, Models and Decisions

Juan Elenter, Ignacio Hounie and Alejandro Ribeiro*

August 26, 2024

1 Systems

A system is an entity that processes some inputs to produce some outputs. In engineering practice, inputs and outputs are often quantified and for this reason we denote inputs with a variable x, outputs with a variable y, and the system itself is represented by the function $f(\cdot)$ that codifies the relationship $y = f(x)$. A schema is shown in Figure 1.

This definition is vague on purpose because it is intended to maintain generality. Pretty much any task can be codified as the action of a system. For instance, when we hear a word and understand its meaning we are acting as a system that takes a time varying sequence of air pressures as inputs and produces a categorical representation as an output. When we read a digit, we are a system that takes light patterns as inputs and produces numbers as outputs – a number being the common property of finite sets that can be related with a bijection. When we watch and rate a movie, we are a system that takes movies as inputs and produces ratings as outputs. Driving a car around a circuit requires a driver that takes as input the desired trajectory and produces a sequence of acceleration inputs that results in the car following the circuit. The driver is a system.

Just as important, this definition of a system is vague because we can keep it vague and yet make it useful. Despite their significant differences all of the systems that we describe above can be represented by the schema

*In alphabetical order.

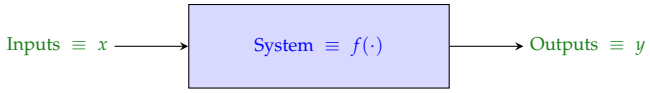

Figure 1. Systems. A system $f(\cdot)$ is an entity that processes inputs x to produce outputs $y = f(x)$. This definition is vague so as to yield a wide range of specific instances. However general, it abstracts properties that make it useful.

in Figure 1. They all process some input – audio, light, movies, or target trajectories – to produce some output – concepts, numbers, ratings, or acceleration sequences. They are, in fact, four systems that we will study in this course.

2 Artificial Intelligence

A first definition of an artificial intelligence (AI) is that of a system that mimics the input-output relationship of a natural system; see Figure 2. When the natural system is presented with the input x it responds by producing the output y. When the AI is presented with the same input x it responds by producing the output $\hat{y}$. If the AI is a good AI, the outputs it produces in response to a given input are similar to the outputs produced by the natural system.

Having an AI is useful because it can be used in lieu of the natural system. For instance, suppose that the input x represents an image of a digit. The natural system is a standard human that looks at this image and reads the number y that it represents. If the AI can successfully mimic a human by spitting numbers $\hat{y}$ that are equal to the numbers recognized by a human reader we can use it in lieu of humans to recognize digits. This is good because the AI frees humans from the drudgery of reading digits. AI's have been in use since the 1990's to recognize digits in checks.

We must point out that the schema in Figure 2 is a flawed definition of AI. A 100 gram tungsten ball dropped from 1 meter mimics quite well any other tungsten ball dropped from 1 meter. Some would say that the definition is less flawed if we add the restriction that the emulated natural system is intelligent but this begs the question of what it means for a natural system to be intelligent. Besides, an artificial intelligence can

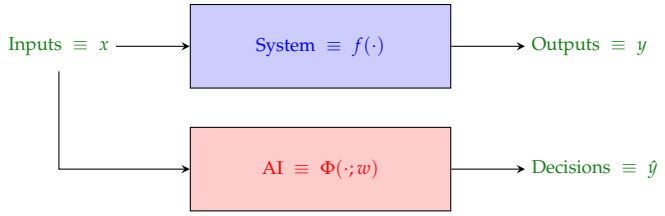

Figure 2. Artificial Intelligence. An artificial intelligence (AI) is a system that mimics the input-output relationship of a natural system. This is a flawed definitions which is nonetheless a good operational definition of the practice of AI.

still be useful if the system that it imitates is not intelligent.

Flawed or not, the most important fact is that Figure 2 is a good operational definition which captures well the current practice of AI.

2.1 Data, Models, and Decisions

To design an AI system the first step is to acquire *data*. This is typically in the form of a set of N input-output pairs (x_i, y_i). We call this collection of examples the training set.

The next step is the selection of a *model*. This is usually in the form of a postulated relationship between inputs and outputs. This is written in Figure 2 as the function $\Phi(\cdot, w)$. The choice of model follows from our knowledge and understanding of the system. For example, convolutional neural networks (CNNs) have invariance and stability properties that make them adequate to process times series and images – as we will see in Labs 2 and 4. The choice of model also follows from accumulated empirical evidence of which models are known to work for specific kinds of systems.

