Data Processing

CPSC 383: Explorations in Artificial Intelligence and Machine Learning
Fall 2025

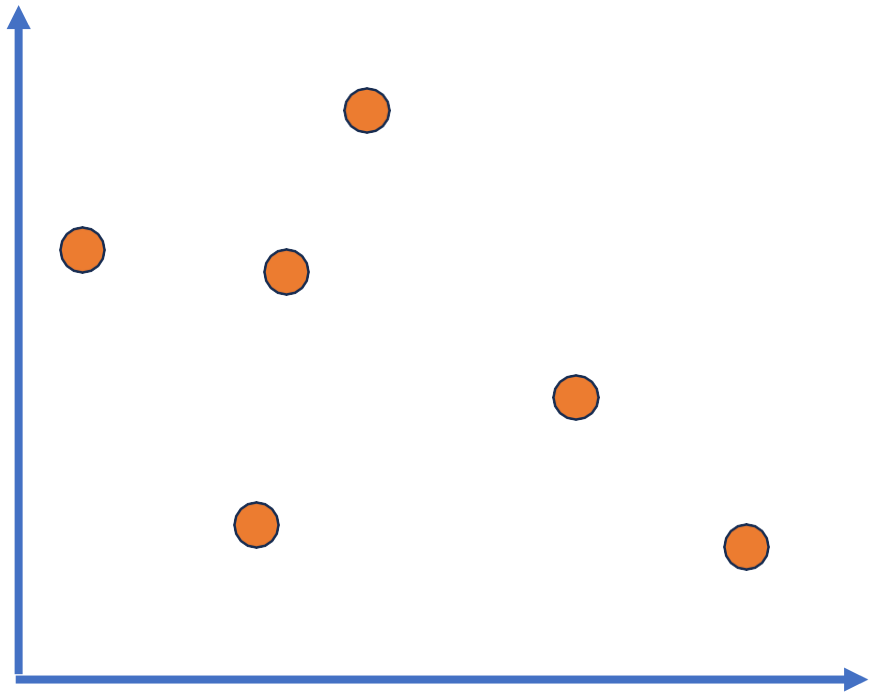
Jonathan Hudson, Ph.D.
Associate Professor (Teaching)
Department of Computer Science
University of Calgary

August 27, 2025

UNIVERSITY OF CALGARY

Vectors (review)

A **vector** in n dimensions is a tuple of n values, $\vec{\boldsymbol{x}} = (x_1, x_2, x_3, \ldots x_n)$.



Data encoding

In machine learning, we like to work with vectors.

To do this, we need to find ways to encode whatever data we have into vectors in n-dimensional space.

- Applies to both values and labels
- We may also need to translate the output back into “natural” format

Data encoding

Why do we like vectors?

- Numeric representation of data
- Abstracts away unnecessary details
- Consistency across different domains

Data encoding

It is important to find a good **representation** of your data that works well for the types of patterns you are looking for.

- Ideally we want “similar” data points to have similar encodings

Feature Vectors

Feature vectors

A single input value x might consist of many **features** representing a particular data point.

Our first task is to convert x into a **feature vector** that can be input to our ML algorithm.

- We will do this by encoding each feature separately and then concatenating

Feature vectors

For our purposes, a feature vector will be a vector of size n.

A dataset with m values can then be represented as a $m \times n$ array.

Discrete features

- For a **discrete** feature that can take on one of k possible values, there are a couple approaches to encoding.

Discrete features

One-hot code is best when there is no extra structure to the values, and they just represent choosing one of k distinct options.

$$\begin{bmatrix}1\\0\\0\\\dots\\0\end{bmatrix} \quad \begin{bmatrix}0\\1\\0\\\dots\\0\end{bmatrix} \quad \begin{bmatrix}0\\0\\1\\\dots\\0\end{bmatrix} \quad \dots \quad \begin{bmatrix}0\\0\\0\\\dots\\1\end{bmatrix}$$

Discrete features

Thermometer code is good for where the ordering conveys some information, but there aren't natural numeric values.

$$\begin{bmatrix}1\\0\\0\\\dots\\0\end{bmatrix} \quad \begin{bmatrix}1\\1\\0\\\dots\\0\end{bmatrix} \quad \begin{bmatrix}1\\1\\1\\\dots\\0\end{bmatrix} \quad \dots \quad \begin{bmatrix}1\\1\\1\\\dots\\1\end{bmatrix}$$

Exercise

- Suppose you were asked to encode a label from the set {cat, dog, horse, frog} in one-hot encoding. What would the encoding of each label look like?

