Artificial Intelligence: *Modeling Human Intelligence with Networks*

Jeová Farias Sales Rocha Neto
jeova_farias@brown.edu

Some news

- Last Friday presentations.

Some news

- Last Friday presentations.
- Next Friday presentations.

From the previous chapters

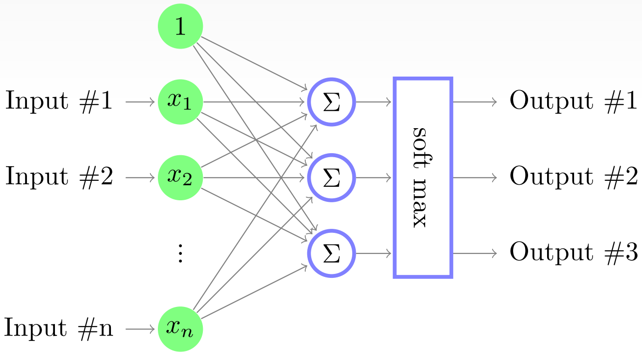
- We've learned a powerful optimization technique last class called

From the previous chapters

- We've learned a powerful optimization technique last class called **Gradient Descent**.

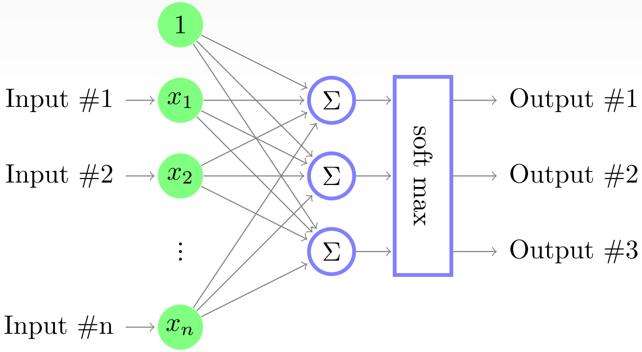
From the previous chapters

- We've learned a powerful optimization technique last class called **Gradient Descent**.
- We can come back to where we left our previous neural network:



From the previous chapters

- We've learned a powerful optimization technique last class called **Gradient Descent**.
- We can come back to where we left our previous neural network:



$$\text{Output} = o = [o_1, o_2, o_3] = \text{softmax}(Wx)$$

Loss Function

- What do we wanna minimize with Gradient Descent?

Loss Function

- What do we wanna minimize with Gradient Descent? A loss function (cf. this slide's title)!
- What is the loss we've been working with?

Loss Function

- What do we wanna minimize with Gradient Descent? A loss function (cf. this slide's title)!
- What is the loss we've been working with?
- The zero-one loss (for the perceptron algorithm):

$$\mathcal{L}(X, w) = \sum_{i=1}^{m} [f_{act}(w \cdot x^i) \neq class_i]$$

Loss Function

- What do we wanna minimize with Gradient Descent? A loss function (cf. this slide's title)!
- What is the loss we've been working with?
- The zero-one loss (for the perceptron algorithm):

$$\mathcal{L}(X, w) = \sum_{i=1}^{m} [f_{act}(w \cdot x^i) \neq class_i]$$

- What kind of output is the output of the softmax function?

Loss Function

- What do we wanna minimize with Gradient Descent? A loss function (cf. this slide's title)!
- What is the loss we've been working with?
- The zero-one loss (for the perceptron algorithm):

$$\mathcal{L}(X, w) = \sum_{i=1}^{m} [f_{act}(w \cdot x^i) \neq class_i]$$

- What kind of output is the output of the softmax function?
- Does it work in the zero-one loss? (say no)

Loss Function

- What do we wanna minimize with Gradient Descent? A loss function (cf. this slide's title)!
- What is the loss we've been working with?
- The zero-one loss (for the perceptron algorithm):

$$\mathcal{L}(X, w) = \sum_{i=1}^{m} [f_{act}(w \cdot x^i) \neq class_i]$$

- What kind of output is the output of the softmax function?
- Does it work in the zero-one loss? (say no) Why?

Loss Function

- What do we wanna minimize with Gradient Descent? A loss function (cf. this slide's title)!
- What is the loss we've been working with?
- The zero-one loss (for the perceptron algorithm):

$$\mathcal{L}(X, w) = \sum_{i=1}^{m} [f_{act}(w \cdot x^i) \neq class_i]$$

- What kind of output is the output of the softmax function?
- Does it work in the zero-one loss? (say no) Why?
- What to do? Cry?