As per Figure 2, when the AI is presented with the input x it produces the output $\hat{y} = \Phi(x, w)$. This output is the AI's estimate or prediction of the actual output y that the system produces when presented with input x. We also say that $\hat{y}$ is the AI's *decision*.

2.2 Machine Learning

In the AI's decisions $\hat{y} = \Phi(x, w)$ the variable w is a parameter that has to be chosen. To choose this parameter we introduce a metric $\ell(y, \hat{y})$ to compare predicted outputs $\hat{y}$ with actual outputs y. We then search for the parameter w that minimizes this loss over the given set of input output pairs,

$$w^* = \underset{w}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \ell\Big(y_i, \Phi(x_i, w) \Big). \tag{1}$$

We call this formulation a supervised machine learning problem (ML). The process of finding the parameter w^* that minimizes the loss averaged over the available data is called training.

We call (1) a supervised learning problem because the AI is given examples of inputs x_i and their corresponding outputs y_i. Alternatively, we may be given example inputs and a cost function $c(\cdot)$ that assesses the merit of the AI decision $\Phi(x_i, w)$. In this case we formulate the unsupervised ML problem,

$$w^* = \underset{w}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} c\Big(\Phi(x_i, w) \Big). \tag{2}$$

We must point out that (1) and (2) are controversial definitions of learning. There is nothing in the specifications to represent understanding, although some people argue that sufficiently complex imitation *is* understanding. As we did with the definition of AI, we remain agnostic to this discussion. Equations (1) and (2) are good operational definitions of learning which capture well the current practice of ML.

3 Admissions at the University of Pennsylvania

Let us pretend that we are tasked with designing a system to make admission decisions at the University of Pennsylvania (Penn). In order to design this system we need to acquire data, choose a model, and train it to make admission decisions.

ID	HS GPA	SAT	Gender	Penn GPA
A41675	3.93	1,540	F	3.67
CE58D3	3.79	1,540	M	3.53
C38C00	3.93	1,560	F	3.80
CFA232	3.90	1,570	M	3.53
B57BF6	3.95	1,500	F	3.67
CA694D	3.76	1,520	M	3.40

Figure 3. Grade point average (GPA) and Scholastic Assessment Test (SAT) data samples.

3.1 Data

Figure 3 shows data that we have available to make admission decisions. For a collection of *former* students we have access to their high school (HS) grade point average (GPA), their Scholastic Assessment Test (SAT) scores, their gender and their Penn GPA.

The table shows five representative examples, but we have data for a total of 600 students. Figure 4 shows plots in which the horizontal axes are high school GPAs or SAT scores and the vertical axes are Penn GPAs. These plots show that high school GPA is predictive of Penn GPA. Although there is significant variation we can see that higher high school GPA corresponds with higher Penn GPA. This indicates that high school GPAs are useful information for admission decisions as they can predict with some accuracy the GPA that an admitted student may attain at Penn. We can squint and see that the same is more or less true of SAT scores, although the correlation between SAT scores and Penn GPA is weaker.

Task 1 Follow this link to download the GPA and SAT score data. Reproduce the plots in Figure 4. ■

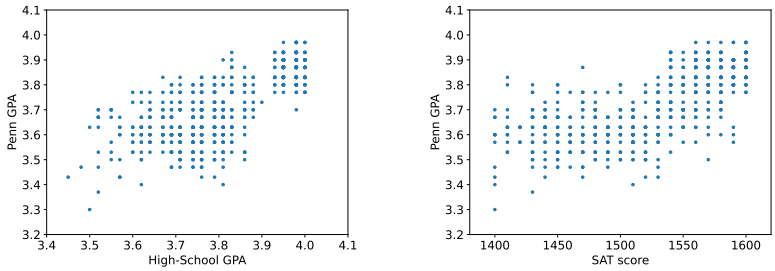

Figure 4. Penn GPA plotted with respect to high school GPA and SAT scores.

3.2 System

To make admission decisions we interpret Penn as the system shown in Figure 5. This system takes as inputs high school GPA, SAT scores and gender information of a student and produces as an output the Penn GPA of the corresponding graduate. If we define the input data as a vector $x = [\text{HS GPA}; \text{SAT}; \text{Gender}]$ and we denote the output as $y = \text{Penn GPA}$ we can represent this system as the function,

$$\text{Penn GPA} \;=\; y \;=\; P(x) \;=\; P\begin{bmatrix} \text{HS GPA} \\ \text{SAT} \\ \text{Gender} \end{bmatrix}. \tag{3}$$

Notice that this is a poor representation of Penn. Incoming students are much more than their gender, High School GPA, and SAT scores. Penn graduates are much more than their Penn GPAs and the institution itself does much more to a high school graduate than transforming their High School GPA and SAT scores into a Penn GPA. This is just one aspect of the whole system on which we are choosing to focus. The distinction between what a system is and what an engineer chooses to say that a system is warrants some discussion that we undertake in Section 5.