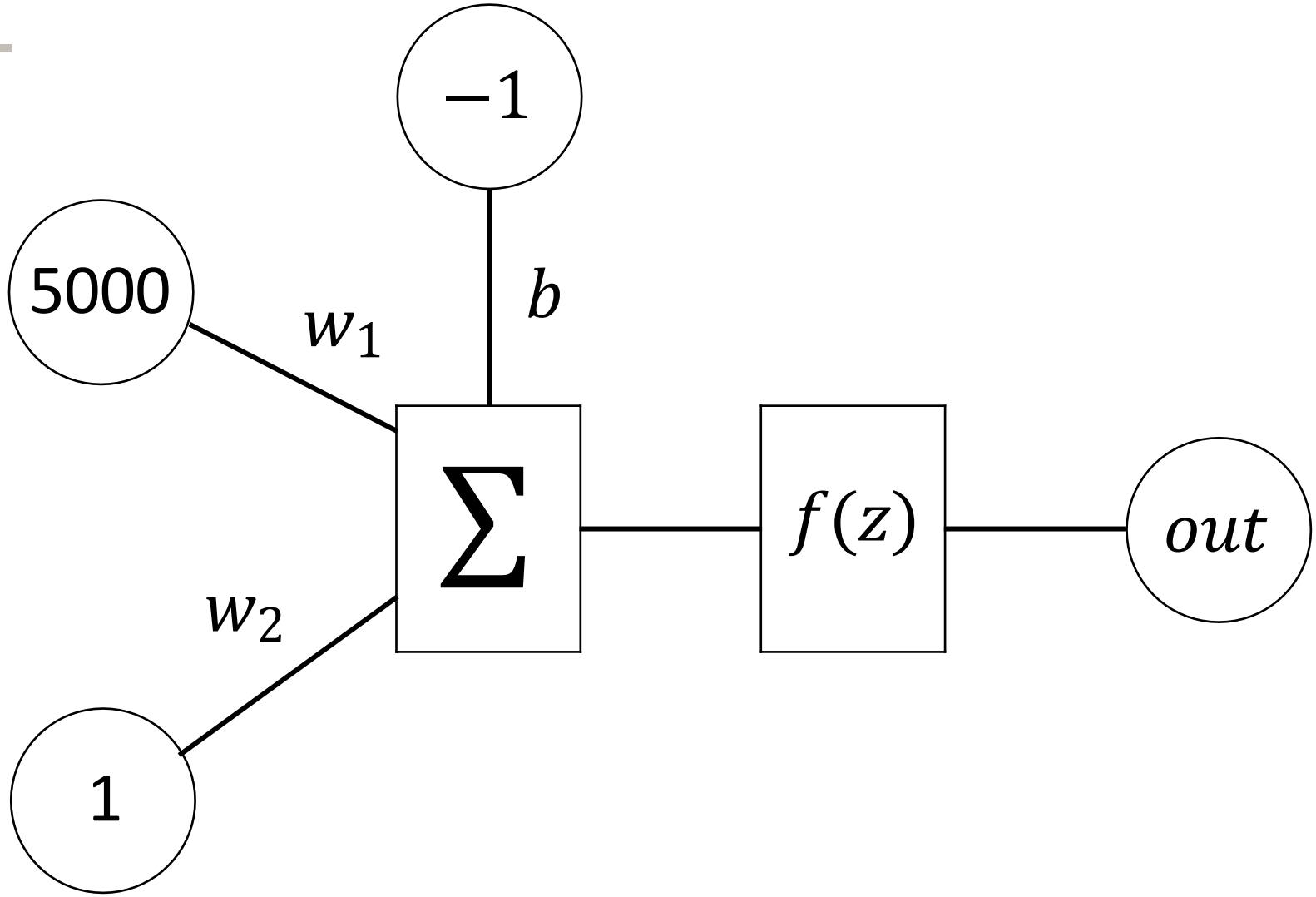
Discrete features

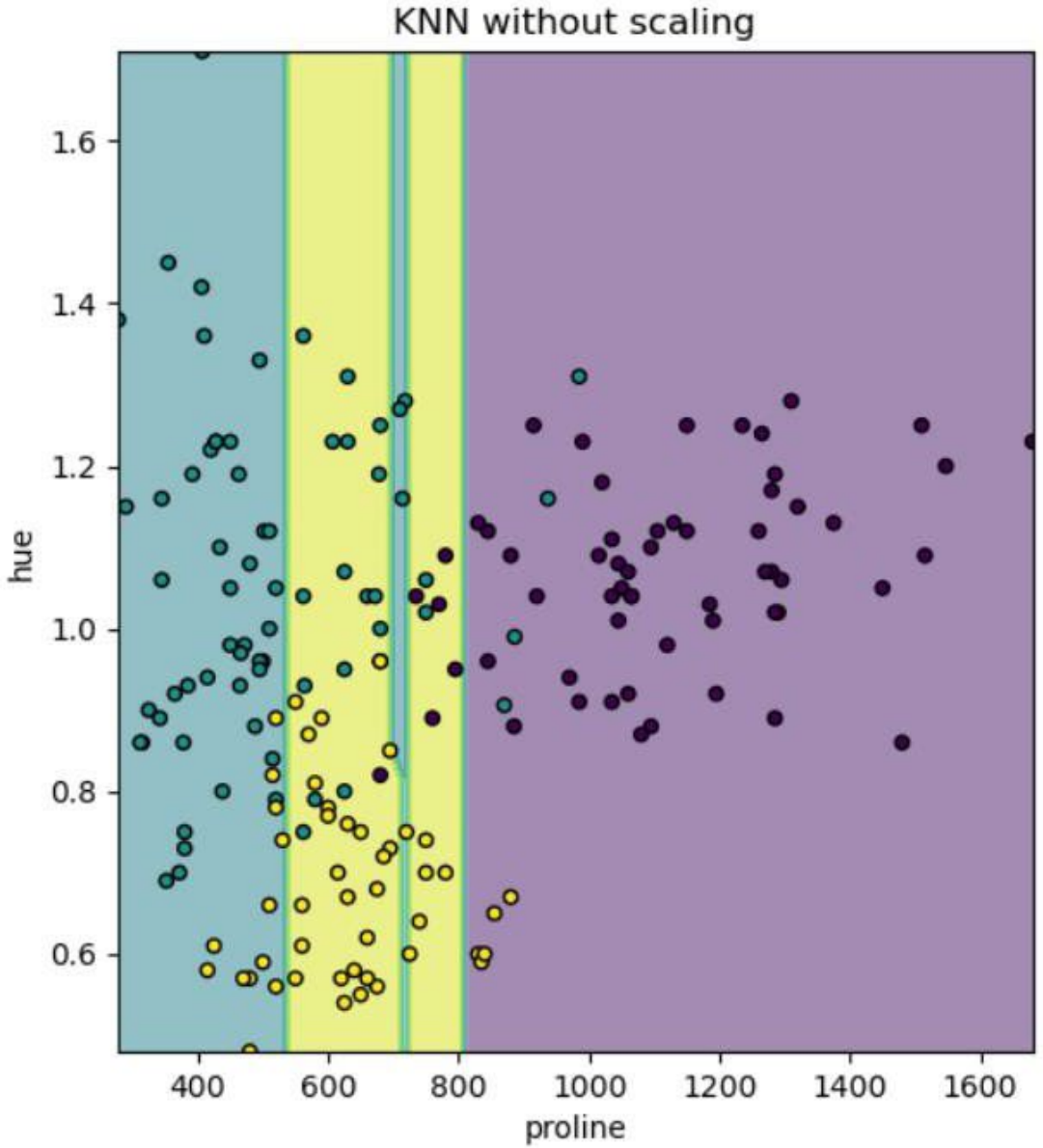
Numeric values [1]…[k] should only be used where the values signify some inherently numeric quantity.