Cross entropy

- First with need to define the Cross Entropy between two probability distributions (percentage) $p, q \in \mathbb{R}^n$ is:

$$\text{CrossEntropy}(p, q) = -\sum_{j=1}^{n} p_i \log(q_i)$$

Cross entropy

- First with need to define the Cross Entropy between two probability distributions (percentage) $p, q \in \mathbb{R}^n$ is:

$$\text{CrossEntropy}(p,q) = -\sum_{j=1}^{n} p_i \log(q_i)$$

Ex1.: $p = [.5, .2, .3]$, $q = [.3, .3, .4]$,
$\text{CrossEntropy}(p,q) \approx -.5 \times (-1.21) - .2 \times (-1.21) - .3 \times (-0.92) = 1.12$

Cross entropy

- First with need to define the Cross Entropy between two probability distributions (percentage) $p, q \in \mathbb{R}^n$ is:

$$\text{CrossEntropy}(p, q) = -\sum_{j=1}^{n} p_i \log(q_i)$$

Ex1.: $p = [.5, .2, .3]$, $q = [.3, .3, .4]$,
$\text{CrossEntropy}(p, q) \approx -.5 \times (-1.21) - .2 \times (-1.21) - .3 \times (-0.92) = 1.12$

Ex2.: $p = [1, 0, 0]$, $q = [1, 0, 0]$,
$\text{CrossEntropy}(p, q) = 1 \times (\log(1)) + 0 \times (\log(0)) + 0 \times (\log(0))$

Cross entropy

- First with need to define the Cross Entropy between two probability distributions (percentage) $p, q \in \mathbb{R}^n$ is:

$$\text{CrossEntropy}(p,q) = -\sum_{j=1}^{n} p_i \log(q_i)$$

Ex1.: $p = [.5, .2, .3]$, $q = [.3, .3, .4]$,
$\text{CrossEntropy}(p,q) \approx -.5 \times (-1.21) - .2 \times (-1.21) - .3 \times (-0.92) = 1.12$

Ex2.: $p = [1, 0, 0]$, $q = [1, 0, 0]$,
$\text{CrossEntropy}(p,q) = 1 \times (\log(1)) + 0 \times (\log(0)) + 0 \times (\log(0)) = 0$

Cross entropy

- First with need to define the Cross Entropy between two probability distributions (percentage) $p, q \in \mathbb{R}^n$ is:

$$\text{CrossEntropy}(p,q) = -\sum_{j=1}^{n} p_i \log(q_i)$$

Ex1.: $p = [.5, .2, .3]$, $q = [.3, .3, .4]$,
$\text{CrossEntropy}(p,q) \approx -.5 \times (-1.21) - .2 \times (-1.21) - .3 \times (-0.92) = 1.12$

Ex2.: $p = [1, 0, 0]$, $q = [1, 0, 0]$,
$\text{CrossEntropy}(p,q) = 1 \times (\log(1)) + 0 \times (\log(0)) + 0 \times (\log(0)) = 0$

Ex3.: $p = [1, 0, 0]$, $q = [0, 1, 0]$,
$\text{CrossEntropy}(p,q) = 1 \times (\log(0)) + 0 \times (\log(1)) + 0 \times (\log(0))$

Cross entropy

- First with need to define the Cross Entropy between two probability distributions (percentage) $p, q \in \mathbb{R}^n$ is:

$$\text{CrossEntropy}(p, q) = -\sum_{j=1}^{n} p_i \log(q_i)$$

Ex1.: $p = [.5, .2, .3]$, $q = [.3, .3, .4]$,
$\text{CrossEntropy}(p, q) \approx -.5 \times (-1.21) - .2 \times (-1.21) - .3 \times (-0.92) = 1.12$

Ex2.: $p = [1, 0, 0]$, $q = [1, 0, 0]$,
$\text{CrossEntropy}(p, q) = 1 \times (\log(1)) + 0 \times (\log(0)) + 0 \times (\log(0)) = 0$

Ex3.: $p = [1, 0, 0]$, $q = [0, 1, 0]$,
$\text{CrossEntropy}(p, q) = 1 \times (\log(0)) + 0 \times (\log(1)) + 0 \times (\log(0)) = \infty$

- Does the definition of Cross Entropy remind you of something we saw in class?

