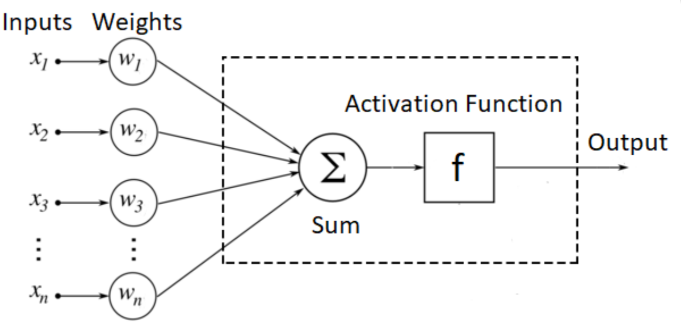
Lecture 2: Introduction to Neural Networks. Deep Cross-Entropy Method

Anton Plaksin

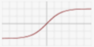
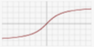
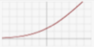
Perceptron (Neuron)



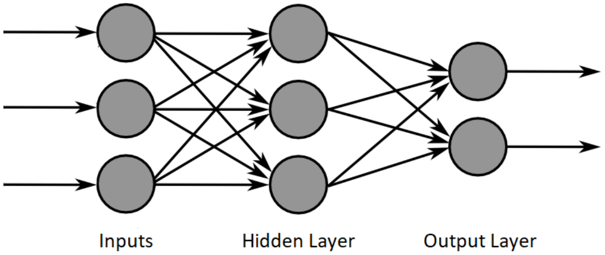
На картинке представлен нейрон, который представляет собой сумму входных параметров $x_1 ... x_n$ умноженных на весовые коэффициенты при них $w_1 ... w_n$ (что собственно представляет собой формулы линейной регрессии) и функцию активации (перевод выхода сумму в область интересующих нас значений).

$$y = f\left(\sum_{i=1}^{n} w_i x_i\right) \quad \text{or} \quad y = f\left(b + \sum_{i=1}^{n} w_i x_i\right)$$

Activation Functions

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Neuron Network

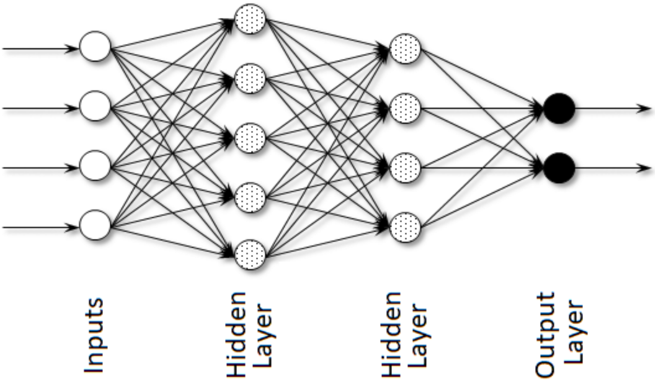


$$F_j^\theta(X) = f_{out}\left(b_j + \sum_{k=1}^{3} w_{j,k} f\left(\hat{b}_k + \sum_{i=1}^{3} \hat{w}_{k,i} x_i\right)\right), \quad j \in \overline{1,2}.$$

$$F^\theta(X) \in \mathbb{R}^2, \quad X = (x_1, x_2, x_3) \in \mathbb{R}^3,$$

$$\theta = (b_1, b_2, w_{1,1}, \dots, w_{2,3}, \hat{b}_1, \hat{b}_2, \hat{b}_3, \hat{w}_{1,1}, \dots, \hat{w}_{3,3}) \in \mathbb{R}^{20}$$

Neuron Network



$$F_j^\theta(X) = f_{out}\left(b_j + \sum_{k=1}^{4} w_{j,k} f\left(\hat{b}_k + \sum_{l=1}^{5} \hat{w}_{k,l} f\left(\tilde{b}_l + \sum_{i=1}^{4} \tilde{w}_{l,i} x_i\right)\right)\right), \quad j \in \overline{1,2}.$$

$$F^\theta(X) \in \mathbb{R}^2, \quad X \in \mathbb{R}^4, \quad \theta \in \mathbb{R}^{59}$$