4 Model and Decisions

To make admission decisions we leverage the system of Section 3.2 and the data of Section 3.1 to design an AI model that *predicts* the Penn GPA

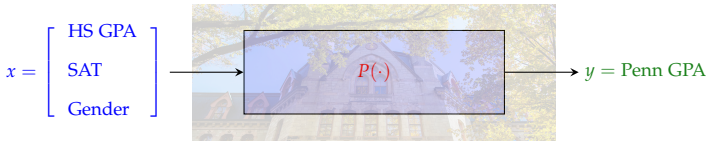

Figure 5. The University of Pennsylvania.

of prospective students.

To make matters simpler let us begin by ignoring SAT scores and gender and attempt predictions based on high school GPAs. This means that the system in (4) is replaced by the system

$$\text{Penn GPA} \;=\; y \;=\; P(x) \;=\; P(\,\text{HS GPA}\,). \tag{4}$$

The function $P(x)$ is the true effect of Penn on scholastic accomplishment. This is information that becomes available after the fact. When a student graduates Penn, we have access to their high school GPA x and their Penn GPA y.

Penn GPA predictions are to be made prior the fact. Before a student attends Penn we want to estimate their Penn GPA based on their high school GPA x. We *choose* to postulate a linear relationship and make predictions of the form

$$\hat{y} = \alpha x. \tag{5}$$

In (5), $\hat{y}$ is a *prediction* of the *true* Penn GPA y that will be available after the fact. The coefficient α is to be determined with the goal of making predictions $\hat{y}$ close to to actual Penn GPA y. We can then use Penn GPA predictions to make admission *decisions.*

4.1 Least Squares Estimation

To determine a proper value for the coefficient α in (5) we utilize the data we have available on the scholastic performance of past students. Use N to denote the total number of available data points. Introduce a subindex i to differentiate past students so that the pair (x_i, y_i) denotes the high school GPA and Penn GPA of student i. For these students we can make GPA *predictions* $\hat{y}_i = \alpha x_i$. For a given coefficient α we define the mean

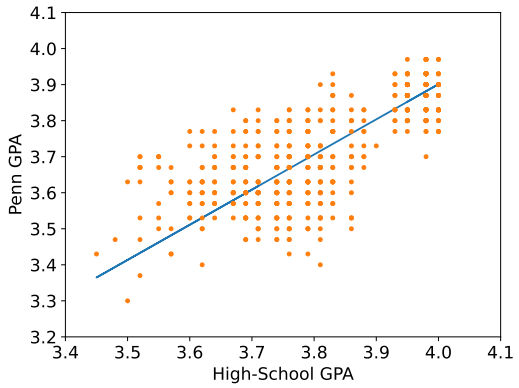

Figure 6. Linear minimum mean squared error (MMSE) prediction of Penn GPA from high school GPA.

squared error (MSE),

$$\text{MSE}(\alpha) \;=\; \frac{1}{N}\sum_{i=1}^{N}\Big(y_i - \hat{y}_i\Big)^2 \;=\; \frac{1}{N}\sum_{i=1}^{N}\Big(y_i - \alpha x_i\Big)^2. \tag{6}$$

The mean squared error $\text{MSE}(\alpha)$ measures the predictive power of coefficient α. The quantity $(y_i - \hat{y}_i)^2$ is always nonnegative and indicates how good the predicted GPA $\hat{y}_i$ is to the true GPA y_i. The MSE averages this metric over all students. It follows that a natural choice for α is the value that makes the MSE smallest. We therefore define the optimal coefficient

$$\alpha^* \;=\; \underset{\alpha}{\text{argmin}}\,\frac{1}{2}\text{MSE}(\alpha) \;=\; \underset{\alpha}{\text{argmin}}\,\frac{1}{2N}\sum_{i=1}^{N}\Big(y_i - \alpha x_i\Big)^2, \tag{7}$$

and proceed to make Penn GPA predictions as $\hat{y} = \alpha^* x$ [cf. (5)]. This GPA predictor is called the linear minimum mean squared error (MMSE) prediction. This is because the predictor is the linear function that minimizes the MSE.