Cross entropy

- First with need to define the Cross Entropy between two probability distributions (percentage) $p, q \in \mathbb{R}^n$ is:

$$\text{CrossEntropy}(p, q) = -\sum_{j=1}^{n} p_i \log(q_i)$$

Ex1.: $p = [.5, .2, .3]$, $q = [.3, .3, .4]$,
$\text{CrossEntropy}(p, q) \approx -.5 \times (-1.21) - .2 \times (-1.21) - .3 \times (-0.92) = 1.12$

Ex2.: $p = [1, 0, 0]$, $q = [1, 0, 0]$,
$\text{CrossEntropy}(p, q) = 1 \times (\log(1)) + 0 \times (\log(0)) + 0 \times (\log(0)) = 0$

Ex3.: $p = [1, 0, 0]$, $q = [0, 1, 0]$,
$\text{CrossEntropy}(p, q) = 1 \times (\log(0)) + 0 \times (\log(1)) + 0 \times (\log(0)) = \infty$

- Does the definition of Cross Entropy remind you of something we saw in class? Dot product!

Cross entropy

- First with need to define the Cross Entropy between two probability distributions (percentage) $p, q \in \mathbb{R}^n$ is:

$$\text{CrossEntropy}(p,q) = -\sum_{j=1}^{n} p_i \log(q_i)$$

Ex1.: $p = [.5, .2, .3]$, $q = [.3, .3, .4]$,
$\text{CrossEntropy}(p,q) \approx -.5\times(-1.21) - .2\times(-1.21) - .3\times(-0.92) = 1.12$

Ex2.: $p = [1, 0, 0]$, $q = [1, 0, 0]$,
$\text{CrossEntropy}(p,q) = 1 \times (\log(1)) + 0 \times (\log(0)) + 0 \times (\log(0)) = 0$

Ex3.: $p = [1, 0, 0]$, $q = [0, 1, 0]$,
$\text{CrossEntropy}(p,q) = 1 \times (\log(0)) + 0 \times (\log(1)) + 0 \times (\log(0)) = \infty$

- Does the definition of Cross Entropy remind you of something we saw in class? Dot product!

$$\text{CrossEntropy}(p,q) = -p \cdot \log(q)$$

Cross entropy loss

- For the kinds of output of softmax (percentages), there's another loss called **Cross Entropy Loss**.

$$\begin{aligned}\mathcal{L}(X, W) &= \sum_{i=1}^{m} \mathrm{CrossEntropy}(y^i, o^i) \\ &= \sum_{i=1}^{m} \mathrm{CrossEntropy}(y^i, \mathrm{softmax}(Wx^i))\end{aligned}$$

Cross entropy loss

- For the kinds of output of softmax (percentages), there's another loss called **Cross Entropy Loss**.

$$\mathcal{L}(X,W) = \sum_{i=1}^{m} \text{CrossEntropy}(y^i, o^i)$$
$$= \sum_{i=1}^{m} \text{CrossEntropy}(y^i, \text{softmax}(Wx^i))$$

- There is a detail on y_i now:
  - Previously it was either -1 or 1.

Cross entropy loss

- For the kinds of output of softmax (percentages), there's another loss called **Cross Entropy Loss**.

$$\begin{aligned}\mathcal{L}(X, W) &= \sum_{i=1}^{m} \mathrm{CrossEntropy}(y^i, o^i) \\ &= \sum_{i=1}^{m} \mathrm{CrossEntropy}(y^i, \mathrm{softmax}(Wx^i))\end{aligned}$$

- There is a detail on y_i now:
  - Previously it was either -1 or 1.
  - Now we know that the classes are 0, 1 or 2.

Cross entropy loss

- For the kinds of output of softmax (percentages), there's another loss called **Cross Entropy Loss**.

$$\begin{aligned}\mathcal{L}(X,W) &= \sum_{i=1}^{m} \mathrm{CrossEntropy}(y^i, o^i) \\ &= \sum_{i=1}^{m} \mathrm{CrossEntropy}(y^i, \mathrm{softmax}(Wx^i))\end{aligned}$$

- There is a detail on y_i now:
  - Previously it was either -1 or 1.
  - Now we know that the classes are 0, 1 or 2.
  - So, we should create, for each y_i, a vector Y^i *one-hot*.

Cross entropy loss

- For the kinds of output of softmax (percentages), there's another loss called **Cross Entropy Loss**.

$$\begin{aligned}\mathcal{L}(X, W) &= \sum_{i=1}^{m} \text{CrossEntropy}(y^i, o^i) \\ &= \sum_{i=1}^{m} \text{CrossEntropy}(y^i, \text{softmax}(Wx^i))\end{aligned}$$

- There is a detail on y_i now:
  - Previously it was either -1 or 1.
  - Now we know that the classes are 0, 1 or 2.
  - So, we should create, for each y_i, a vector Y^i *one-hot*.
  - Example: if $y_i = 1$, $Y^i = [0, 1, 0]$. If $y_i = 0$, $Y^i = [1, 0, 0]$.