Regression

- $$\{(X_1, Y_1), (X_2, Y_2), \dots, (X_k, Y_k)\}, \quad X_i \in \mathbb{R}^n, \quad Y_i \in \mathbb{R}^m$$
- $$\Downarrow$$
  - $$\sum_{i=1}^{k} \|F(X_i) - Y_i\|^2 \to \min_{F \text{ is continuous}}$$

Нейронная сеть является отображением признаков X в пространство зависимой переменной Y

Neural Network Approach

Set a NN structure F^θ and initial parameters θ_0

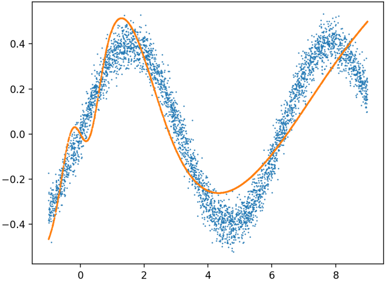
Define $Loss(\theta) = \frac{1}{k} \sum_{i=1}^{k} \|F^\theta(X_i) - Y_i\|^2$

Solve the optimization problem $Loss(\theta) \to \min_\theta$

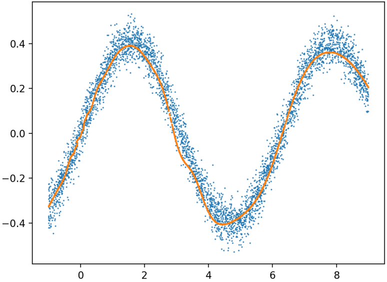
Gradient Descent: $\theta_{j+1} = \theta_j - \eta \nabla_\theta Loss(\theta)$, $\eta > 0$

Построение нейросети заключается в создании ее структуры (слои, количество нейронов в слоях, функции активации) и инициализации начальных весов.
Определении функции потерь и ее минимизация методом градиентного бустинга.

Example



128 neurons



$32 \times 128 \times 32$ neurons

Cybenko Theorem (1989)

Theorem

For every continuous function $G\colon [0,1]^n \mapsto \mathbb{R}^m$ and every $\varepsilon > 0$, there exist N, w_i, $\hat{b}_i$, $\hat{w}_{i,j}$, $i \in \overline{1,N}$, $j \in \overline{1,n}$ such that

$$\|F^\theta(X) - G(X)\| \le \varepsilon, \quad X \in [0,1]^n,$$

where

$$F^\theta(X) = \sum_{i=1}^{N} w_i \sigma\left(\hat{b}_i + \sum_{j=1}^{n} \hat{w}_{i,j} x_j\right),$$

$$X = (x_1, \ldots, x_n), \quad \theta = (w_1, \ldots w_N, \hat{b}_1, \ldots, \hat{b}_N, \hat{w}_{1,1}, \ldots, \hat{w}_{N,n})$$

Теорема Цыбенко

О том, что непрерывную функцию можно аппроксимировать однослойной нейросетью с некоторой точностью с определенным количеством нейронов, однако на практике достаточно часто применяют многослойные нейросети.

Gradient Calculation

$$Loss(\theta) = \sum_{i=1}^{k} \|F^{\theta}(X_i) - Y_i\|^2,$$

$$F_j^{\theta}(X) = f_{out}\left(b_j + \sum_{k=1}^{3} w_{j,k} f\left(\hat{b}_k + \sum_{i=1}^{3} \hat{w}_{k,i} x_i\right)\right), \quad j \in \overline{1,2}.$$

$$\frac{\partial Loss(\theta)}{\partial \hat{w}_{2,3}} = 2 \sum_{i=1}^{k} \langle F^{\theta}(X_i) - Y_i, \frac{\partial F^{\theta}(X_i)}{\partial \hat{w}_{2,3}} \rangle$$

$$F_j^{\theta}(X) = f_{out}\left(b_j + \sum_{k=1}^{3} w_{j,k} f\left(\hat{b}_k + \sum_{i=1}^{3} \hat{w}_{k,i} x_i\right)\right), \quad j \in \overline{1,2}.$$

$$\begin{aligned} \frac{\partial F_j^{\theta}(X)}{\partial \hat{w}_{2,3}} &= f'_{out}\left(b_j + \sum_{k=1}^{3} w_{j,k} f\left(\hat{b}_k + \sum_{i=1}^{3} \hat{w}_{k,i} x_i\right)\right) \\ &\times \quad w_{j,2} f'\left(\hat{b}_2 + \sum_{i=1}^{3} \hat{w}_{2,i} x_i\right) x_3 \end{aligned}$$