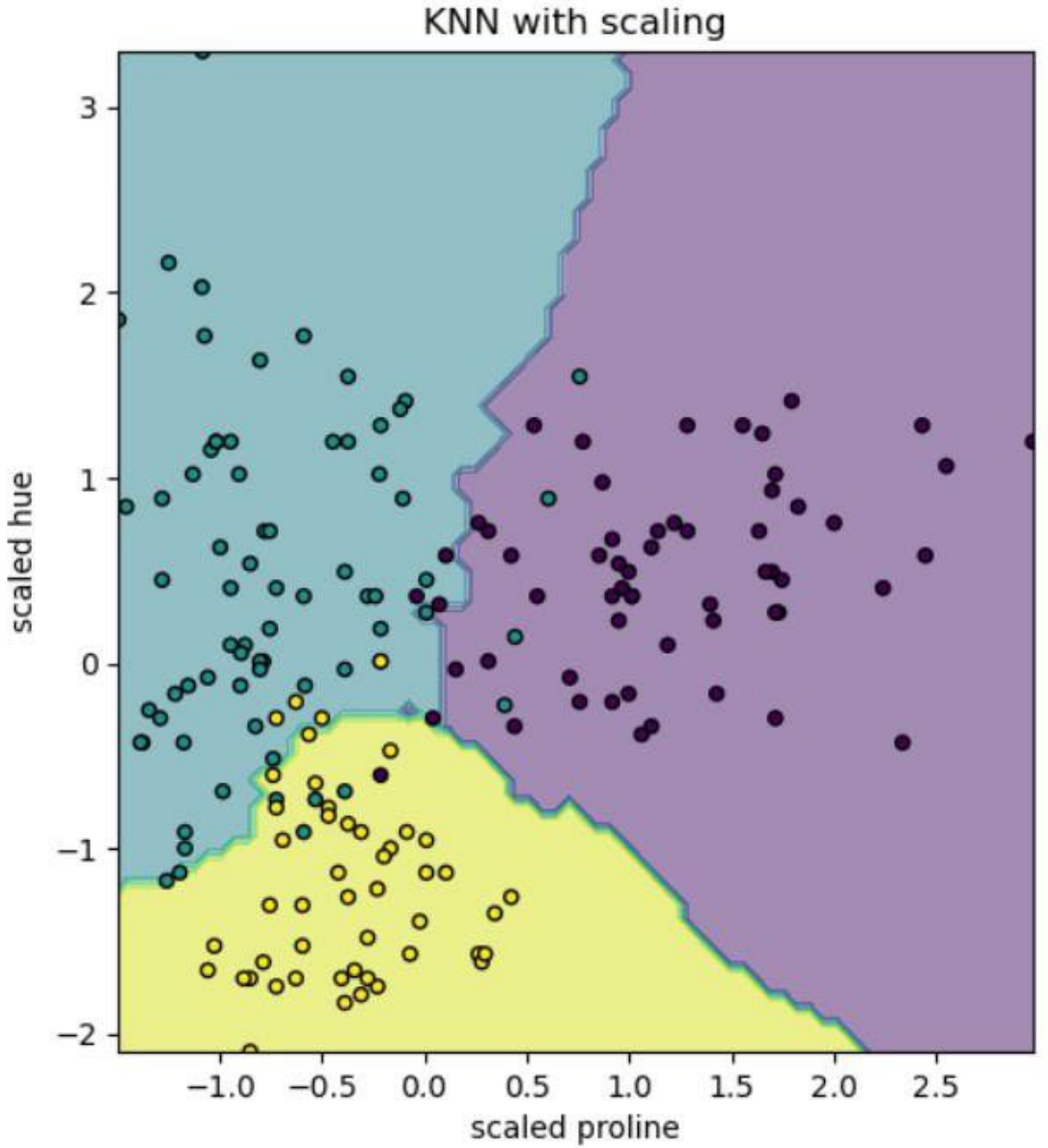
These would likely be subject to further numerical processing.