Cross entropy loss

- For the kinds of output of softmax (percentages), there's another loss called **Cross Entropy Loss**.

$$\begin{aligned}\mathcal{L}(X,W) &= \sum_{i=1}^{m} \mathrm{CrossEntropy}(y^i, o^i)\\ &= \sum_{i=1}^{m} \mathrm{CrossEntropy}(y^i, \mathrm{softmax}(Wx^i))\end{aligned}$$

- There is a detail on y_i now:
  - Previously it was either -1 or 1.
  - Now we know that the classes are 0, 1 or 2.
  - So, we should create, for each y_i, a vector Y^i *one-hot*.
  - Example: if $y_i = 1$, $Y^i = [0,1,0]$. If $y_i = 0$, $Y^i = [1,0,0]$.
- Jupyter Notebooks!

Gradient Descent on the Multilayer perceptron

- Ok, now we need to find a $W = [w^0, w^1, w^2]$ that minimizes $\mathcal{L}(X, W)$.

$$\mathcal{L}(X, W) = \sum_{i=1}^{m} \text{CrossEntropy}(y^i, \text{softmax}(Wx^i)) \tag{1}$$

Gradient Descent on the Multilayer perceptron

- Ok, now we need to find a $W = [w^0, w^1, w^2]$ that minimizes $\mathcal{L}(X, W)$.

$$\mathcal{L}(X, W) = \sum_{i=1}^{m} \text{CrossEntropy}(y^i, \text{softmax}(Wx^i)) \tag{1}$$

- Let's use Gradient Descent!

Gradient Descent on the Multilayer perceptron

- Ok, now we need to find a $W = [w^0, w^1, w^2]$ that minimizes $\mathcal{L}(X, W)$.

$$\mathcal{L}(X, W) = \sum_{i=1}^{m} \text{CrossEntropy}(y^i, \text{softmax}(Wx^i)) \tag{1}$$

- Let's use Gradient Descent! Wait,

Gradient Descent on the Multilayer perceptron

- Ok, now we need to find a $W = [w^0, w^1, w^2]$ that minimizes $\mathcal{L}(X, W)$.

$$\mathcal{L}(X, W) = \sum_{i=1}^{m} \text{CrossEntropy}(y^i, \text{softmax}(Wx^i)) \qquad (1)$$

- Let's use Gradient Descent! Wait, how do we do it?

Gradient Descent on the Multilayer perceptron

- Ok, now we need to find a $W = [w^0, w^1, w^2]$ that minimizes $\mathcal{L}(X, W)$.

$$\mathcal{L}(X, W) = \sum_{i=1}^{m} \text{CrossEntropy}(y^i, \text{softmax}(Wx^i)) \tag{1}$$

- Let's use Gradient Descent! Wait, how do we do it?
- First notice that we want to find W that minimizes $\mathcal{L}(X, W)$.

Gradient Descent on the Multilayer perceptron

- Ok, now we need to find a $W = [w^0, w^1, w^2]$ that minimizes $\mathcal{L}(X, W)$.

$$\mathcal{L}(X, W) = \sum_{i=1}^{m} \text{CrossEntropy}(y^i, \text{softmax}(Wx^i)) \tag{1}$$

- Let's use Gradient Descent! Wait, how do we do it?
- First notice that we want to find W that minimizes $\mathcal{L}(X, W)$.
- So the gradient descent iterations are like:

$$W = W - \eta \frac{\mathrm{d}\mathcal{L}(X, W)}{\mathrm{d}W},$$

where η is the step size.

Gradient Descent on the Multilayer perceptron

- Ok, now we need to find a $W = [w^0, w^1, w^2]$ that minimizes $\mathcal{L}(X, W)$.

$$\mathcal{L}(X, W) = \sum_{i=1}^{m} \text{CrossEntropy}(y^i, \text{softmax}(Wx^i)) \tag{1}$$

- Let's use Gradient Descent! Wait, how do we do it?
- First notice that we want to find W that minimizes $\mathcal{L}(X, W)$.
- So the gradient descent iterations are like:

$$W = W - \eta \frac{\mathrm{d}\mathcal{L}(X, W)}{\mathrm{d}W},$$

  where η is the step size.
- How to compute the $\frac{\mathrm{d}\mathcal{L}(X,W)}{\mathrm{d}W}$?

