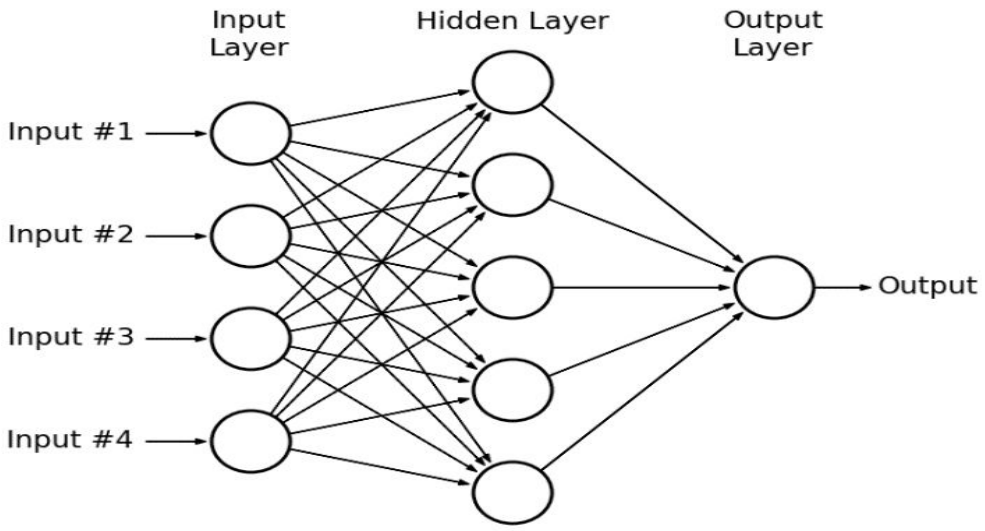
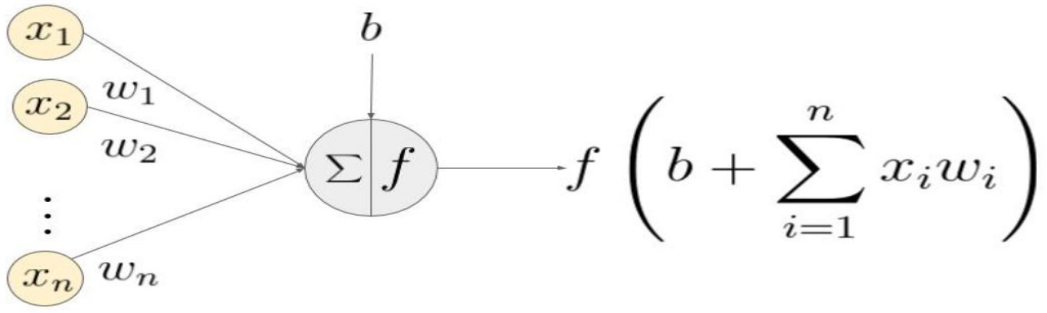
Multilayer Perceptron Classifier

Multilayer Perceptron Classifier - архітектура нейронної мережі (модель бінарного класифікатора в глибинному навчанні). Складається з шарів нейронів, які послідовно передають та редагують інформацію.



