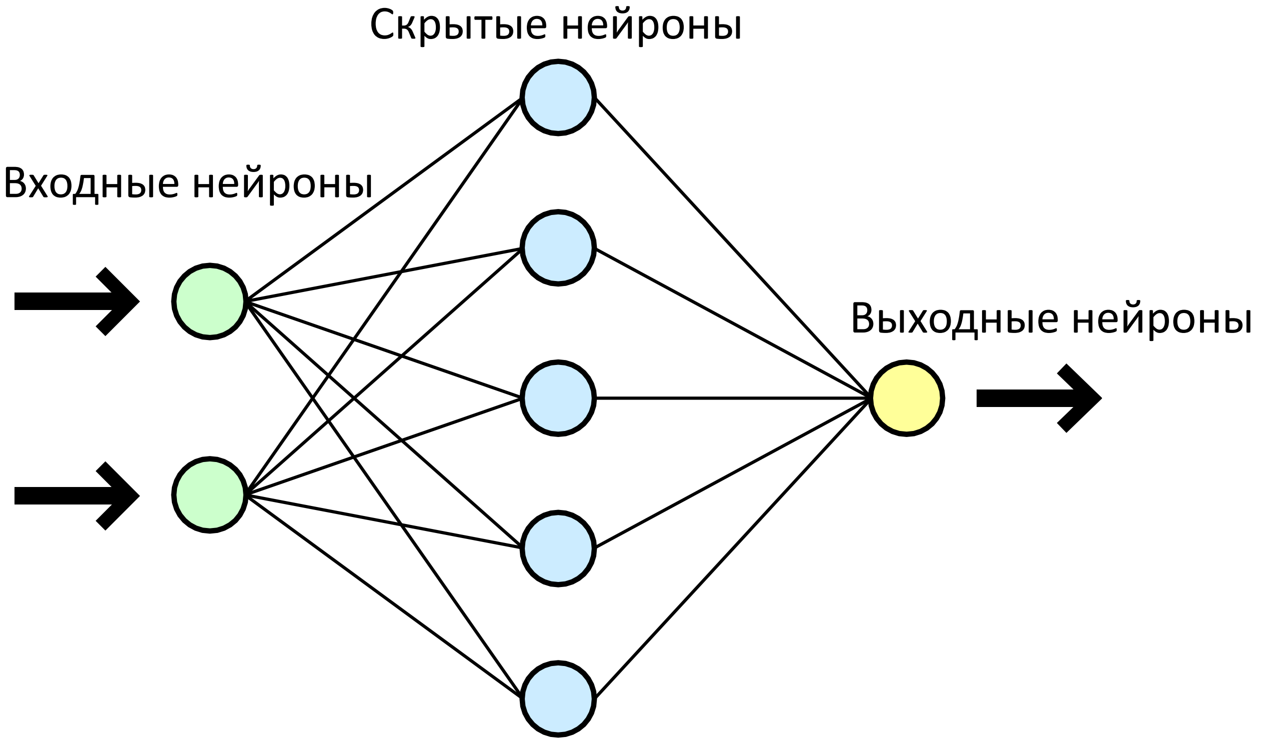
Введение в нейронные сети

Охрименко Дмитрий, БПМИ-172

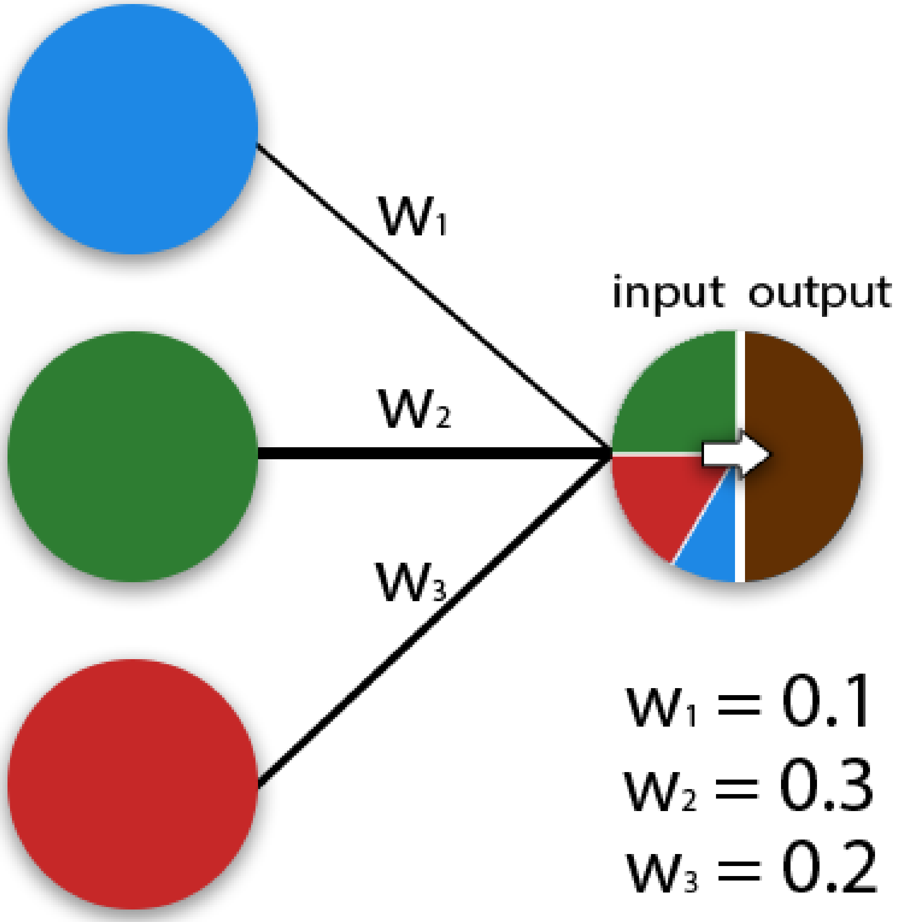
Нейронные сети. Определение

- Последовательность нейронов, соединенных между собой синапсами
- Нейроны получают информацию, производят вычисления и передают дальше
- Два основных параметра – unput и output data
- Если нейронов достаточно много, вводится термин слоя