Normalization

Exercise







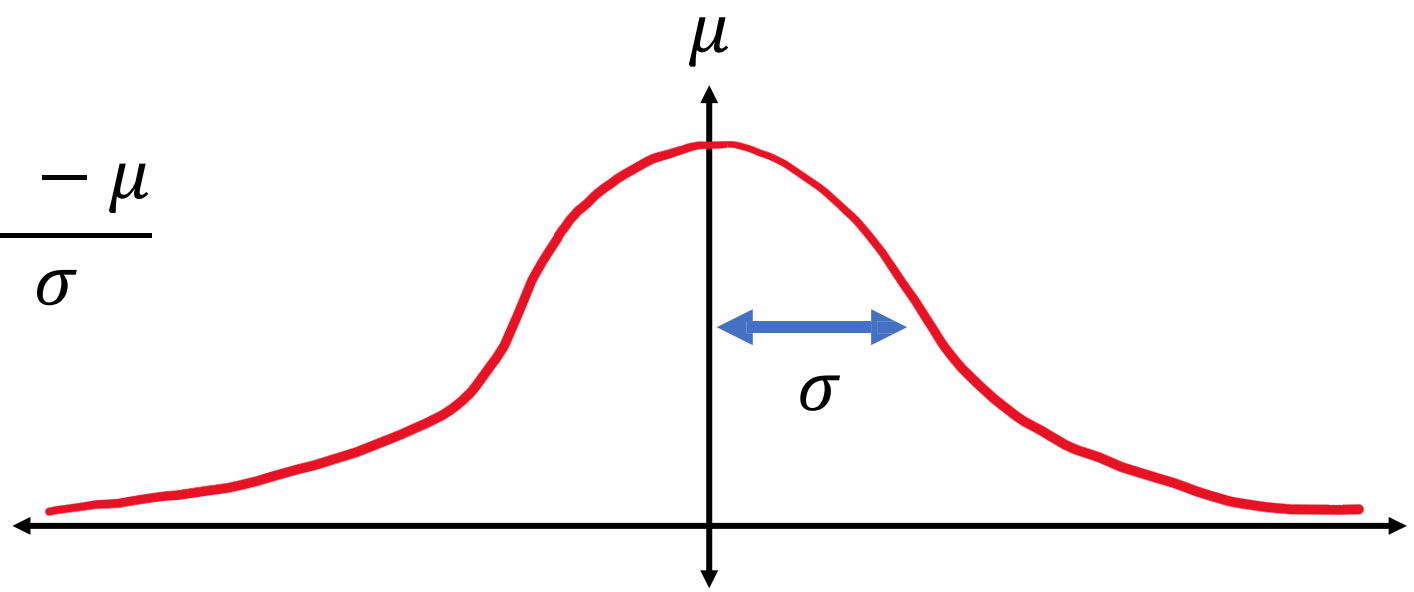
https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html#sphx-glr-auto-examples-preprocessing-plot-scaling-importance-py

Numeric features

For numeric values, best practice is usually to convert the values into their z-scores.

- This is called **standardizing** the values
- μ is the mean (average) and σ is the standard deviation

$$z = \frac{x - \mu}{\sigma}$$



Exercise

Suppose you have the following set of values in your dataset measuring the weight of different loaves of bread:

```
12.3, 11.5, 13.2, 11.8, 12.2
```

What are their values when converted to z-scores (i.e. standardized)?

Values: 12.3, 11.5, 13.2, 11.8, 12.2

Numeric features

If z-scores don’t make sense, it is often still a good idea to **rescale** the values so they lie in (-1, 1) or (0, 1).

- e.g. if values are uniformly distributed

Question

Are Boolean values (0 or 1) discrete or numeric?

Compound Features

Compound features

If a feature can be split into two component features, it is often better to encode each of the smaller features separately.

- Benefits from a space and a training perspective

Exercise

Suppose you are encoding the model and year of a car as a feature vector. There are three possible models and ten possible years.

What is the dimension of your resulting vector if you encode them as a pair? What if you encode them separately?

Multi-dimensional Data

Example

In the MNIST dataset, images are 28 x 28 pixels, and each pixel is a grayscale value from 0-255.

How can we convert this into a vector?

Example

In the MNIST dataset, images are 28 x 28 pixels, and each pixel is a grayscale value from 0-255.

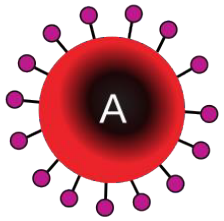
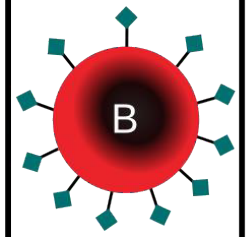
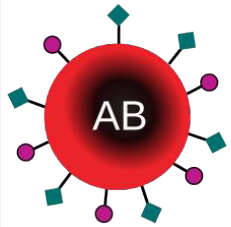
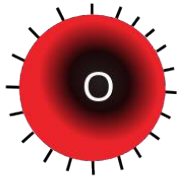
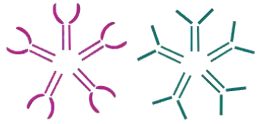
- One way to get a feature vector is to simply **flatten** the image
- Remember to rescale!

$$\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \cdots \\ 0 \\ 0 \end{bmatrix}$$

Example

How could you encode someone's blood type as a feature vector?