Task 2 Prove that the MMSE estimator coefficient α^* defined in (7) is given by the expression

$$\alpha^* \;=\; \sum_{i=1}^{N} x_i y_i \;\Big/\; \sum_{i=1}^{N} x_i^2. \tag{8}$$

Compute α^* for the data loaded in Task 1. Plot the Penn GPA with respect to HS GPA and superimpose the prediction line $\hat{y} = \alpha^* x$. This plot is shown in Figure 6.

In Task 2 we make predictions of the Penn GPA of students that have graduated Penn. Predicting Penn GPAs of past students is unnecessary given that we already know their true GPAs. Our motivation for solving this unnecessary problem is to determine the coefficient α^* that we can use to make predictions $\hat{y} = \alpha^* x$ of students that have not yet attended Penn – for which x is available but y is not. The effectiveness of this prediction depends on the extent to which the past is a good representation of the future.

It is germane to emphasize that in Task 2 we are using something we know – the GPA of former students – to answer a new question – the GPA of a prospective student. However primitive, this is a form of intelligence.

4.2 Root Mean Squared Error

We evaluate the merit of α^* with the root mean squared error (RMSE)

$$\text{MSE}(\alpha) \;=\; \sqrt{\text{MSE}(\alpha)} \;=\; \left[\frac{1}{N} \sum_{i=1}^{N} \Big(y_i - \alpha x_i \Big)^2 \right]^{1/2} . \tag{9}$$

The reason we use the RMSE to evaluate the merit of α^* instead of the MSE is that the RMSE has the same units. It is easier to interpret than the MSE. Since the difference between the two is just a square root function, the coefficient α^* that minimizes the MSE also minimizes the RMSE.

Task 3 Compute the RMSE of α^* and comment on the quality of the Penn GPA predictions. ■

You should observe that the RMSE is 0.094. We can think of this number as the accuracy of our Penn GPA predictions. This number seems to imply that our Penn GPA predictions are quite accurate because Penn GPAs can range from 0 to 4. However, the actual range of Penn GPAs observed in the dataset is between 3.3 and 4.0. In a variable whose range spans 0.7

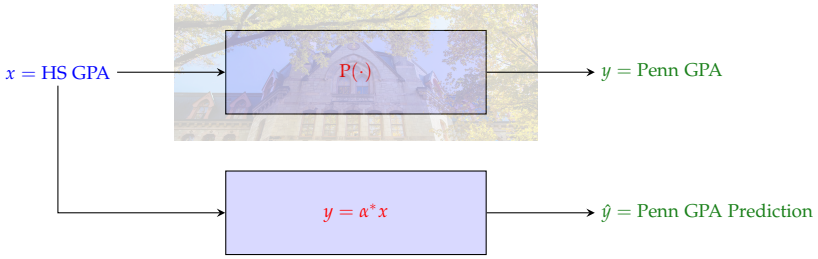

Figure 7. Penn grade point average prediction. We design an artificial intelligence that predicts Penn GPAs based on high school GPA and SAT scores. The AI mimics the same relationship observed in past students.

units, a prediction error of 0.094 is not very accurate. This is apparent in Figure 6 where the line of predicted Penn GPAs is a rough estimate of observed Penn GPAs.

4.3 Linear Regression

We consider now a more complete system in which the Penn GPA is deemed to depend on the high school GPA and the SAT score. We therefore define the input vector $x = [x_1; x_2] = [\text{HS GPA}; \text{SAT}]$ which stacks the High School GPA and the SAT score of a particular student. The system of interest is therefore of the form,

$$\text{Penn GPA} \;=\; y \;=\; P(x) \;=\; P\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \;=\; P\begin{bmatrix} \text{HS GPA} \\ \text{SAT} \end{bmatrix}. \tag{10}$$

To make Penn GPA predictions we postulate a linear model. That is, we define a vector of coefficients $w = [w_1; w_2]$ and postulate that Penn GPAs adhere to the relationship

$$\hat{y} \;=\; w_1 x_1 + w_2 x_2 \;:=\; w^T x. \tag{11}$$

In the second equality in (11) we use the *definition* of the inner product between the coefficient vector w^T and the data vector x.

As is the case of (5), $\hat{y}$ in (11) is a *prediction* of the *true* Penn GPA y that will be available after the fact. The coefficient $w = [w_1; w_2]$ is to be determined with the goal of making predictions $\hat{y}$ close to to actual Penn GPA y. We

do that by considering the MSE associated with the students available in our dataset. For each of these students the vector $x_i = [x_{i1}; x_{i2}]$ stacks the corresponding High School GPA x_{i1} and SAT score x_{i2}. We can therefore write the MSE associated with coefficient w as

$$\text{MSE}(w) \;=\; \frac{1}{N}\sum_{i=1}^{N}\Big(y_i - \hat{y}_i\Big)^2 \;=\; \frac{1}{N}\sum_{i=1}^{N}\Big(y_i - w^T x_i\Big)^2. \tag{12}$$

Task 4 Define and compute the MMSE estimator coefficients w^*, that would extend the MMSE definition in (7). Show that this coefficient is given by the expression

$$w^* = \left[\sum_{i=1}^{N} x_i x_i^T\right]^{-1} \sum_{i=1}^{N} x_i y_i. \tag{13}$$

Compute w^* for the data loaded in Task 1. Compute the RMSE of w^* and comment on the quality of the Penn GPA predictions. ■

5 Requirements

The AI of Section 4 is an indefensible strategy for making admission decisions to Penn. To talk about why this strategy is indefensible we need to talk about requirements. This is just a way of saying what is the goal of the AI system that we are designing. As it happens, we never made this goal explicit. However, implicit in the prediction of Penn GPAs is the fact that we intend to admit students with higher GPA potential. Thus, the specification (the requirement) of the AI system is the following:

(R1) Admit the students that will attain the highest graduation GPA.

Making requirements explicit is important. If the AI makes decisions that are incompatible with our principles, it is not the model's fault, the data's fault, or the training process's fault. It is a problem of having misspecified requirements.

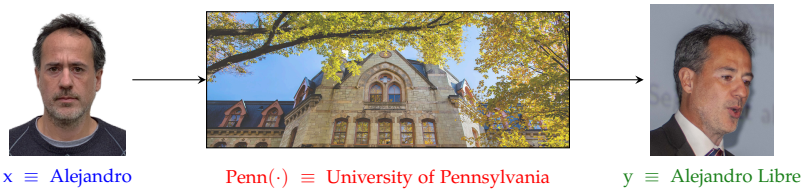

Figure 8. The University of Pennsylvania (Penn). Penn is a system that takes a student as an input and produces a graduate as an output.

5.1 User Requirements

Requirement (R1) is a choice we made as engineers. This is not the same as the user requirement. The specification that people actually in charge of admission decisions would give.

This requirement is actually well known. Penn is a liberal arts institution. As such, our goal is to make students free. This is the literal meaning of liberal arts; the skill (art) of being free (liberal). Penn being Penn, the meaning of free is more concrete and was given to us by Benjamin Franklin[1]:

> "The Idea of what is *true Merit*, should also be often presented to Youth ..., as consisting in an *Inclination* join'd with an *Ability* to serve Mankind, one's Country, Friends and Family; which *Ability* is (with the Blessing of God) to be acquir'd or greatly encreas'd by *true Learning*." [Emphasis mine]

Learn, so that you can develop the *inclination* and the *ability* to serve.

I did not attend Penn, but had I attended Penn, I expect the outcome would had been something like what is depicted in Figure 8. The student Alejandro would be transformed into the graduate Alejandro. The latter is more *libre*. More inclined and more able to serve mankind, his countries, his friends and his family. He would be happier for that.

Having this in mind, the following is a sensible requirement for the admission system:

[1]Benjamin Franklin, "Proposals Relating to the Education of Youth in Pennsylvania." October 1749.

> **(R2)** Admit the students that we can make the most free. Those that after attending Penn will be the most inclined and the most able to serve mankind and their countries, friends and families.

If you ever wondered why university admissions are so fraught, this is the reason. Penn does try to live up to its charter. We do believe in education as a service to our communities and we want to admit and teach students that can have the most positive impact in their communities.

This parenthetical comment aside, what is relevant here is the contrast between (R2) and (R1). Two remarks are warranted in this regard: (i) There is a lot of distance between the *user* requirement (R2) and the *engineering* requirement (R1). (ii) Requirement (R1) is partly motivated by the data that we have available. Indeed, Requirement (R1) reduces the beautiful complexity of an institution of higher education to a map between performance indicators. These performance indicators are reducing the beautiful complexity of students and graduates to numbers and genders, both of which don't say much about their inclination and ability to serve mankind and their countries, friends and families. This coarse simplification of Penn is necessary, however, because the data that we have is limited.

