ECE-GY 6143, Spring 2020

Lecture 1: Vector representations of data

In several (most?) applications, "data" usually refers to a list of numerical attributes associated with an object of interest.

For example: consider meteorological data collected by a network of weather sensors. Suppose each sensor measures:

- wind speed (w) in miles per hour
- temperature (t) in degrees Fahrenheit

Consider a set of such readings ordered as tuples (w, t); for example: (4,27), (10,32), (11,47), $\ldots$.

It will be *convenient* to model each tuple as a point in a two-dimensional vector space.

More generally, if data has d attributes (that we will call *features*), then each data point can be viewed as an element in a d-dimensional vector space, say $\mathbb{R}^d$.

Here are some examples of vector space models for data:

1. Sensor readings (such as the weather sensor example as above).
2. Image data. Every image can be modeled as a vector of pixel intensity values. For example, a 1024×768 RGB image can be viewed as a vector of $d = 1024 \times 768 \times 3$ dimensions.
3. Time-series data. For example, if we measure the price of a stock over $d = 1000$ days, then the aggregate data can be modeled as a point in 1000-dimensional space.

Properties of vector spaces

Recall the two fundamental properties of vector spaces:

- Linearity: two vectors $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ can be *added* to obtain:

$$x + y = (x_1 + y_1, \ldots, x_d + y_d).$$

- Scaling: a vector $x = (x_1, \ldots, x_d)$ can be scaled by a real number $\alpha \in \mathbb{R}$ to obtain:

$$\alpha x = (\alpha x_1, \ldots, \alpha x_d).$$

Vector space representations of data are surprisingly general and powerful. Moreover, several tools from linear algebra/Cartesian geometry will be very useful to us:

1. Norms. Each vector can be associated with a *norm*, loosely interpreted as the "length" of a vector. For example, the ℓ_2, or *Euclidean*, norm of $x = (x_1, \ldots, x_d)$ is given by $\|x\|_2 = \sqrt{\sum_{i=1}^d x_i^2}$. The ℓ_1, or *Manhattan*, norm of x is given by $\|x\|_1 = \sum_{i=1}^d |x|_i$.
2. Distances. Vector spaces can be endowed with a notion of distance as follows: the "distance" between x and y can be interpreted as the norm of the vector $x - y$. For example, the ℓ_2, or

Euclidean, distance between x and y is given by:

$$\|x - y\|_2 = \sqrt{\sum_{i=1}^{d} x_i^2}.$$

One can similarly define the ℓ_1-distance, etc. The choice of distance will be crucial in several applications when we wish to compare how close two vectors are.

3. Similarities. (These are, in some sense, the opposite of distance.) Define the Euclidean *inner product* between vectors x and y as:

   $$\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i.$$

   Then, the *cosine* similarity is given by:

   $$\text{sim}(x, y) = \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2}.$$

   The *inverse cosine* of this quantity is the generalized notion of *angle* between x and y.

Application: Document search

Let us instantiate these ideas for a concrete warmup application in text processing. For the purpose of this discussion, let a *document* refer to any collection of words in the English language. This could be a (literal) document, or a webpage, or a book, or just a search phrase, etc. We consider the following

> Problem: Given a dataset of n documents $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$ and a query document D^*, find the document in the database that best matches D^*.

For example, if you were to build a simple querying engine over a database of webpages, $\mathcal{D}$ is your webpage database, and any set of input keywords can be your query "document".

The first (and most crucial) course of action is to decide a suitable *representation* of your data. Once we do that, then we can think of applying some combination of the aforementioned linear algebraic tools.

Stage 1: Modeling the data

Can we somehow naturally model "documents" in English as "vectors" in some space?

There is no unique way to do this, but here is a first attempt. Let d denote the number of all words in the English language. (This is a very large number, and depending on how you define "English word", this can range from 10,000 to more than half a million.) Index all English words from 1 through d.

Convert every document into a d-dimensional vector x, by simply letting the j^{th} co-ordinate of x to be the **number** of occurrences of word j in the document. (This number is also sometimes called the "term-frequency".)

This gives us a set of n vectors $\{x_1, \ldots, x_n\}$, where each x_i is an element of $\mathbb{R}^d$, and represents the i^{th} document.

This has the deficiency of throwing out *context* in a document. For example, consider the following two (short) "documents":

> The quick brown fox jumped over the lazy dog.
>
> The lazy brown dog jumped over the quick fox.

will both have the **exact** same vector representation, even though they are clearly different documents. But let's assume for now that we won't face this ambiguity issue in actual applications. In the last couple of lectures we will discuss advanced ML methods which *can* distinguish between such cases but we will skip that for now.

Do the same to the query document as well, so that we also obtain a query vector x^*.

Stage 2: Translating into a linear algebra problem

Once we have a vector representation of every document, as well as the query vector x^*, we now have to define what "best match" means in the context of vector spaces.

Again, there is no unique way to do this. In text analysis, a common measure of "best" is the cosine similarity. More precisely, we calculate – for each i – the cosine similarity between the query vector with each other vector in the dataset:

$$\cos\theta_i = \frac{\langle x^*, x_i\rangle}{\|x^*\|_2\|x_i\|_2},$$

and output the index i^* as __ :

$$i^* = \arg\max_i \cos\theta_i.$$

The "max" in the above problem can be found by simply scanning through the dataset, calculating cosine similarities, and picking the document with the largest one. In big-oh notation, this takes $O(nd)$ FLOPS (Is this good or bad? Why?). We will discover better/faster approaches later on in the course.

To summarize, once a good representation of the data is found, the rest of the problem is entirely computational: find a metric that captures the meaning of "best" (specified in terms of a similarity measure or a *loss* function), and find a way to optimize that metric.

An improved approach

The above method works well, but the vector space representation is a bit brittle. Specifically, the definition on term frequency means that certain commonly occurring words ("the", "be", "to", "of", "and", etc) will constitute the dominant coordinate values of each document.

This heavily inflates the influence of the co-ordinates of the vector space corresponding to these words. However, generally these words are uninformative and their effect should be ideally minimized.

In order to do this coherently, we define a new vector space representation. For document i, we construct a vector x_i such that the j^{th} co-ordinate is given by:

$$x_i(j) = \text{tf}_i(j) \cdot \text{idf}(j).$$

The term $\text{tf}_i(j)$ is the same as before; it represents *term-frequency* and counts the number of occurrences of word j in document i.

The term idf(j) is called the *inverse-document* frequency, and is defined as follows. Let n_j be the number of documents in the database which contain at least one occurrence of word j. Then,

$$\text{idf}(j) = \log \left(\frac{n_j}{n} \right)^{-1} = \log \frac{n}{n_j}.$$

Therefore, if a word occurred in every document, its idf value would be zero, and the word would not contribute to the similarity calculations.

Note that the idf value is the same for all documents, and is only a property of the dataset under consideration.

Concluding note

Much of the rest of this course will involve solving problems similar to (but more challenging than) the one above.

A lot of our theory emphasis in the beginning will primarily be on Stage 2: given the "right" vector space representation of a particular dataset, what mathematical tools and techniques are available to us for further analysis?

However, Stage 1 is where most of the power of machine learning arises. Indeed, choosing the "right" representation (or features) of the data is indeed what can be viewed as the goal of most modern machine learning techniques. Until rather recently, a significant amount of *human* intuition was required to hand-select "good" features based on domain knowledge; the field of *deep learning* has changed this way of thinking quite a bit. Much to come in later lectures.