	Group A	Group B	Group AB	Group O
Red blood cell type				
Antibodies in plasma	 Anti-B	 Anti-A	None	 Anti-A and Anti-B
Antigens in red blood cell	A antigen	B antigen	 A and B antigens	None

Labelling

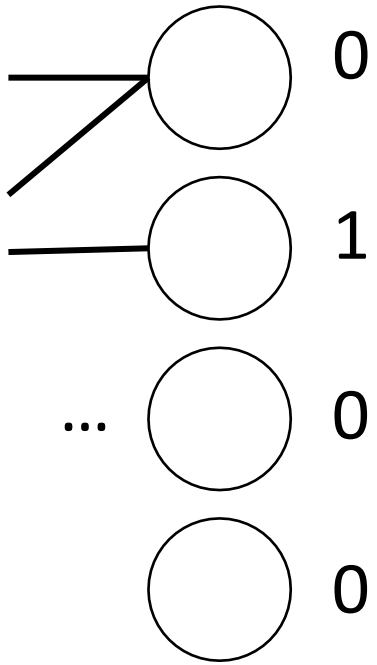
Outputs

So far, our neural nets are restricted to simple single-neuron outputs.

This works well for regression and binary classification, but what if you want to output one of 10 possible labels, for example?

Classification

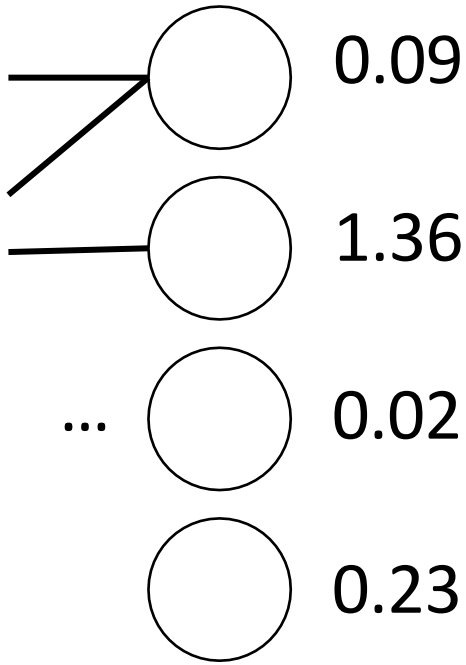
If there are 10 possible labels, you would want 10 neurons in the output layer.



Classification

In reality, the output will be a vector of 10 decimal values called **logits**.

- But these aren't probabilities!



Classification

The raw outputs of the neural net can be interpreted as the relative certainties of the labels.

To convert this to a vector of probabilities, we use the **softmax** function:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Classification

The raw outputs of the neural net can be interpreted as the relative certainties of the labels.

To convert this to a vector of probabilities, we use the **softmax** function:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Outputs from normalizing: [0.05, 0.80, 0.01, 0.14]

Outputs from softmax: [0.15, 0.54, 0.14, 0.17]

Labels

We also need to represent our labels in a way we can work with them!

- Need to be comparable to the output of our model

For classification problems, labels will be from a set of k discrete values. These can be encoded using either a numeric or a one-hot encoding.

For regression problems, the output of the model is usually taken directly as a numeric value.

Labels

Tensorflow will support either one-hot or numeric labels, but you need to indicate which one you are using.

- MNIST labels are stored as numbers 0-9

The keyword “sparse” means you want Tensorflow to convert your labels to one-hot vectors for you.

Loss Functions and Labelling

Loss functions

Tensorflow has different loss functions depending on how you are capturing the output of your neural net **and** how you are encoding your labels:

- Categorical Cross-Entropy
- Sparse Categorical Cross-Entropy
- Binary Cross-Entropy (note: no softmax!)

Loss functions

For numerical stability reasons, we often don't build softmax into our neural net.