Throughout, we will talk a lot about data, models and decisions but we won't talk much about requirements. This is because the focus of this course is on designing systems that satisfy given requirements. In actual engineering practice, requirements are flexible and it is important to think about how design considerations affect system requirements.

6 Report

Do not take much time to prepare a lab report. We do not want you to report your code and we don't want you to report your work. Just give us answers to questions we ask. Specifically give us the following:

Question	**Report deliverable**
Task 1	Plot of Penn GPA vs HS GPA
Task 1	Plot of Penn GPA vs SAT
Task 2	Derivation of (8)
Task 2	Value of α^*
Task 3	RMSE
Task 3	Comment on the quality of the Penn GPA predictions
Task 4	Derivation of (13)
Task 4	Value of w^*
Task 4	RMS
Task 4	Comment on the quality of the Penn GPA predictions

We will check that your answers are correct. If they are not, we will get back to you and ask you to correct them. As long as you submit responses, you get an A for the assignment. It counts for 4 points of your lab grade.

A Data Tensors

In Section 4.1 the variables x_i and y_i are numbers (scalars) that denote High School and Penn GPAs of individual students. In Section 4.3 we redefine $x_i = [x_1; x_{i2}]$ as a vector that stacks High School GPA and SAT scores of Student i. The data of all N students can then be arranged into either vectors (Section A.1) or matrices (Section A.2). In subsequent labs we will arrange data into multidimensional prisms. We call these arrangements data tensors.

A.1 Data Vectors

We arrange High School and Penn GPA data of all students into vectors. We do that by stacking GPAs of different students on top of each other. Thus, we define vectors of High School and Penn GPAs as stacks of the form,

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}. \tag{14}$$

In this equation we use uppercase in the definitions of the data vectors X and Y to signify that they contain all of the entries in the dataset. The i-th row of these vectors $X_i = x_i$ and $Y_i = y_i$ contain the High School and Penn GPA of Student i.

A useful operation to perform on vectors is a transposition. This is just a rearrangement of the data so that instead of having different students stacked on top of each other, we have data for different students appearing right next to each other. Formally, the transposes X^T and Y^T of X and Y are defined as

$$X^T = \begin{bmatrix} x_1, x_2, \ldots, x_N \end{bmatrix}, \qquad Y^T = \begin{bmatrix} y_1, y_2, \ldots, y_N \end{bmatrix}. \tag{15}$$

In these vectors, the i-th column contains the information of Student i. This is because different columns ox X and Y have become different rows of X^T and Y^T. To differentiate these two different ways of arranging information, we say that X and Y are column vectors and that X^T and Y^T are row vectors.

This notation is convenient because we can use vector operations to write the expressions in (8) in a more compact manner. Indeed, the *definition* of the inner product $X^T Y$ between vectors X^T and Y is

$$X^T Y \;=\; \left[\; x_1, x_2, \ldots, x_N \;\right] \times \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \;:=\; \sum_{i=1}^{N} X_i Y_i \;=\; \sum_{i=1}^{N} x_i y_i. \qquad (16)$$

It follows from the same definition that the inner product of the data vector X with itself is

$$X^T X \;=\; \left[\; x_1, x_2, \ldots, x_N \;\right] \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \;=\; \sum_{i=1}^{N} X_i^2 \;=\; \sum_{i=1}^{N} x_i^2. \qquad (17)$$

This is sometimes called the energy of vector X.

Comparing the expression of the MMSE estimator in (8) with the definitions of inner product and energy in (16) and (17) we conclude that the MMSE estimator can be equivalently written as

$$\alpha^* \;=\; X^T Y \;\Big/\; X^T X. \qquad (18)$$

Observe that (8) may look like an interesting derivation but it is in fact just a consequence of proper definitions. The inner produs is *defined* the way it is defined in (16) to be able to rewrite (8) as in (18)

A.2 Data Matrices

In the same way in which we arranged data intro matrices in Section A.1, we can rearrange High School GPA and SAT scores into a data matrix. This is defined as a stack of the form,

$$X \;=\; \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \;=\; \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix}. \qquad (19)$$

In this matrix each row represents a different student and each column represents a different data type. Column 1 contains High School GPA data. Column 2 contains SAT scores.