Stochastic Gradient Descent

Randomly choose $\{(X, Y)\}$

Define $Loss(\theta) = \|F^{\theta}(X) - Y\|^2$

Update parameters $\theta_{j+1} = \theta_j - \eta \nabla_{\theta} Loss(\theta)$

Стохастический градиентный спуск, когда вычисляется не полный градиент, а градиент по одному (случайному) наблюдению из признакового пространства и движение происходит в сторону его (несмотря на более неточное движение в сторону минимума, он все равно сходится при правильном подборе шага обучения), но существенно менее сложный в вичислительном отношении.

Также существует минибатч спуск, когда градиент вычисляется на некоторой выборке.

Classification

$$\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_k, Y_k)\}, X_i \in \mathbb{R}^n, Y_i \in 1...\mathrm{m}$$

$$\Downarrow$$

$$\left|\{i \in \overline{1, k} \mid F(X_i) \neq Y_i\}\right| \to \min_F$$

$$S_i = \mathrm{Softmax}(Z)_i = \frac{e^{z_i}}{\sum\limits_{j=1}^{m} e^{z_j}}, \quad i \in \overline{1, m}, \quad Z = (z_1, \ldots, z_m)$$

$$S_i \in (0, 1), \ i \in \overline{1, m}, \quad \sum_{i=1}^{m} S_i = 1.$$

При классификации для отображения признакового пространства используют функцию Softmax из выхода которой берется только значение, соответствующее реальному классу объекта обучающей выборки.
Softmax обладает свойствами перевода вектора в вектор со свойствами на слайде, который можно интерпритировать как вероятноть принадлежности к классу.

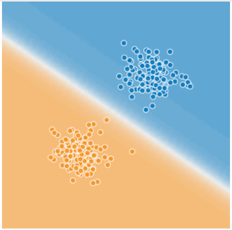
Classification

Neural Network Approach

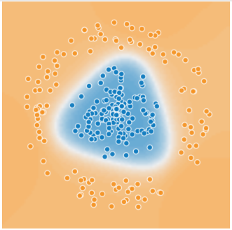
- Set a NN structure F^θ and initial parameters θ_0
- Define $Loss(\theta) = -\frac{1}{k}\sum_{i=1}^{k} \ln \mathrm{Softmax}(F^\theta(X_i))_{Y_i}$
- Solve the optimization problem $Loss(\theta) \mapsto \min_\theta$
  - Gradient Descent: $\theta_{j+1} = \theta_j - \eta \nabla_\theta Loss(\theta)$

Функция потерь определяется как отрицательный логарифм, поскольку логарифм 1 равен 0, а значения меньше 1 (но больше 0) положительное число, поэтому для того чтобы функцию минимизировать вводится знак минус.

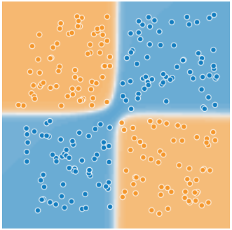
Examples



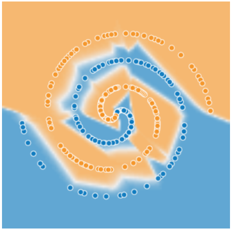
1 layer × 1 neuron



1 layer × 3 neuron



2 layer × 3 neuron



4 layer × 8 neuron

https://playground.tensorflow.org

Neural Network Elements

- Structure (number of layers and neurons, activation functions)
- Loss Functions (MSE, L1, Cross Entropy, ...)
- Gradient Descent Methods (SGD, Adam, Adadelta, ...)
- Weight initialization
- Regularization (L2, L1, Batch normalization, Dropout, ...)