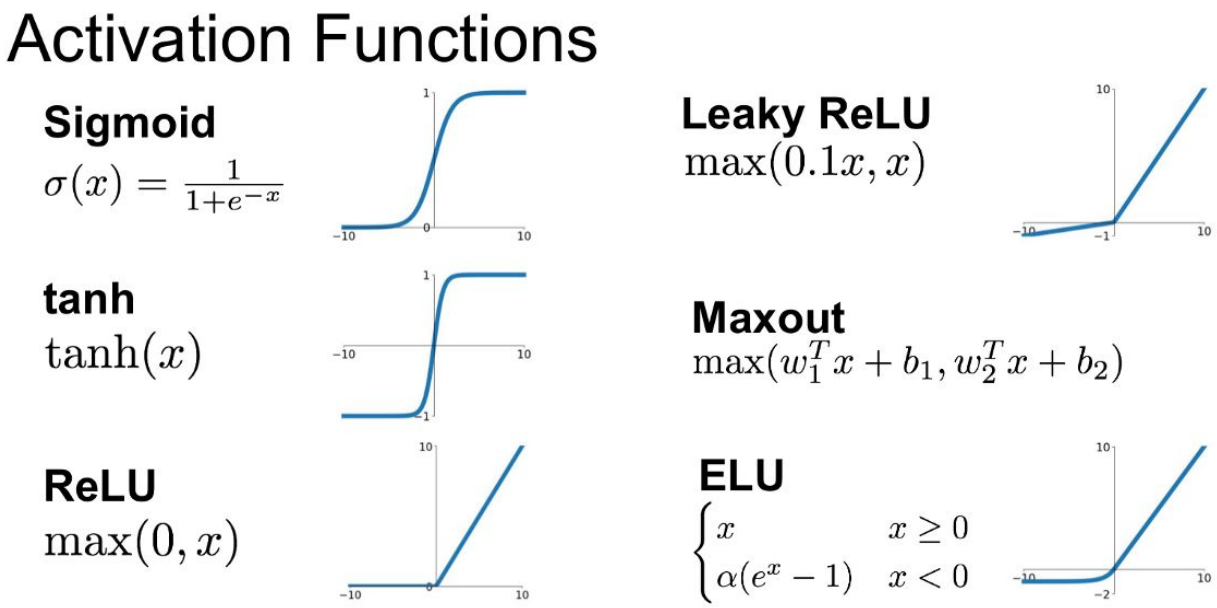
Кожен нейрон приймає інформацію (дійсне число) від усіх нейронів попереднього шару та обчислює лінійну комбінацію таких величин (а також додає незалежний параметр). Коефіцієнти комбінації - параметри моделі, які потрібно оптимізувати в процесі навчання моделі.



An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.

Activation Functions and Loss Function

Нейрон застосовує активуючу функцію до лінійного виходу для нелінійного виходу (щоб моделювати складні, нелінійні залежності). Найчастіше використовують наступні активуючі функції:



Далі нейрон передає вихідний результат до наступного шару.

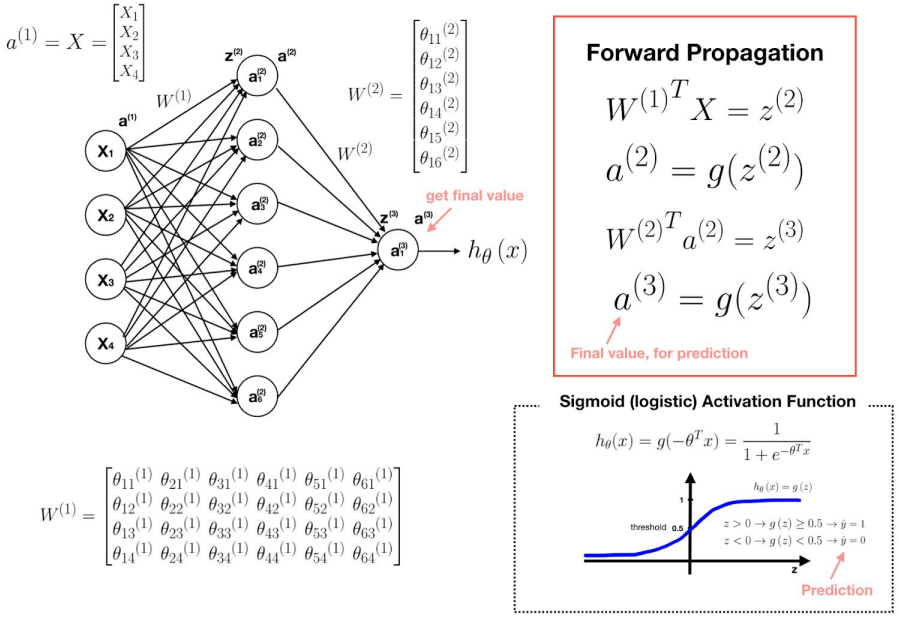
На виході мережі є одне дійсне число (вектор, якщо декілька тренувальних прикладів), для якого обчислюється функція втрати (з урахуванням актуальних значень відповіді):