Gradient Descent on the Multilayer perceptron

- Chain Rule!

Gradient Descent on the Multilayer perceptron

- Chain Rule!
- First, write $\mathcal{L}(X, W)$ as $\mathcal{L}(X, W) = \sum_{i=1}^{m} h_i(W)$ and $h_i(W) = \text{CrossEntropy}(y^i, \text{softmax}(Wx^i))$

Gradient Descent on the Multilayer perceptron

- Chain Rule!
- First, write $\mathcal{L}(X,W)$ as $\mathcal{L}(X,W) = \sum_{i=1}^{m} h_i(W)$ and $h_i(W) = \text{CrossEntropy}(y^i, \text{softmax}(Wx^i))$
- Now we have that:

$$\begin{aligned} h_i(W) &= g_i(f_i(W)) \text{ with} \\ g_i(v) &= \text{CrossEntropy}(y^i, \text{softmax}(v)) \\ f(W) &= Wx^i \end{aligned} \tag{2}$$

Gradient Descent on the Multilayer perceptron

- Chain Rule!
- First, write $\mathcal{L}(X, W)$ as $\mathcal{L}(X, W) = \sum_{i=1}^{m} h_i(W)$ and $h_i(W) = \text{CrossEntropy}(y^i, \text{softmax}(Wx^i))$
- Now we have that:

$$
\begin{aligned}
h_i(W) = g_i(f_i(W)) \text{ with} \\
g_i(v) &= \text{CrossEntropy}(y^i, \text{softmax}(v)) \\
f(W) &= Wx^i
\end{aligned}
\tag{2}
$$

- Then, we have:

$$
\frac{\mathrm{d}h_i(W)}{\mathrm{d}W} = \frac{\mathrm{d}g(f(W))}{\mathrm{d}f(W)} \frac{\mathrm{d}f(W)}{\mathrm{d}W}
$$

Gradient Descent on the Multilayer perceptron

- Chain Rule!
- First, write $\mathcal{L}(X,W)$ as $\mathcal{L}(X,W) = \sum_{i=1}^{m} h_i(W)$ and $h_i(W) = \text{CrossEntropy}(y^i, \text{softmax}(Wx^i))$
- Now we have that:

$$\begin{aligned} h_i(W) = g_i(f_i(W)) \text{ with} & \\ g_i(v) &= \text{CrossEntropy}(y^i, \text{softmax}(v)) \\ f(W) &= Wx^i \end{aligned} \tag{2}$$

- Then, we have:

$$\frac{\mathrm{d}h_i(W)}{\mathrm{d}W} = \frac{\mathrm{d}g(f(W))}{\mathrm{d}f(W)} \frac{\mathrm{d}f(W)}{\mathrm{d}W}$$

- Notice that (after a looot of algebra):

$$\frac{\mathrm{d}g(f(W))}{\mathrm{d}f(W)} = \text{softmax}(Wx^i) - y^i$$

Gradient Descent on the Multilayer perceptron

- Chain Rule!
- First, write $\mathcal{L}(X, W)$ as $\mathcal{L}(X, W) = \sum_{i=1}^{m} h_i(W)$ and $h_i(W) = \text{CrossEntropy}(y^i, \text{softmax}(Wx^i))$
- Now we have that:

$$\begin{aligned} h_i(W) = g_i(f_i(W)) \text{ with} \\ g_i(v) = \text{CrossEntropy}(y^i, \text{softmax}(v)) \\ f(W) = Wx^i \end{aligned} \tag{2}$$

- Then, we have:

$$\frac{\mathrm{d}h_i(W)}{\mathrm{d}W} = \frac{\mathrm{d}g(f(W))}{\mathrm{d}f(W)} \frac{\mathrm{d}f(W)}{\mathrm{d}W}$$

- Notice that (after a looot of algebra):

$$\frac{\mathrm{d}g(f(W))}{\mathrm{d}f(W)} = \text{softmax}(Wx^i) - y^i = o^i - y^i,$$

Gradient Descent on the Multilayer perceptron

- Chain Rule!
- First, write $\mathcal{L}(X, W)$ as $\mathcal{L}(X, W) = \sum_{i=1}^{m} h_i(W)$ and $h_i(W) = \text{CrossEntropy}(y^i, \text{softmax}(Wx^i))$
- Now we have that:

$$h_i(W) = g_i(f_i(W)) \text{ with} \tag{2}$$
$$g_i(v) = \text{CrossEntropy}(y^i, \text{softmax}(v))$$
$$f(W) = Wx^i$$