С. Николенко, Е. Архангельская «Глубокое обучение. Погружение в мир нейронных сетей»

Markov Decision Process

Markov Property

- $$\mathbb{P}[S_{t+1}|S_t, A_t] = \mathbb{P}[S_{t+1}|S_1, A_1, S_2, A_2 \ldots, S_t, A_t]$$
- $$\mathbb{P}[R_t|S_t, A_t] = \mathbb{P}[R_t|S_1, A_1, S_2, A_2 \ldots, S_t, A_t] = 1$$

Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

$\mathcal{S}$ — a ~~finite ($|\mathcal{S}| = n$)~~ state space

$\mathcal{A}$ — a ~~finite ($|\mathcal{A}| = m$)~~ action space

$\mathcal{P}$ — a deterministic transition probability function

$$\mathcal{P}(s'|s, a) = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]$$

$\mathcal{P}_0$ — a deterministic initial state function

$\mathcal{R}$ — a reward function

$$\mathcal{R}(s, a) = R_t \quad \Leftrightarrow \quad \mathbb{P}[R_t|S_t = s, A_t = a] = 1$$

$\gamma \in [0, 1]$ — discount coefficient

Cross-Entropy Method. Case of $\mathcal{S} \subset \mathbb{R}^n$ и $\mathcal{A} \subset \mathbb{R}^m$

Let $\pi^\theta \colon \mathbb{R}^n \mapsto \mathbb{R}^m$ be a neural network, θ_0 be initial parameters, N be a number of iterations, K be a number of trajectories, $q \in (0,1)$ be a parameter for defning «elite» trajectories, $\eta > 0$ be a learning rate, $\varepsilon = 1$ be an exploration parameters.
For each $n \in \overline{1,N}$, do

Задача кросс-энтропии с применением нейросети представляет собой схожую с уже изученным алгоритмом задачу, за исключением отсутствия матрицы состояние-действие, а выдачи текущего действия нейросетью, которая на вход принимает состояние.

На этапе **оценки** после инициализации нейросети случайными весами она будет некоторым детерминированным образом генерировать ответ (текущее действие) на входное состояние среды. Для формирования множества траекторий (добавлении исследования) к выходу нейросети добавляется выход функции шума, далее выбираются элитные траектории и проводится улучшение политики (обновление весов нейросети).

На этапе **улучшения** необходимо определить функцию потерь (на слайде) в которой для каждой траектории необходимо считаем действия, отличающие от "элитного" действия и оптимизируем веса нейросети минимизируя функцию потерь градиентным бустингом.

Cross-Entropy Method. Case of $\mathcal{S} \subset \mathbb{R}^n$ и $\mathcal{A} \subset \mathbb{R}^m$

Let $\pi^\theta \colon \mathbb{R}^n \mapsto \mathbb{R}^m$ be a neural network, θ_0 be initial parameters, N be a number of iterations, K be a number of trajectories, $q \in (0,1)$ be a parameter for defning «elite» trajectories, $\eta > 0$ be a learning rate, $\varepsilon = 1$ be an exploration parameters.
For each $n \in \overline{1,N}$, do

(Policy evaluation) Acting in accordance with the policy

$$\pi_n(s) = \left[\pi^{\theta_n}(s) + Noise(\varepsilon)\right]_{\mathcal{A}},$$

get K trajectories τ_k and total rewards $G(\tau_k)$. Evaluate π_n:

$$\mathbb{E}_{\pi_n}[G] \approx V_n := \frac{1}{K}\sum_{k=1}^{K} G(\tau_k)$$

Cross-Entropy Method. Case of $\mathcal{S} \subset \mathbb{R}^n$ и $\mathcal{A} \subset \mathbb{R}^m$

Let $\pi^\theta \colon \mathbb{R}^n \mapsto \mathbb{R}^m$ be a neural network, θ_0 be initial parameters, N be a number of iterations, K be a number of trajectories, $q \in (0,1)$ be a parameter for defning «elite» trajectories, $\eta > 0$ be a learning rate, $\varepsilon = 1$ be an exploration parameters.
For each $n \in \overline{1,N}$, do

