1.5em 0pt

Harvard CS 182: Artificial Intelligence November 3, 2022

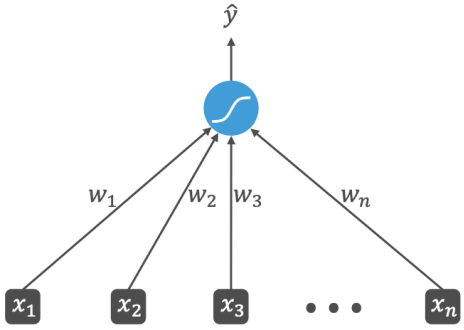
Section 11: Neural Networks and Fairness

Lecturer: Ariel Procaccia *Authors: Sanjana Singh and Catherine Cui*

Neural Networks

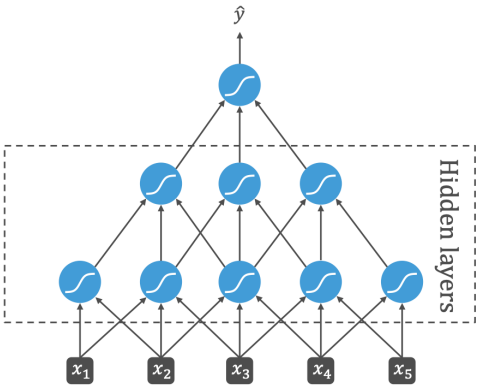
Logistic Regression, Recapped

As a recap, we previously talked about logistic regression. We have inputs x_i and a weight w_i associated to each input. The weighted sum is fed into the sigmoid/logistic function which then outputs $\hat{y}$, a number between 0 and 1, which represents the predicted probability of the label being 1.



What are neural networks?

Neural networks essentially consist of many sigmoid functions (or some other activation functions), and the outputs of these sigmoid functions are also connected by weights and fed into other sigmoid functions. The layers between the input and output layers are called the hidden layers and take care of the computations behind the scenes. Below is a visual representation of a neural network:

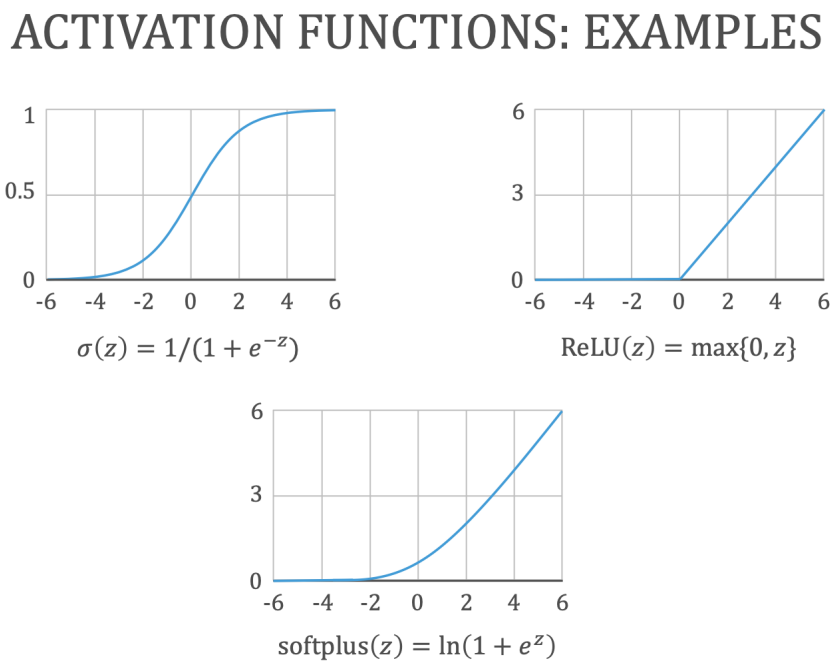


In this lecture, we mainly focus on feed-forward networks which are acyclic. Each node is called a **unit** and each unit calculates the weighted sum of its predecessors and applies an **activation function** to it (in our previous exampe, this activation function was the sigmoid function). In lecture, we discovered that if all activation functions are linear, the entire neural network is just a linear function, which is very restricted. Therefore, some examples of non-linear activation functions include

(i) Sigmoid function

(ii) Rectified linear unit (ReLU): zero until z is zero, then linear in the input; one potential issue with ReLU is it's non-differential at zero

(iii) Softplus: essentially ReLU, but doesn't have the potential issue of being non-differentiable at zero.