- Then, we have:

$$\frac{\mathrm{d}h_i(W)}{\mathrm{d}W} = \frac{\mathrm{d}g(f(W))}{\mathrm{d}f(W)} \frac{\mathrm{d}f(W)}{\mathrm{d}W}$$

- Notice that (after a looot of algebra):

$$\frac{\mathrm{d}g(f(W))}{\mathrm{d}f(W)} = \text{softmax}(Wx^i) - y^i = o^i - y^i, \quad \frac{\mathrm{d}f(W)}{\mathrm{d}W} = x^i$$

Gradient Descent on the Multilayer perceptron

- And finally:

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{L}(X,W)}{\mathrm{d}W} &= \frac{\mathrm{d}\left(\sum_{i=1}^{m} h_i(W)\right)}{\mathrm{d}W} \\
&= \sum_{i=1}^{m} \frac{\mathrm{d}h_i(W)}{\mathrm{d}W} \\
&= \sum_{i=1}^{m} (y^i - \mathrm{softmax}(Wx^i)) \cdot x^i, \\
&= \sum_{i=1}^{m} (o^i - y^i) \cdot x^i, \\
&= (O - Y)X,
\end{aligned}$$

  with y_i being a one-hot vector and Y the concatenation of y^i.
- So the Gradient Descent update rule is

$$W = W - \eta \sum_{i=1}^{m} (y^i - \mathrm{softmax}(Wx^i))x^i,$$

Gradient Descent on the Multilayer perceptron

- And finally:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}(X,W)}{\mathrm{d}W} &= \frac{\mathrm{d}\left(\sum_{i=1}^{m} h_i(W)\right)}{\mathrm{d}W}\\ &= \sum_{i=1}^{m}\frac{\mathrm{d}h_i(W)}{\mathrm{d}W}\\ &= \sum_{i=1}^{m}(y^i - \mathrm{softmax}(Wx^i))\cdot x^i,\\ &= \sum_{i=1}^{m}(o^i - y^i)\cdot x^i,\\ &= (O-Y)X,\end{aligned}$$

  with y_i being a one-hot vector and Y the concatenation of y^i.
- So the Gradient Descent update rule is

$$W = W - \eta\sum_{i=1}^{m}(y^i - \mathrm{softmax}(Wx^i))x^i,$$

Gradient Descent on the Multilayer perceptron

Algorithm 1 $GradientDescent - MultilayerPerceptron$

Input: A matrix of points X, a vector of classes y

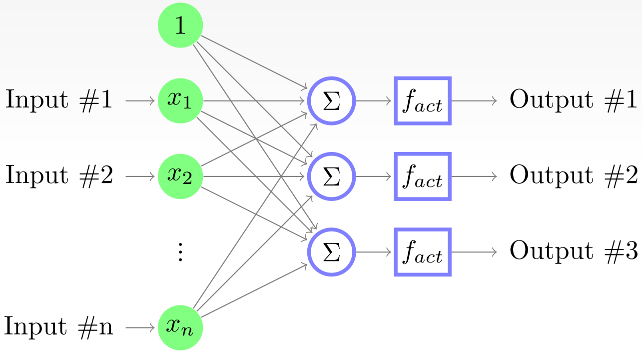
Output: A matrix of weights W.

```
Add bias to X
Find Y as the one hot version of y.
Initialize W
η = 1
for a given number of epochs (repetitions) do
    O = softmax(WX) ⟼ Feed Forward step
    dL(X,W)/dW = (O − Y)X ⟼ Back Propagation step
    W = W − η dL(X,W)/dW
end for
```

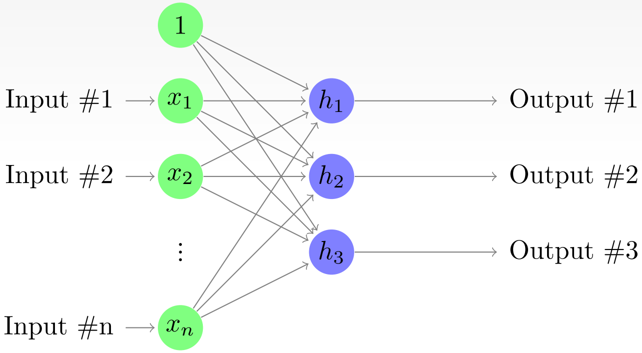
$$
\begin{aligned}
&\text{Add bias to } X\\
&\text{Find } Y \text{ as the one hot version of } y.\\
&\text{Initialize } W\\
&\eta = 1\\
&\textbf{for } \text{a given number of epochs (repetitions) } \textbf{do}\\
&\quad O = \mathrm{softmax}(WX) \longmapsto \text{Feed Forward step}\\
&\quad \frac{\mathrm{d}\mathcal{L}(X,W)}{\mathrm{d}W} = (O - Y)X \longmapsto \text{Back Propagation step}\\
&\quad W = W - \eta\frac{\mathrm{d}\mathcal{L}(X,W)}{\mathrm{d}W}\\
&\textbf{end for}
\end{aligned}
$$