(Policy improvement) Select «elite» trajectories $\mathcal{T}_n = \{\tau_k, k \in \overline{1,K} \colon G(\tau_k) > \gamma_q\}$, where γ_q is a q-quantile of the numbers $G(\tau_k)$, $k \in \overline{1,K}$. Define

$$Loss(\theta) = \frac{1}{|\mathcal{T}_n|} \sum_{(a|s) \text{«}\in\text{»} \mathcal{T}_n} \|\pi^{\theta_n}(s) - a\|^2$$

and update parameters by the (stochastic) gradient descent:

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta Loss(\theta_n).$$

Reduce ε ($\varepsilon = 1/N$).

Cross-Entropy Method. Case of $\mathcal{S} \subset \mathbb{R}^n$ и $|\mathcal{A}| = m$

Let $F^\theta \colon \mathbb{R}^n \mapsto \mathbb{R}^m$ be a neural network. Define

$$\pi^\theta(i|s) = \text{Softmax}(F^\theta(s))_i, \quad \pi_{\text{uniform}}(i|s) = 1/m, \quad i \in \overline{1, m}.$$

Let θ_0 be initial parameters, N be a number of iterations,
K be a number of trajectories, $q \in (0, 1)$ be a parameter for defning «elite» trajectories, $\eta > 0$ be a learning rate,
$\varepsilon = 1$ be an exploration parameters.
For each $n \in \overline{1, N}$, do

Cross-Entropy Method. Case of $\mathcal{S} \subset \mathbb{R}^n$ и $|\mathcal{A}| = m$

Let $F^\theta \colon \mathbb{R}^n \mapsto \mathbb{R}^m$ be a neural network. Define

$$\pi^\theta(i|s) = \text{Softmax}(F^\theta(s))_i, \quad \pi_{\text{uniform}}(i|s) = 1/m, \quad i \in \overline{1,m}.$$

Let θ_0 be initial parameters, N be a number of iterations,
K be a number of trajectories, $q \in (0,1)$ be a parameter for defning «elite» trajectories, $\eta > 0$ be a learning rate,
$\varepsilon = 1$ be an exploration parameters.
For each $n \in \overline{1,N}$, do

(Policy evaluation) Acting in accordance with the policy

$$\pi_n(\cdot|s) = (1-\varepsilon)\pi^{\theta_n}(\cdot|s) + \varepsilon\pi_{\text{uniform}}(\cdot|s),$$

get K trajectories τ_k and total rewards $G(\tau_k)$. Evaluate π_n:

$$\mathbb{E}_{\pi_n}[G] \approx V_n := \frac{1}{K}\sum_{k=1}^{K} G(\tau_k)$$

Cross-Entropy Method. Case of $\mathcal{S} \subset \mathbb{R}^n$ и $|\mathcal{A}| = m$

Let $F^\theta \colon \mathbb{R}^n \mapsto \mathbb{R}^m$ be a neural network. Define

$$\pi^\theta(i|s) = \text{Softmax}(F^\theta(s))_i, \quad \pi_{\text{uniform}}(i|s) = 1/m, \quad i \in \overline{1,m}.$$

Let θ_0 be initial parameters, N be a number of iterations,
K be a number of trajectories, $q \in (0,1)$ be a parameter for defning «elite» trajectories, $\eta > 0$ be a learning rate,
$\varepsilon = 1$ be an exploration parameters.
For each $n \in \overline{1,N}$, do

(Policy improvement) Select «elite» trajectories $\mathcal{T}_n = \{\tau_k, k \in \overline{1,K} \colon G(\tau_k) > \gamma_q\}$, where γ_q is a q-quantile of the numbers $G(\tau_k)$, $k \in \overline{1,K}$. Define

$$Loss(\theta) = -\frac{1}{|\mathcal{T}_n|} \sum_{(a|s) \text{«}\in\text{»} \mathcal{T}_n} \ln \pi^\theta(a|s)$$

and update parameters by the (stochastic) gradient descent:

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta Loss(\theta_n)$$

Reduce ε ($\varepsilon = 1/N$).