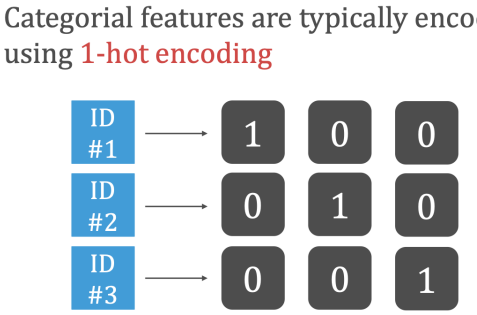
A choice of network architecture is a choice of units, activation functions, and edges. This defines a hypothesis class whose parameters are weights on edges; changing weights results in different functions. In fact, the hypothesis space is very expressive: with just two layers and nonlinear activation functions, neural networks can approximate any continuous function arbitrarily well.

Training a neural network can be done with gradient descent. There is an objective function, parameters are the weights on edges, and we need to configure these parameters to do well with respect to the objective function. Optimization is done using some variant of gradient descent.

Input Representation

We often have categorical features (i.e. type of restaurant: Mexican, French, Italian, fast-food, etc.) A natural way to represent each feature is to assign natural number ids: $1, 2, 3, \cdots$. However, then our neural network would think of the feature with id 3 and feature with id 4 as being "close" in some way, which doesn't make sense.

To resolve the issue of giving semantic meaning to adjacent numbers where there isn't any, we use 1-hot encoding: represent each feature as a unit vector. An example is shown below:



Output Representation

For **binary classification**, the output is interpreted as the probability of the positive class. For **multiclass classification** (i.e. categorizing animal pictures into d categories: dogs, cats, lions, tigers, etc.) we want d output nodes representing probabilities summing to 1. This is typically done via a **softmax** layer as it gives more weight to higher numbers/probabilities:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{d} e^{z_j}}$$

Convolutional Networks

Convolutional networks are neural networks appropriate for images and largely started the deep learning revolution. Note that images shouldn't be thought of as a vector of pixels, because adjacency matters. Furthermore, if there are n pixels and n units in the first hidden layer which are fully connected, there are already n^2 weights. Since images can be extremely large, there can be trillions of weights to train for a single image. Convolutional neural networks deal with the issues mentioned above using two ideas:

(i) To respect adjacency, each hidden unit receives input from a local region of the image.

(ii) Anything detectable in one local region would look the same in another region.

In order to introduce the architecture of convolutional networks, we first need to introduce **kernels** and **convolutions**.

Kernel and Convolution

A pattern of weights is a **kernel** and an application of the kernel is a **convolution**. Suppose a 1-D image is represented as a vector $\mathbf{x}$ of size n and a we have vector kernel k of size ℓ. The convolution operation is denoted by $\mathbf{z} = \mathbf{x} * \mathbf{k}$ and defined by