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

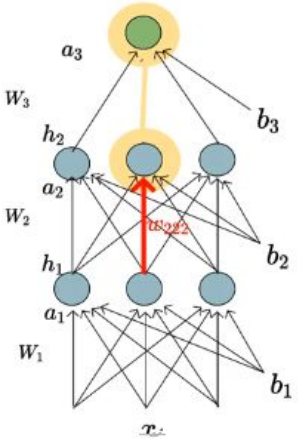
Binary Cross-Entropy / Log Loss

Forward and Backward Propagation

Обчислення всіх виходів нейронів та кінцевого виходу мережі за допомогою параметрів (ваг) та вхідних даних називають *Forward Propagation*:



Потім обчислюють часткові похідні loss function по всіх параметрах (вагах) системи. Тут використовують calculus chain rule, ця фаза тренування називається *Backward Propagation*:



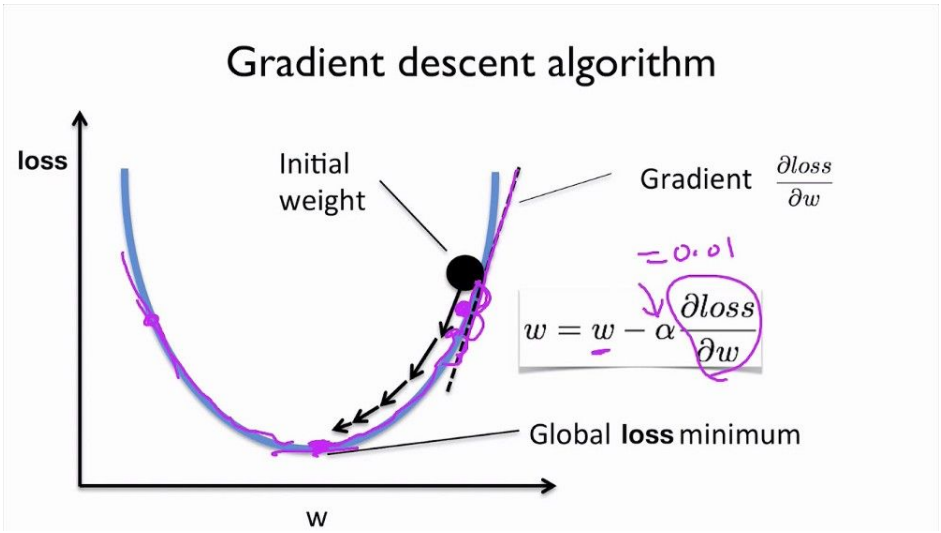
- Let us focus on the highlighted weight (w_{222})
- To learn this weight, we have to compute partial derivative w.r.t loss function

$$(w_{222})_{t+1} = (w_{222})_t - \eta * (\frac{\partial L}{\partial w_{222}})$$

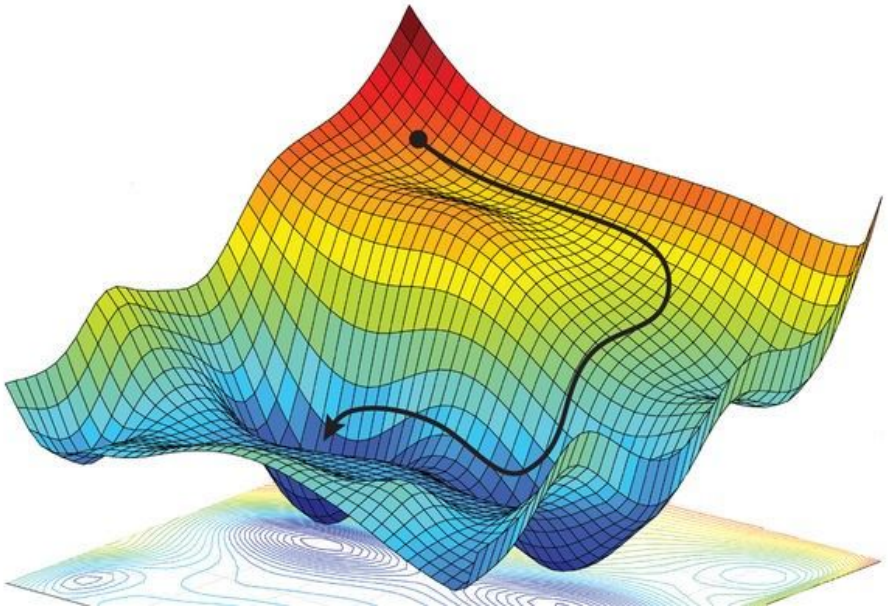
$$\begin{aligned} \frac{\partial L}{\partial w_{222}} &= (\frac{\partial L}{\partial a_{22}}).(\frac{\partial a_{22}}{\partial w_{222}}) \\ &= (\frac{\partial L}{\partial h_{22}}).(\frac{\partial h_{22}}{\partial a_{22}}).(\frac{\partial a_{22}}{\partial w_{222}}) \\ &= (\frac{\partial L}{\partial a_{31}}).(\frac{\partial a_{31}}{\partial h_{22}}).(\frac{\partial h_{22}}{\partial a_{22}}).(\frac{\partial a_{22}}{\partial w_{222}}) \\ &= (\frac{\partial L}{\partial \hat{y}}).(\frac{\partial \hat{y}}{\partial a_{31}}).(\frac{\partial a_{31}}{\partial h_{22}}).(\frac{\partial h_{22}}{\partial a_{22}}).(\frac{\partial a_{22}}{\partial w_{222}}) \end{aligned}$$

Gradient Descent Algorithm

Для тренування моделі потрібно мінімізувати loss function - знайти її глобальний мінімум. Ця функція (cross-entropy loss) є функцією параметрів (ваг) моделі. Для оптимізації використовується ітеративний алгоритм пошуку мінімуму *gradient descend*:



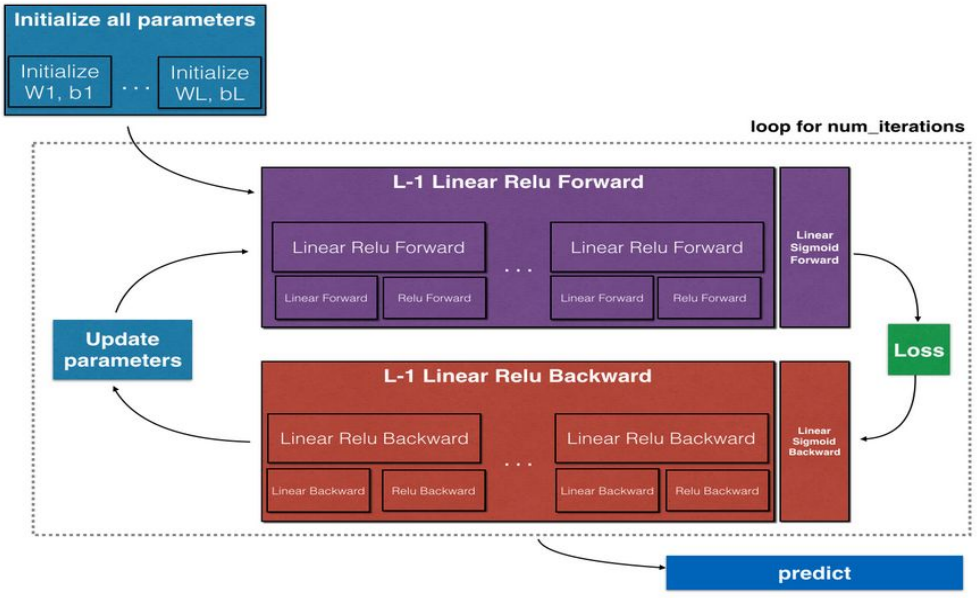
Принцип дії - параметри системи ітеративно змінюються, рухаючись на кожному кроці в протилежний бік до градієнту функції. Швидкість руху задається параметром learning rate:

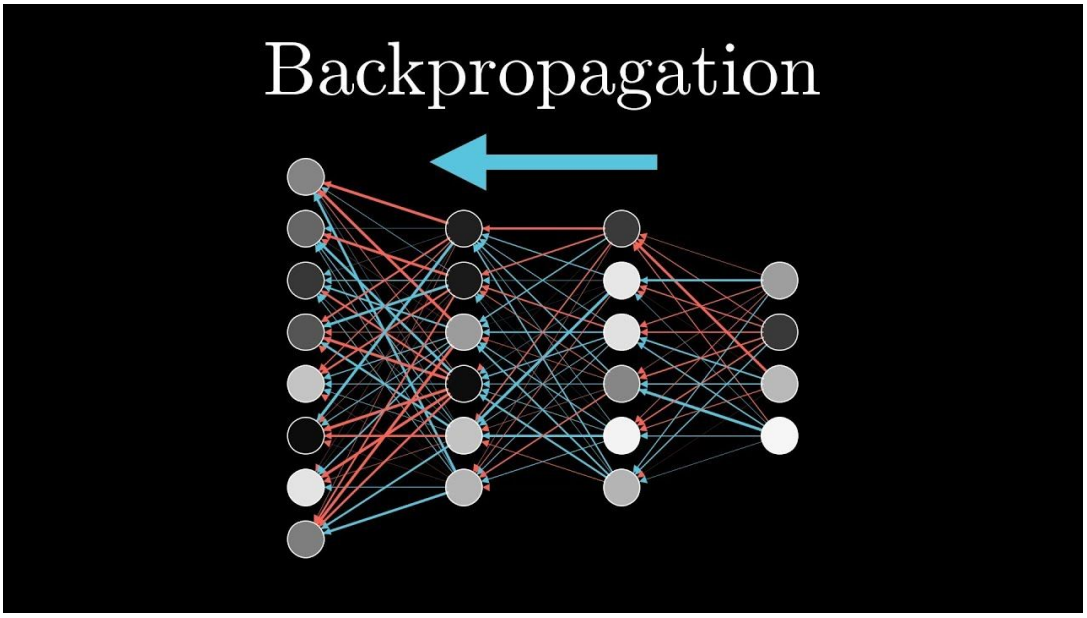


Full cycle of Neural Network training

Параметри оновлюються фіксовану кількість раз (при спуску до мінімуму) і тоді модель вважають натренованою (параметри визначені та оптимізовані).

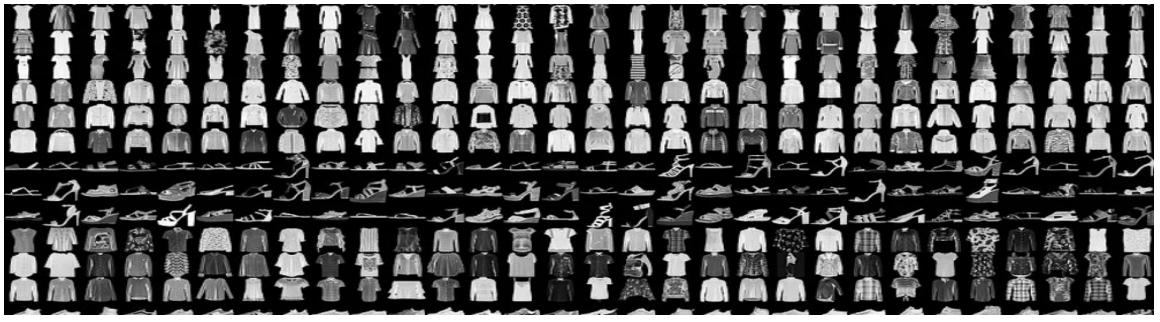
Для класифікації, натренована модель приймає вхідні дані і за допомогою Forward Propagation обчислює бінарний вихід.





Training and Practical Application

Я створив Python клас для моделей такої архітектури довільного розміру і форми. Створив модель з двома внутрішніми шарами та натренував її на датасеті MNIST fashion (тільки футболки і штани для бінарної класифікації). Цей датасет містить 12000 залейблених прикладів фотографій штанів чи футболок (всього 60000 фотографій 10 типів одягу). Я досягнув точності 97% на тестових даних (до них модель не мала доступ при тренуванні).



Для перевірки роботоздатності нейронної мережі я завантажив фотографію своїх штанів та своєї сорочки (попередньо зробивши фон чорним, а одяг білими):

Нейронна мережа коректно передбачила вид одягу, що можна вважати чудовим результатом для такої простої і швидкої мережі. Для більш поглибленого аналізу зображень, відео використовують Convolutional Neural Networks в Computer Vision