Синапс

- Связь между нейронам
- Имеет параметр веса, влияет на итоговый результат

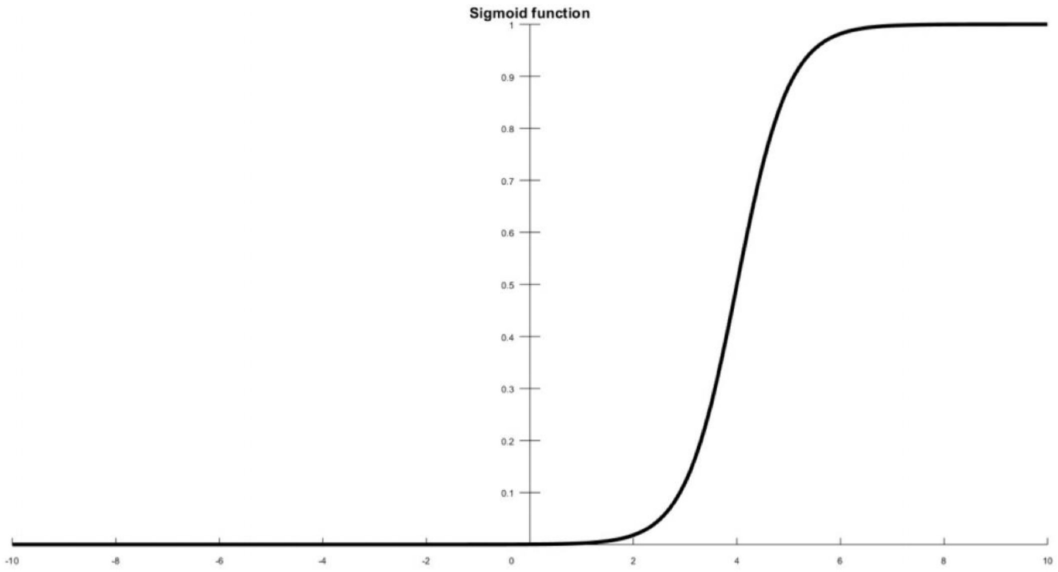


Функция активации

Способ нормализации входных данных

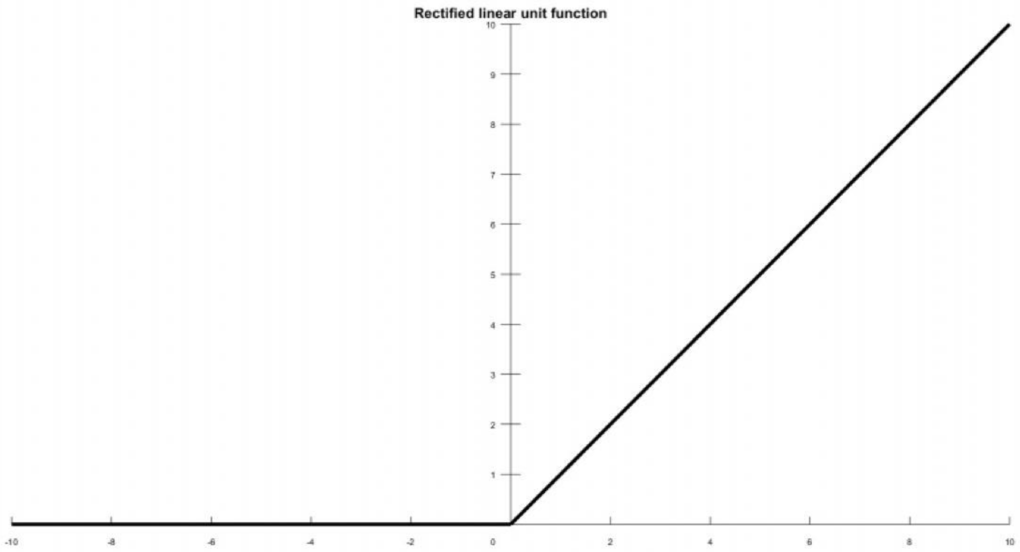
Сигмоид

$$f(x) = \frac{1}{1+\ e^{-x}}$$



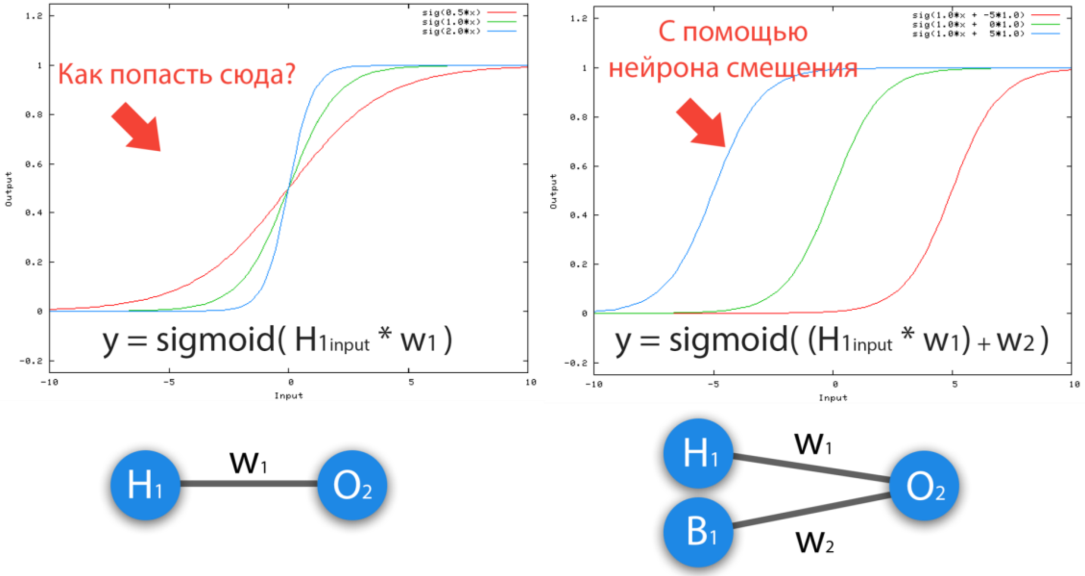
ReLu

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$



Нейрон смещения

- Не имеет входных синапсов, вход = выход = 1
- С помощью весов можно менять только наклон функции
- Для движения вправо и влево помогает нейрон смещения



Применение нейронных сетей

- Распознавание образов и классификация
- Принятие решений и управление
- Кластеризация
- Прогнозирование
- Апроксимизация
- Сжатие данных и ассоциативная память
- Анализ данных
- Оптимизация

Классификация

- Классификация по типу входной информации
  - **Аналоговые нейронные сети** – работают с действительными числами
  - **Двоичные нейронные сети** – работают с двоичными числами
  - **Образные нейронные сети** – работают с образами – знаками, символами и тд
- Классификация по характеру обучения
  - **Обучение с учителем** - выходное пространство решений нейронной сети известно
  - **Обучение без учителя** - нейронная сеть формирует выходное пространство решений только на основе входных воздействий
  - **Обучение с подкреплением** - система назначения штрафов и поощрений от среды
- Классификация по характеру настройки синапсов
  - **Сети с фиксированными связями** – веса выбираются сразу
  - **Сети с динамическами связями** – веса настраиваются во время обучения