Instead, we leave the outputs as raw logits and tell our loss function to take this into account:

```
tf.keras.losses.SparseCategoricalCrossentropy(from_logits = True)
```

Curse of dimensionality

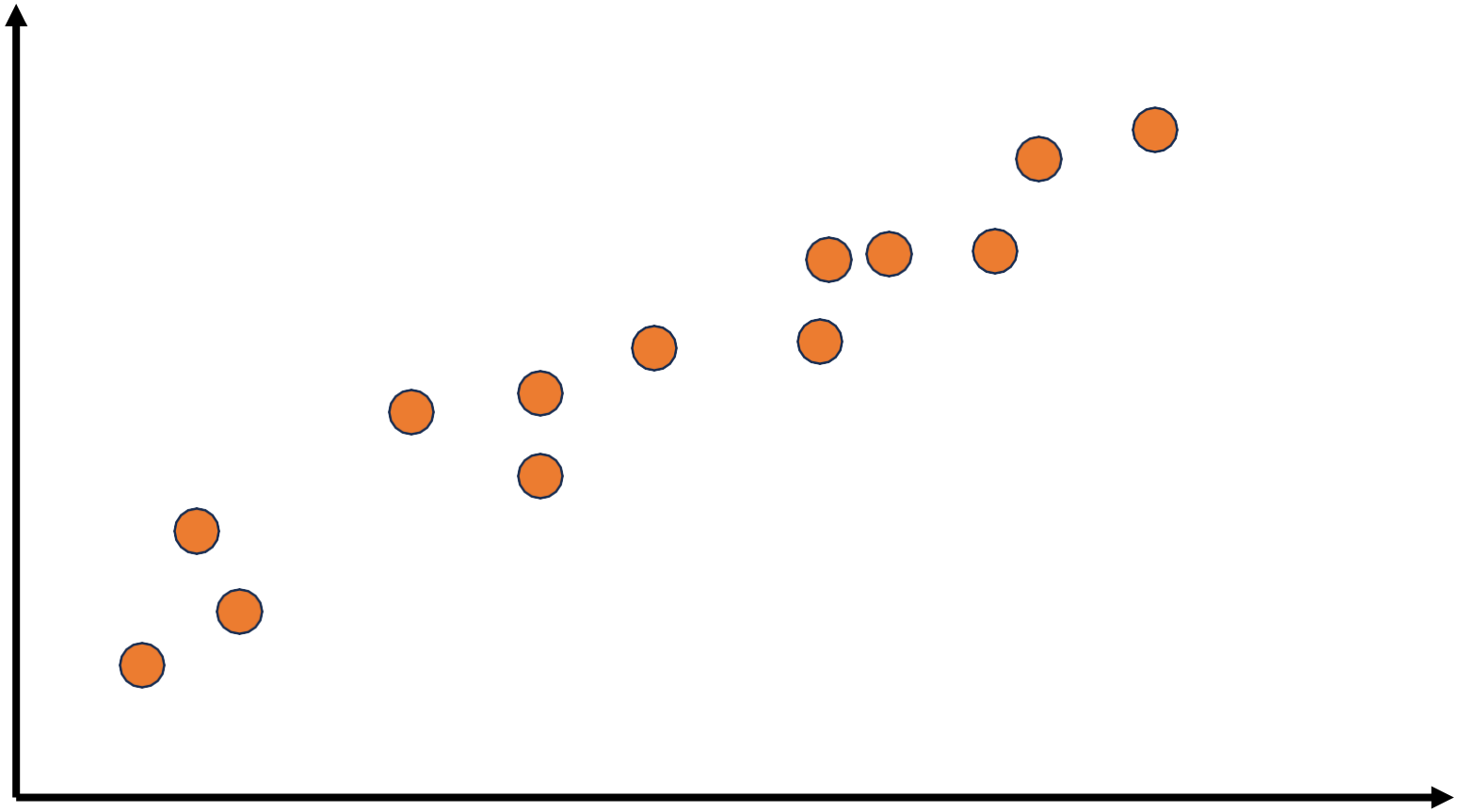
One issue when encoding features as vectors, especially using one-hot, is that the resulting vector space is both very large and very sparse.

- This makes it harder for algorithms to learn meaningful patterns

Dimensionality reduction is one way of combatting this.

- Goal is to reduce the number of dimensions while still retaining representative information

Example (revisited)



Next…stochastic gradient descent

Jonathan Hudson, Ph.D.
jwhudson@ucalgary.ca
https://cspages.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY