Explanatory Notes for 6.390

Shaunticlair Ruiz (Current TA)

Fall 2022

Thermometer Code

Next, we'll relax how number-like our feature is. This time, we don't need our data to behave like a number, but it does have an **ordering**.

By "relax", we mean we'll remove some requirements for our feature, like being able to add them together.

Some examples:

- Results of an opinion poll:
  - "Strongly Agree", "Agree", "Neutral", "Disagree", "Strongly Disagree"
- Education level:
  - "Below High School", "High School Degree", "Associates Degree", "Bachelors" "Advanced Degree"
- Ranking of athletes

In this case, we can't just use numbers $\{1, 2, 3, ...\}$. Why not?

Because that implies that there's a specific "scaling" between points: Is the #1 athlete twice as good as the #2 athlete? Maybe, but that's not what the ranking tells us!

> **Concept 1**
> Data that is **ordered** but not **numerical** cannot be represented with **a single real number**.
>
> Otherwise, we might consider one element to be a certain amount "larger" or "smaller" than another, when that's not what **ordering** means.

Example: Suppose we assign $\{1, 2, 3\}$ for $\{\mathtt{Disagree}, \mathtt{Neutral}, \mathtt{Agree}\}$. The person who writes 'agree' is doesn't "agree three times as much" as the person who writes 'disagree'!

But, we still want to keep that ordering: counting up from one element to the next. How do create an order without creating an exact, numeric difference?

Just now, we tried to count by increasing a single variable. But, there's another way to count: counting up using multiple different variables!

This approach is more similar to counting on your fingers.

$$\text{Class 1} \longrightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad \text{Class 2} \longrightarrow \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \qquad \text{Class 3} \longrightarrow \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad \text{Class 4} \longrightarrow \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

This version allows us to avoid the problems we had before: this doesn't behave the same way as a **numerical** value.

To better understand what's going on, here's another way to frame it:

Class(x) is just shorthand for, "which class is x in?"

$$\phi(x) = \begin{bmatrix} \text{Class}(x) \geqslant 4 \\ \text{Class}(x) \geqslant 3 \\ \text{Class}(x) \geqslant 2 \\ \text{Class}(x) \geqslant 1 \end{bmatrix} \tag{1}$$

Example: Suppose x is in class 3. The bottom three statements are all true, the top one is false: so we get $[0, 1, 1, 1]^T$.

This helps us understand why this encoding is so useful:

- We aren't directly "adding" variables to each other: they stay separated by **index**.
- When using a linear model $\theta^T \phi(x)$, each class matches a different θ_i.
- Despite not behaving like numbers, "higher" classes in the order still keep track of all of the classes below them.
  - **Example:** Class 2-4 all share the feature $\text{Class}(x) \geqslant 2$ (equivalent to $\text{Class}(x) > 1$).

θ_i scales the i^{th} variable. So, each class can be scaled differently!

This technique is called **thermometer encoding**.

Definition 2

Thermometer encoding is a **feature transform** where we take each class and turn it into a feature vector $\phi(x)$ where

$$\text{Class 1} \longrightarrow \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \text{Class 2} \longrightarrow \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \text{Class 3} \longrightarrow \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{Class } k \longrightarrow \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- The **length of the vector** is the **number of classes** k we have.
- The i^{th} class has i **ones**.
- This transformation is only appropriate if the data
  - Is **ordered**,
  - But not **real number-compatible**: we can't add the values, or compare the "amount" of each feature.

Example: We reuse our example from earlier:

$$\phi(x_{\text{Class 1}}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \phi(x_{\text{Class 2}}) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \qquad \phi(x_{\text{Class 3}}) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad \phi(x_{\text{Class 4}}) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

One-hot Code

We introduced this technique in the **previous** chapter:

When there's no clear way to **simplify** our data, we accept the current discrete classes, and **convert** them to a number-like form.