Классификация

- Классификация по характеру связи
  - **Сети прямого распространения**
    - Все связи направлены строго от входных нейронов к выходным
  - **Рекуррентные нейронные сети**
    - Допускаются обратные связи от входных и скрытых нейронов на входные нейроны
  - **Радиально-базисные функции**
    - Используют в качестве активирующей функции радиально-базисные функции:

$$f(x) = \phi\left(\frac{x^2}{\sigma^2}\right), \text{ например, } f(x) = e^{-\frac{x^2}{\sigma^2}}$$

    - Единственный скрытый слой;
    - Только нейроны скрытого слоя имеют нелинейную активационную функцию;
    - Синаптические веса связей входного и скрытого слоев равны единице.

Этапы решения задачи

- Сбор данных для обучения;
- Подготовка и нормализация данных;
- Выбор топологии сети;
- Экспериментальный подбор характеристик сети;
- Экспериментальный подбор параметров обучения;
- Собственно обучение;
- Проверка адекватности обучения;
- Корректировка параметров, окончательное обучение;
- Вербализация сети с целью дальнейшего использования.

Deep learning

- Используется каскад из множества обрабатывающих слоев
- Основывается на изучении признаков (представлении информации) в данных без обучения с учителем. Функции более высокого уровня (которые находятся в последних слоях) получаются из функций нижнего уровня (которые находятся в слоях начальных слоях);
- Изучает многоуровневые представления, которые соответствуют разным уровням абстракции; уровни образуют иерархию представления.

BackPropagation

Метод вычисления градиента для обновления весов.

Идея - распространять сигналы ошибки от выходов сети к её входам

Функции активации:

-экспоненциальная сигмоида

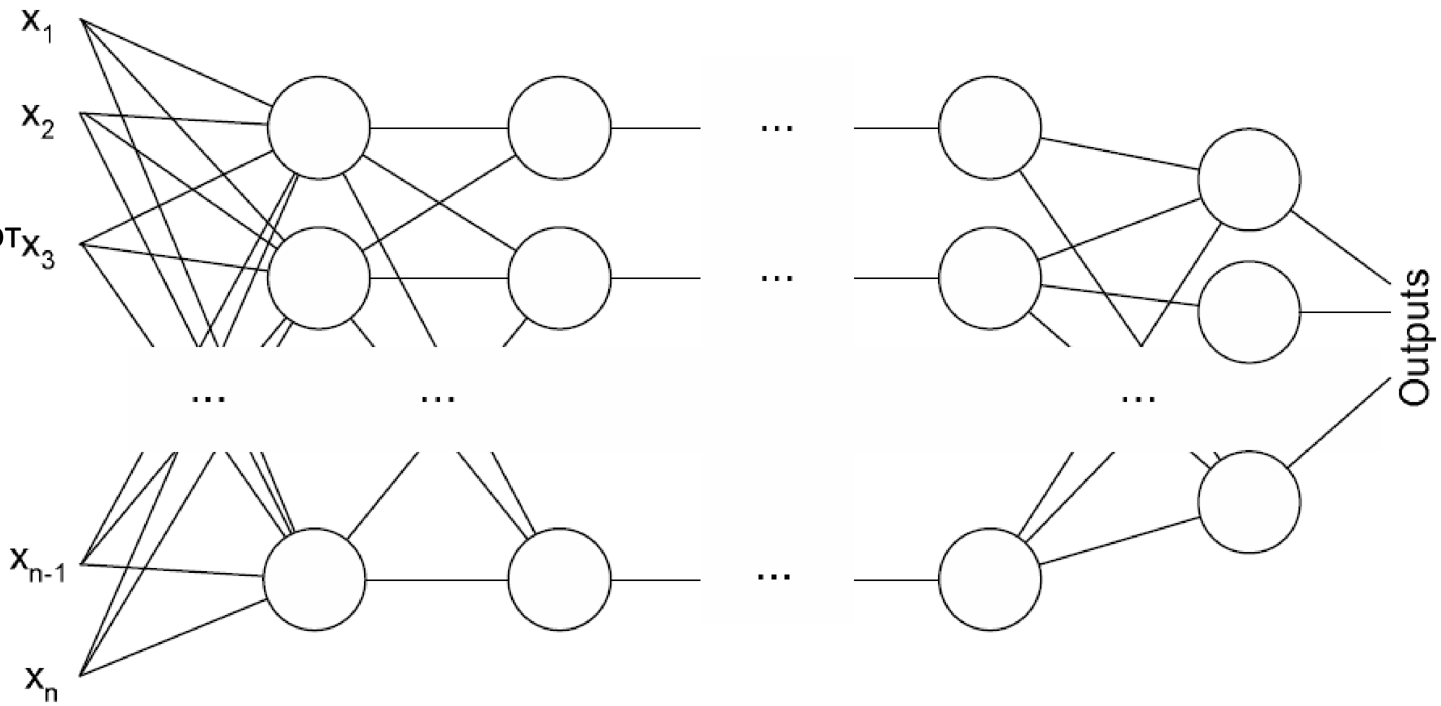
$$f(s) = \frac{1}{1 + e^{-2\alpha s}}$$

- рациональная сигмоида

$$f(s) = \frac{s}{|s| + \alpha}$$

- гиперболический тангенс

$$f(s) = \operatorname{th} \frac{s}{\alpha} = \frac{e^{\frac{s}{\alpha}} - e^{-\frac{s}{\alpha}}}{e^{\frac{s}{\alpha}} + e^{-\frac{s}{\alpha}}}$$



BackPropagation

- Функция ошибки – $E(\{w_{i,j}\}) = \frac{1}{2} \sum_{k \in \text{Outputs}} (t_k - o_k)^2$
- Для модификации весов реализуем стохастический градиентный спуск. Тогда к каждому весу будем добавлять $\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}}$
- Производная $\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial S_j} \frac{\partial S_j}{\partial w_{i,j}} = x_i \frac{\partial E}{\partial S_j}$, где j – нейроны последнего уровня
- Sj влияет на ошибку только в рамках j-го узла oj, тогда получаем:

$$\frac{\partial E}{\partial S_j} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial S_j} = \left(\frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in \text{Outputs}} (t_k - o_k)^2 \right) \left(\frac{\partial \, \mathrm{f}(S)}{\partial S} |_{S=S_j} \right) =$$

$$= \left(\frac{1}{2} \frac{\partial}{\partial o_j} (t_j - o_j)^2 \right) (o_j(1 - o_j)) 2\alpha = -2\alpha o_j (1 - o_j)(t_j - o_j).$$

Где f(S)– выбранная нами сигмоида

BackPropagation

$$\frac{\partial(\frac{1}{1+e^{-2\alpha S}})}{\partial S} = \frac{-1}{(1+e^{-2\alpha S})^2} \times \frac{\partial(1+e^{-2\alpha S})}{\partial S} = \frac{-1}{(1+e^{-2\alpha S})^2} \times (-2\alpha e^{-2\alpha S}) =$$
$$= \frac{2\alpha e^{-2\alpha S}}{(1+e^{-2\alpha S})^2} = \left(\frac{1+e^{-2\alpha S}}{(1+e^{-2\alpha S})^2} - \frac{1}{(1+e^{-2\alpha S})^2}\right) \times 2\alpha = 2\alpha(\mathrm{f}(S) - \mathrm{f}^2(S))$$

- Если же j-й не на последнем уровне, то у него есть выходы Children. Тогда: $\frac{\partial E}{\partial S_j} = \sum_{k \in \mathrm{Children}(j)} \frac{\partial E}{\partial S_k} \frac{\partial S_k}{\partial S_j}$ и $\frac{\partial S_k}{\partial S_j} = \frac{\partial S_k}{\partial o_j} \frac{\partial o_j}{\partial S_j} = w_{j,k} \frac{\partial o_j}{\partial S_j} = 2\alpha w_{j,k} o_j (1 - o_j)$
- Мы научились вычислять поправку для узлов любого уровня, теперь напишем алгоритм