As we do with vectors we can transpose matrices. This is just a rearrangement in which rows become columns and columns become rows. We therefore have that the transpose of X is

$$X^T = \begin{bmatrix} x_{11} & x_{21} & \dots & x_{N1} \\ x_{12} & x_{22} & \dots & x_{N2} \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \dots & x_N \end{bmatrix}. \tag{20}$$

This notation is convenient because we use matrix operations to rewrite the linear MMSE estimator in (13) in a more compact notation. To do that we recall (or learn) the *definition* of the product between the data matrix X^T and the data vector Y. This is given by,

$$X^T Y = \begin{bmatrix} x_1 & x_2 & \dots & x_N \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} := \sum_{i=1}^{N} x_i y_i \tag{21}$$

Observe that this equation looks the same as equation (16) but they are actually slightly different. In (21) the symbol x_i represents the vector $x_i = [x_{i1}, x_{i2}]$ whereas in (16) the symbol x_i is just a (scalar) number. The result of the operation in (16) is a number. The result of the operation in (21) is a vector.

Likewise, using the same *definition* of matrix product we can see that the product between the data matrices X^T and X is given by

$$X^T X = \begin{bmatrix} x_1 & x_2 & \dots & x_N \end{bmatrix} \times \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix} := \sum_{i=1}^{N} x_i x_i^T \tag{22}$$

Again, this equation looks similar to (17) but it is different. In (17) we have products $x_i x_i = x_i^2$ between scalars. In (22) we have products between vectors x_i and x_i^T. The results of this operation is a matrix with four entries,

$$x_i x_i^T = \begin{bmatrix} x_{i1} & x_{i2} \end{bmatrix} \times \begin{bmatrix} x_{i1} & x_{i2} \end{bmatrix} := \begin{bmatrix} x_{i1}x_{i1} & x_{i1}x_{i2} \\ x_{i2}x_{i1} & x_{i2}x_{i1} \end{bmatrix} \tag{23}$$

Comparing the expression of the MMSE estimator in (18) with the data matrix products in (21) and (22) we see that the MMSE estimator can be equivalently written as

$$w^* = \Big[X^T X\Big]^{-1} X^T Y. \tag{24}$$

This may look like an interesting derivation but it is in fact just a consequence of proper definitions. The matrix products in (21) and (22) are defined the way they are defined to be able to rewrite (18) as in (24).

B Vectors and Matrices

Almost all of the models that are used in the current practice of AI are either linear or minor variations of linear models. This is the reason why the definition of the matrix product in (21) is important to us. We therefore summarize it here.

A vector x is an arrangement of n numbers stacked on top of each other and the transpose of this vector is the same arrangement with the numbers arranged side by side,

$$x \;=\; \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad x^T \;=\; \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}. \tag{25}$$

When we want to emphasize the difference between x and x^T we say that x is a column vector and that x^T is a row vector.

A matrix A is an arrangement of numbers with m rows and n columns,

$$A \;=\; \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}. \tag{26}$$

Observe that we can always think of a matrix as an arrangement of n column vectors side by side or as a stack of row vectors. Indeed convening

that $a_{:j}$ is a column vector stacking all of the rows of Column j and that $a_{i:}^T$ is a row vector with all of the columns of Row i written side by side we can rewrite (26) as,

$$A = \begin{bmatrix} a_{:1} & a_{:2} & \cdots & a_{:n} \end{bmatrix} = \begin{bmatrix} a_{1:}^T \\ a_{2:}^T \\ \vdots \\ a_{m:}^T \end{bmatrix}. \tag{27}$$

Writing matrices as collections of vectors is fundamental. Professionals think more often about (27) than they do about (26). In particular, this is true when defining products as we show next.

B.1 Vector and Matrix Products

Given two column vectors x and y their inner product is a *scalar* number given by

$$x^T y = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} := \sum_{i=1}^{n} x_i y_i. \tag{28}$$

In this definition it is required that the vectors x and y be of the same dimension. They both need to have the same number of entries. The inner product is a measure of similarity between vectors x and y. When its value is large and positive it indicates that the vectors are similar, i.e., that $x \approx y$. When its value is large and negative it indicates that the vectors are anti-similar, i.e., that $x \approx -y$. When its value is close to zero it means that the vectors are not much related to each other.