- Examples:
  - Colors: $\{\text{Red}, \text{Orange}, \text{Yellow}, \text{Green}, \text{Blue}, \text{Purple}\}$
  - Animals: $\{\text{Dog}, \text{Cat}, \text{Bird}, \text{Spider}, \text{Fish}, \text{Scorpion}\}$
  - Companies: $\{\text{Walmart}, \text{Costco}, \text{McDonald's}, \text{Twitter}\}$

We can't use thermometer code, because that suggests a natural **order**. And we definitely can't use real numbers.

Example: $\{\text{Brown}, \text{Pink}, \text{Green}\}$ doesn't necessarily have an obvious order: you could force one, but there's no reason to.

But, we can use one idea from thermometer code: each class in a different variable.

$$\begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_k \end{bmatrix} \tag{2}$$

But in this case, we don't "build up" our vector: we replace $\text{Class}(x) \geqslant 4$ with $\text{Class}(x) = 4$.

$$\phi(x) = \begin{bmatrix} \text{Class}(x) = 4 \\ \text{Class}(x) = 3 \\ \text{Class}(x) = 2 \\ \text{Class}(x) = 1 \end{bmatrix} \tag{3}$$

This approach is called **one-hot encoding**.

Definition 3

One-hot encoding is a way to represent **discrete** information about a data point.

Our k classes are stored in a length-k column **vector**. For **each** variable in the vector,

- The value is **0** if our data point is **not in that class**.
- The value is **1** if our data point is **in that class**.

$$\text{Class } 1 \longrightarrow \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \text{Class } 2 \longrightarrow \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad \text{Class } 3 \longrightarrow \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad \text{Class } k \longrightarrow \begin{bmatrix} 1 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

In one-hot encoding, items are **never** labelled as being in **two** classes at the **same time**.

- This transformation is only appropriate if the data is
  - Does not have another **structure** we can reduce it to: it's neither like a **real number** nor **ordered**
  - We don't have an **alternative** representation that contains more (accurate) information.

Example: Suppose that we want to classify **furniture** as table, bed, couch, or chair.

$$\begin{bmatrix} \text{table} \\ \text{bed} \\ \text{couch} \\ \text{chair} \end{bmatrix} \tag{4}$$

For each class:

$$y_{\texttt{chair}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad y_{\texttt{table}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad y_{\texttt{couch}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad y_{\texttt{bed}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{5}$$

One-hot versus Thermometer

One common question is, "why can't we use one-hot for ordered data? We could sort the indices so they're in order".

However, there's a problem with this logic: the computer **doesn't care** about the order of the variables in an array: it contains no information!

Why is that? If the vector has an order, shouldn't that **affect** the model?

Well, remember that our model is represented by

$$\theta^T x = \sum_i \theta_i x_i \tag{6}$$

The vector format $\theta^T x$ is just a way to **condense** our equation: the ordering goes away when we compute the sum.

Concept 4

Order of elements in a vector **don't** affect the behavior of our model.

This is because a linear model is a **sum**, and sums are the same regardless of **order**.

If our model has the same math regardless of order, then it can't use order information.

Example: We'll take a vector, and rearrange it.

Despite shuffling, these two equations are equivalent!

$$\theta^T \phi(x) = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad \longrightarrow \qquad (\theta^T)^* (\phi(x))^* = \begin{bmatrix} \theta_3 & \theta_1 & \theta_4 & \theta_2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The math is the same, despite changing order: our model knows nothing about ordering.

Clarification 5

One hot encoding cannot encode information about ordering.

Thermometer encoding is required to **represent ordered objects**.

Why is thermometer encoding able to of representing ordering? Let's try shuffling it, too.

$$\theta^T \phi(x) = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{7}$$

$$(\theta^T)^* (\phi(x))^* = \begin{bmatrix} \theta_3 & \theta_1 & \theta_4 & \theta_2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \tag{8}$$

Even though we've changed the order, we still know this is the **third** in the order, because we have **three** 1's!

Concept 6

Even though the **order of elements** in a vector **doesn't matter**, we can retrieve the order of **thermometer coding** based on the **number of 1's in the vector**.