Neural Evolution (*Part II*)

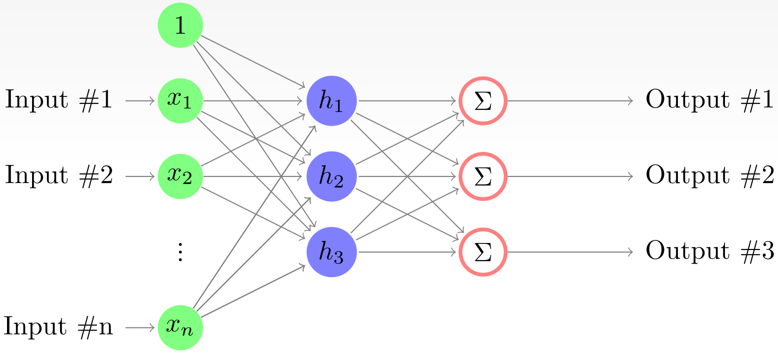
Neural Evolution



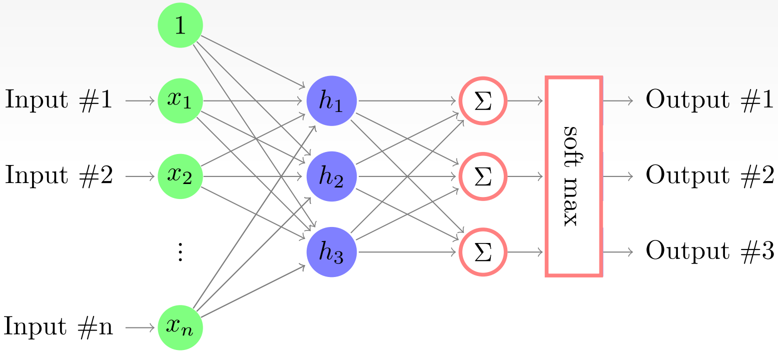
Neural Evolution



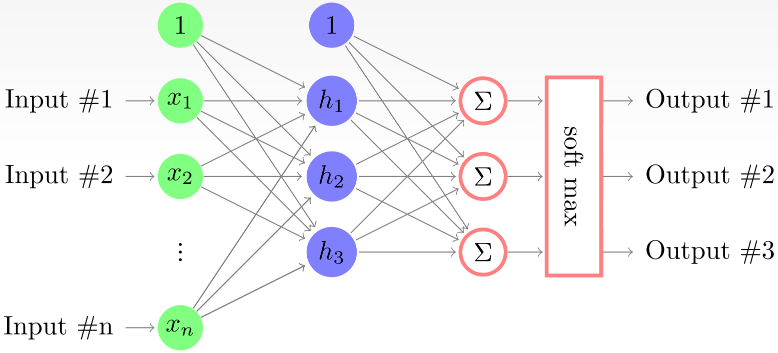
Neural Evolution



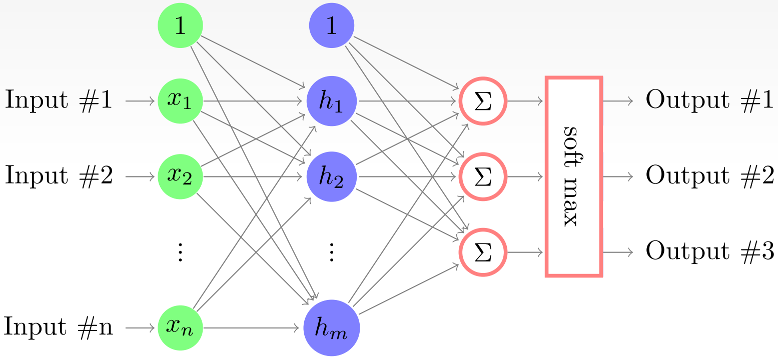
Neural Evolution



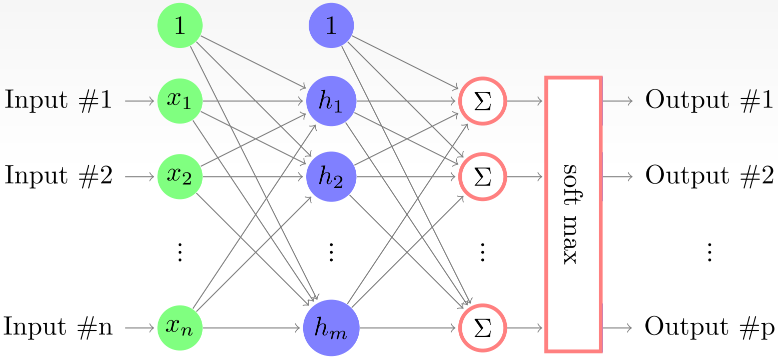
Neural Evolution



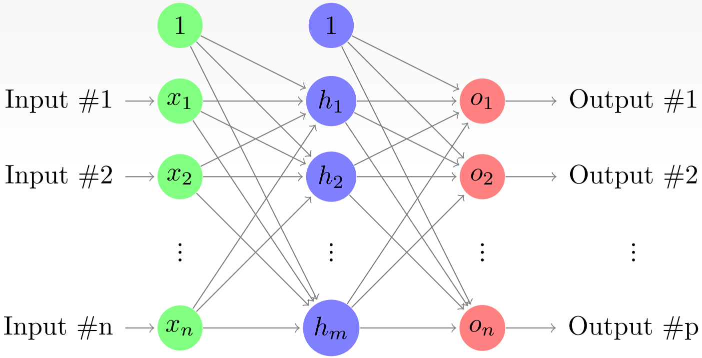
Neural Evolution