$$z_i = \sum_{j=1}^{\ell} k_j x_{i-(\ell+1)/2+j}$$

Pooling

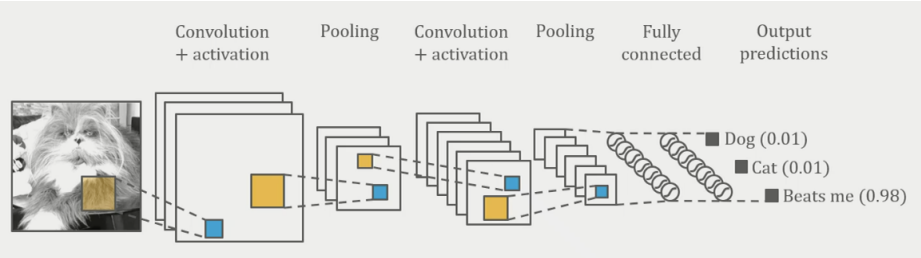
A **pooling layer** summarizes adjacent units from a preceding layer. It acts like a convolution with kernel size ℓ and stride s but the operation is fixed rather than learned and there's no activation function. Two different types of pooling are

(i) **Average pooling** computes the average value of the inputs. If $\ell = s$, this downsamples the image by a factor of s.

(ii) **Max pooling** computes the max value of the inputs. It acts like a logical disjunction, detecting a feature somewhere in the **receptive field**.

CNN Architecture

We can apply different kernels or pooling methods in parallel to the same image to get multiple new images. We do this to detect different things/information in the original image. Then, we can apply pooling to each of the new images in order to get a couple of pooling channels. Then, we can apply two different kernels to each of the three previous channels to get six channels, etc. As you can see, there are many possible architectures. Lastly, we input the resulting numbers into a softmax to get probabilities. A visual representation is included below:



The main takeaway is that the building blocks of CNN architecture are convolutions and pooling layers.

Recurrent Networks

Previously, we worked with acyclic neural networks. However, what if we dropped this assumption? This allows us to implement the idea of sequential memory.

In a **Recurrent neural networks (RNN)**, units take their own output as input, which simulate memory. RNNs are usually used to analyze sequential data, like HMMs. Therefore, we make the Markov assumption: there is a hidden state $\mathbf{z}_t$ which captures the relevant information from previous inputs. We update $\mathbf{z}_t = f_{\mathbf{w}}(\mathbf{z}_{t-1}, \mathbf{x}_t)$; note that this is a function of the previous state and new input.

Below is a diagram of a basic RNN. The new input $\mathbf{x}$ is fed using weight $\mathbf{w}_{xz}$ to $\mathbf{z}$, the previous output of $\mathbf{z}$ is fed into itself with weight $\mathbf{w}_{zz}$, and the output y for the current time step is weighted by $\mathbf{w}_z y$. Interestingly, $\mathbf{w}_{xz}$, $\mathbf{w}_{zz}$, $\mathbf{w}_{zy}$ aren't time dependent.

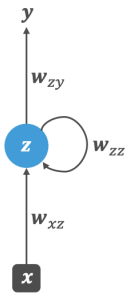


Diagram of a basic RNN
where the hidden layer
has recurrent connections

Practice Problems using Neural Networks

Problem 1 *Consider the following input image:*

1	2	2	1
0	1	2	1
1	2	0	0
0	2	2	1

Suppose we pass this image through a forward pass of a CNN. The first layer is a convolution with kernel

0	1
1	-1

with stride 2. *What is the output?*

1. After applying the first layer, we should have

1	2
0	1

Problem 2 *When passing images through CNNs, we can lose information at the boundaries. Why? And what is one way to deal with this?*

2. As seen in class, a 100×100 image passed through a 5×5 kernel with stride 1 is easily reduced to a 96×96 image; clearly, we lose information at the boundary. One way to deal with this is to pad the image.

Fairness

There is a large body of evidence for discrimination by AI algorithms- consider the disparities in accuracy for facial recognition technologies. We utilize mathematical constraints to define and reduce unfairness in algorithms.

We examine individual fairness, where we strive to guarantee fairness for individuals- intuitively, we want to ensure that similar individuals should be treated similarly. There is also the idea of group fairness, where you want to ensure that different groups (such as divided on demographic characteristics) are treated similarly.