To define the product of a matrix A with a vector x we write the matrix A as a stack of row vectors $a_{i:}^T$. We then define the product between A and x as the corresponding stack of inner products between $a_{i:}^T$ and x,

$$y = Ax = \begin{bmatrix} a_{1:}^T \\ a_{2:}^T \\ \vdots \\ a_{m:}^T \end{bmatrix} \times x := \begin{bmatrix} a_{1:}^T x \\ a_{2:}^T x \\ \vdots \\ a_{m:}^T x \end{bmatrix}. \tag{29}$$

In this definition the row vectors $a_{i:}^T$ and the column vector x must have the same number of entries. This implies that the number of columns of the matrix A and the number of rows of the vector x are the same. The output of the product is a vector in which the number of entries equals the number of rows of the matrix A. In (29) the vector x has n entries and the matrix A has m rows and n columns. The product $y = Ax$ is vector with m entries.

From the definitions in (28) and (29) we see that the ith entry of y is

$$y_i \;=\; a_{i:}^T x \;=\; \sum_{j=1}^{n} a_{ij} x_j. \tag{30}$$

This expression is an equivalent definition of the product $y = Ax$. I.e., the product $y = Ax$ is a vector whose entries are given by (30).

To define the product of a two matrices A and B we write the matrix A as a stack of row vectors $a_{i:}^T$ and the matrix B as a concatenation of column vectors $b_{:,j}$. We then define the product between A and B as the corresponding arrangement of inner products between $a_{i:}^T$ and $b_{:,j}$,

$$\begin{aligned} C &= A \times B \\ &= \begin{bmatrix} a_{1:}^T \\ a_{2:}^T \\ \vdots \\ a_{m:}^T \end{bmatrix} \times \begin{bmatrix} b_{:1} & b_{:2} & \cdots & b_{:n} \end{bmatrix} := \begin{bmatrix} a_{1:}^T b_{:1} & a_{1:}^T b_{:2} & \dots & a_{1:}^T b_{:n} \\ a_{2:}^T b_{:1} & a_{2:}^T b_{:2} & \dots & a_{2:}^T b_{:n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m:}^T b_{:1} & a_{m:}^T b_{:2} & \dots & a_{m:}^T b_{:n} \end{bmatrix}. \end{aligned} \tag{31}$$

In this definition each of the rows of the matrix C contains inner products with the same row vector of A and each of the columns contains inner products with the same column vector of B.

From the definitions in (28) and (31) we see that the ijth entry of C is

$$c_{ij} \;=\; a_{i:}^T b_{:,j} \;=\; \sum_{k=1}^{n} a_{ik} b_{kj}. \tag{32}$$

This expression is an equivalent definition of the product $C = AB$. I.e., the product $C = AB$ is a vector whose entries are given by (32).

Either of the equivalent definitions in (31) and (32) assume that the number of columns of the matrix A and the number of rows of the matrix B

are the same. The output of the product is a matrix in which the number of rows equals the number of rows of the matrix A and the number of columns equals the number of columns of B. In (32) the matrix A has m rows and the matrix B has n columns. The matrix C has m rows and n columns. The number of columns of A and the number of rows of B must be the same, say, a number p.

B.2 Tensors

A scalar is a 0-dimensional arrangement of numbers. A vector is a 1-dimensional arrangement of numbers. A matrix is a 2-dimensional arrangement of numbers. We can define arrangements of arbitrary dimension that we call tensors. For instance a three dimensional tensor A is an arrangement of $m \times n \times p$ scalars a_{ijk}.

Slicing a tensor produces a tensor of smaller dimensionality. Of particular note, slicing a tensor of dimension 3 produces a matrix. We use the notation $A(:,:,k)$ to signify a 2-dimensional slice along the third coordinate of the tensor A. This slice is the matrix

$$A(:,:,k) \;=\; \begin{bmatrix} a_{11k} & a_{12k} & \dots & a_{1nk} \\ a_{21k} & a_{22k} & \dots & a_{2nk} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1k} & a_{m2k} & \dots & a_{mnk} \end{bmatrix}. \tag{33}$$

Likewise, we use the notations $A(i,:,:)$ and A(:,j,:) to denote slices along the first and second coordinate, respectively. This is not unlike the definitions in (27). We are just rewriting subindexes as arguments and considering arrangements of larger dimension.

We can also consider slices along more than one coordinate. For instance, slicing the three dimensional tensor along the last two coordinates produces the vector

$$A(:,j,k) \;=\; \begin{bmatrix} a_{1jk} \\ a_{2jk} \\ \vdots \\ a_{mjk} \end{bmatrix}. \tag{34}$$

Tensors are convenient ways of arranging data and parameters in AI models. We will use sliced tensors often in our computations. When doing

so we need to pay attention to how slicing produces matrices and vectors representing the right operation. This may get cumbersome but it is not difficult.