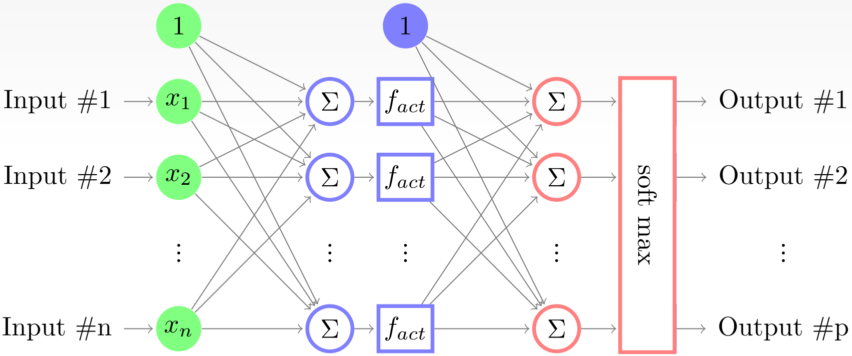
Neural Evolution



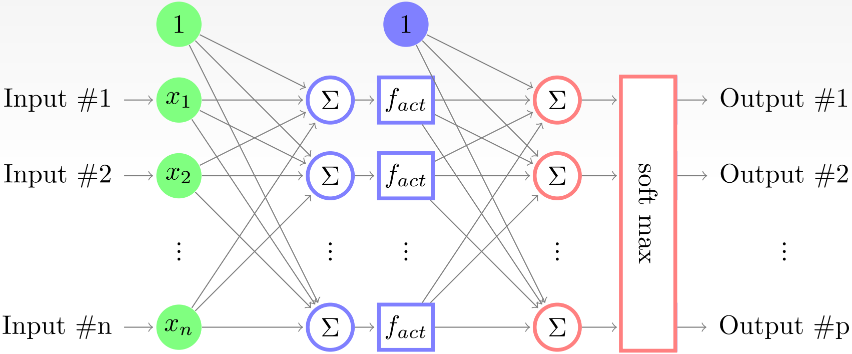
Neural Evolution



Neural Evolution

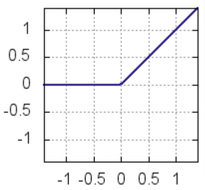


Neural Evolution

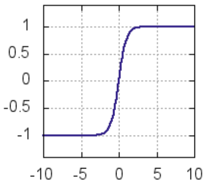


$$o = \text{softmax}(Wh) = \text{softmax}(W(f_{\text{act}}(W_h x)))$$

Activation Functions



(a) ReLU: $f(x) = \max(0, x)$



(b) tanh: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$