Individual Fairness

An individual fairness problem is characterized by 4 main components outlined below

- V is our set of individuals.
- A is our set of outcomes.
- $d : V \times V \to \mathbb{R}^+$ is a metric on the individuals.
- $D : \Delta(A) \times \Delta(A) \to \mathbb{R}^+$ is a metric on the distributions over outcomes.

For a randomized classifier M to be individually fair, we check whether it satisfies the *Lipschitz Property*:

Definition 1 (Lipschitz Property) *If* $\forall x, y \in V, D(M(x), M(y)) \leq d(x, y)$, M *satisfies this property.*

This property ensures that similarly situated individuals (in terms of their distance d) are treated similarly such that the distance in their outcome distributions (captured by D) is at most their individual distance (d).

To ensure the Lipschitz property, we can construct an optimization problem as follows:

$$min \Sigma_{x \in V} \Sigma_{a \in A} \mu_x(a) \cdot L(x, a)$$

such that:

$$\forall x, y \in V, D(\mu_x, \mu_y) \leq d(x, y)$$
$$\forall x \in V, \mu_x \in \Delta(A)$$

Problem 3 *Consider the example of building an individually-fair classifier of whether individuals are approved for loans or not at Central Bank. How could you define an instance* (V, A, d, D) *of an individual fairness problem for this classifier? Assume that the Central Bank has access to a loan applicant's credit scores and their financial history.*

4. Note that there may be multiple solutions that work. One possible solution:

- V are all individuals who are applying for a loan at Central Bank
- The set of outcomes A is either that the individual was approved for a loan or not approved for a loan
- A metric d on the individuals could compute the absolute difference in credit scores. Note that we cannot simply compute the difference in credit scores as the range of values outputted by d must not be negative.
- A metric D could be the total variation distance between distributions. We will define Total Variation Distance shortly.

One widely-utilized metric for D is the *Total Variation Distance*:

$$D_{tv}(P, Q) = \frac{1}{2} \Sigma_{a \in A} |P(a) - Q(a)|$$

It is less clear what a good metric for d would be, as you are calculating the difference between individuals, and different metrics can result in very different Lipschitz classifiers.

Definition 2 (Envy-Freeness) *Recall from earlier, a randomized classifier M is envy-free iff $\forall x, y \in V$,*

$$E_{a \sim M(x)}[u_{xa}] \geq E_{a \sim M(y)}[u_{xa}]$$

where we have that each individual x has a utility u_{xa} for each outcome $a \in A$.

Envy-freeness relates to individual fairness as we want to ensure that each individual $x \in V$ is at least as happy with their outcome $a \sim M(x)$ as they are with another individual's y outcomes from the random classifier $a \sim M(y)$ in terms of their perceived utility u_{xa}.

Group Fairness

For group fairness problems, we have two groups based on a protected attribute (e.g. age above 40): $G \in \{0, 1\}$. Intuitively, with group fairness metrics, we want to ensure that individuals in different groups are treated similarly. Consider the case where have a binary classifier- outputs are of the form $\hat{Y} \in \{0, 1\}$.

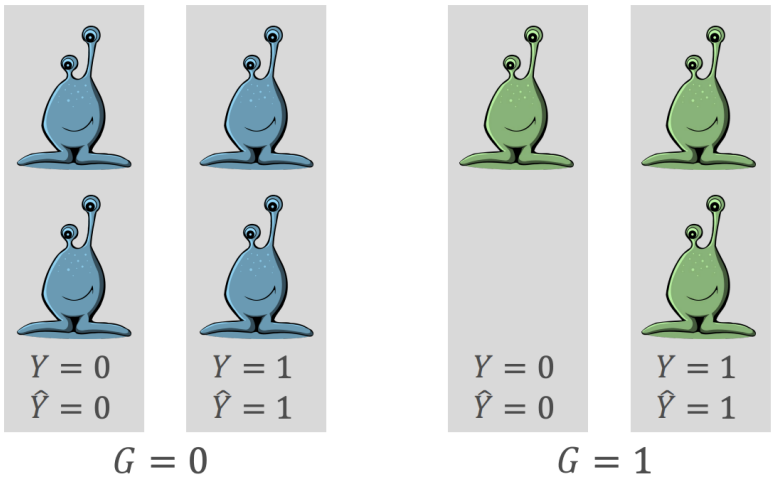
One intuitive notion of group fairness is demographic parity- ensure that equal proportions of each group are predicted to be 1.

Definition 3 (Demographic Parity) *A classifier satisfies demographic parity if*

$$\Pr[\hat{Y} = 1 | G = 0] = Pr[\hat{Y} = 1 | G = 1]$$

Problem 4 *Is demographic parity always guaranteed by a perfect classifier (classifier that is able to accurately predict every individual)? If so, explain why. If not, provide a counterexample.*

4. We can consider the example provided in lecture-



In this example, we have that equalized odds are satisfied as it is a perfect predictor so the false positive rate is 0 and the false negative rate is 0. Demographic parity, however, is not satisfied as the $\Pr[\hat{Y} = 1|G = 0] = 1/2 \neq Pr[\hat{Y} = 1|G = 1] = 2/3$.

Another group fairness guarantee is equalized odds- ensure that there are equal false positive and false negative rates across groups. Note that a guarantee of demographic parity does not necessarily guarantee equalized odds nor does a guarantee of equalized odds guarantee demographic parity.

Definition 4 (Equalized Odds) *A classifier satisfies equalized odds if* $\forall y, \hat{y} \in \{0, 1\}$*:*

$$\Pr[\hat{Y} = \hat{y}|G = 0, Y = y] = Pr[\hat{Y} = \hat{y}|G = 1, Y = y]$$

Expanding this out, we have four equations:

$$\Pr[\hat{Y} = 0|G = 0, Y = 0] = Pr[\hat{Y} = 0|G = 1, Y = 0]$$
$$\Pr[\hat{Y} = 0|G = 0, Y = 1] = Pr[\hat{Y} = 0|G = 1, Y = 1]$$
$$\Pr[\hat{Y} = 1|G = 0, Y = 0] = Pr[\hat{Y} = 1|G = 1, Y = 0]$$
$$\Pr[\hat{Y} = 1|G = 0, Y = 1] = Pr[\hat{Y} = 1|G = 1, Y = 1]$$

Problem 5 *Is equalized odds always guaranteed by a perfect classifier (classifier that is able to accurately predict every individual)? If so, explain why. If not, provide a counterexample.*

6. Yes, this is guaranteed because for a perfect classifier the false positive rate is 0 and the false negative rate is 0. These rates are equal so any perfect classifier satisfies equalized odds.

Calibration and Risk Assignments

Rather than outputting binary predictions, we can also consider the case where you output a risk assignment- a probability of having a positive label. Let σ be a feature vector for an individual and p_σ represents the fraction of people with feature vector σ with a true positive label. Again consider the membership of each individual in a group $G \in \{0, 1\}$.

As a result of this risk assignment, one goal for a classifier is to guarantee *calibration within groups*.

Definition 5 (Calibration within groups) *For each group G and bin b, the expected number of people from group G in b who belong to the positive class is a v_b fraction of the expected number of people from group G assigned to b.*

We often think about calibration in terms of weather- if you have a bin representing 30 percent chance of rain, you would expect it to rain exactly 30 percent of the total days classifed as having a 30 percent chance of rain. In this case, our 'bin' is all days with a 30 percent chance.

We cannot guarantee both equalized odds and calibration unless either:

- Perfect prediction- either $p_\sigma = 0$ or $p_\sigma = 1 \forall \sigma$. This means that each individual has probability either 0 or 1 of being in the positive class.
- Equal base rates- two groups have the same fraction of members in the positive class.