BackPropagation - алгоритм

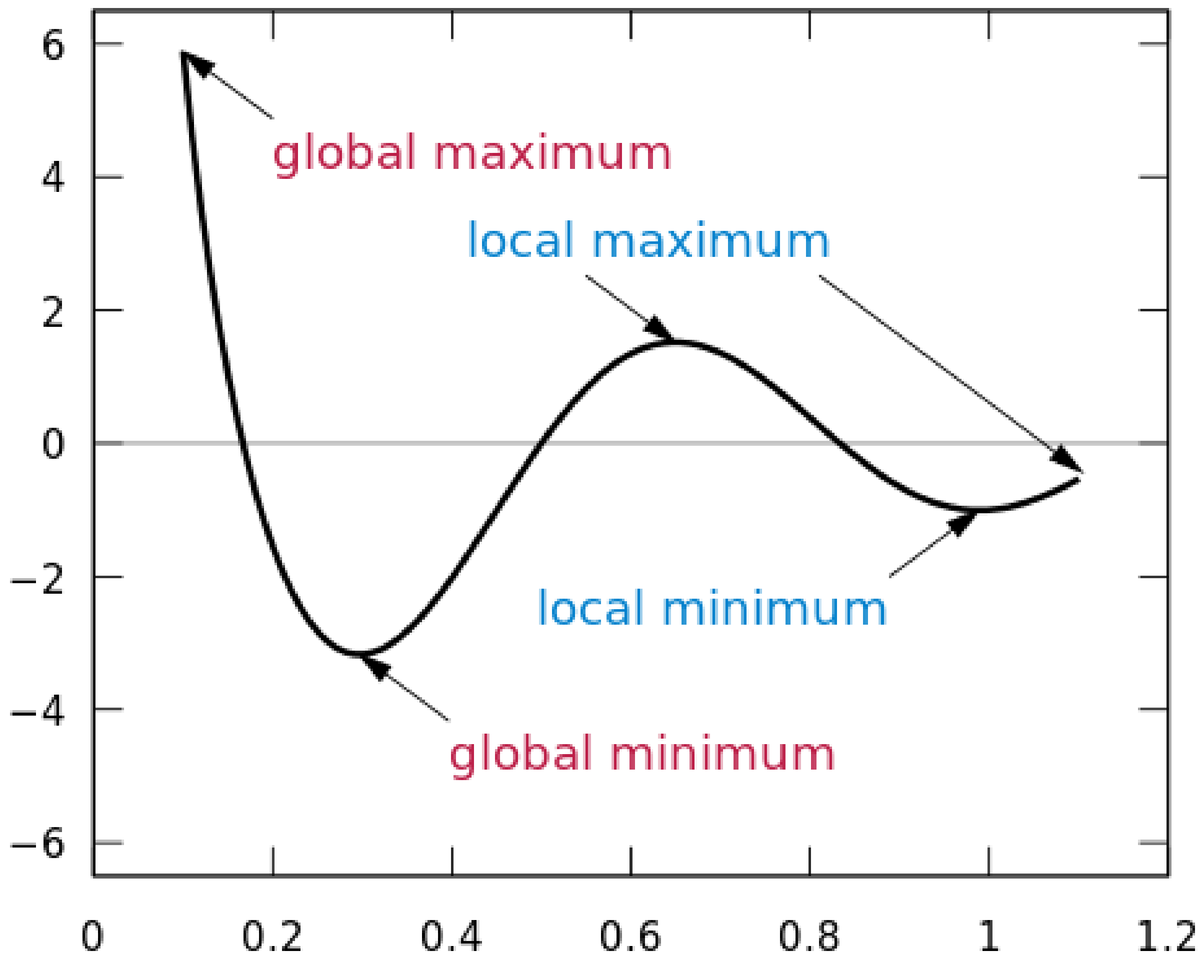
1. Инициализируем веса маленькими случайными значениями
2. Повторить n раз:
   - Для всех d от 1 до m:
   - Подать на вход {x_i^d} на вход сети и найти o_i каждого узла
   - Для всех k из последнего уровня $\delta_k = o_k(1 - o_k)(t_k - o_k)$
   - Для всех уровней l, начиная с предпоследнего
     - Для каждого узла j уровня l: $\delta_j = o_j(1 - o_j) \sum_{k \in Children(j)} \delta_k w_{j,k}$
   - Для каждого ребра сети {i,j}:

$$\Delta w_{i,j}(n) = \alpha \Delta w_{i,j}(n-1) + (1-\alpha)\eta\delta_j o_i$$
$$w_{i,j}(n) = w_{i,j}(n-1) + \Delta w_{i,j}(n).$$

3. Выдать веса w_ij

BackPropagation - недостатки

- Паралич сети
- Локальные минимумы
- Размер шага

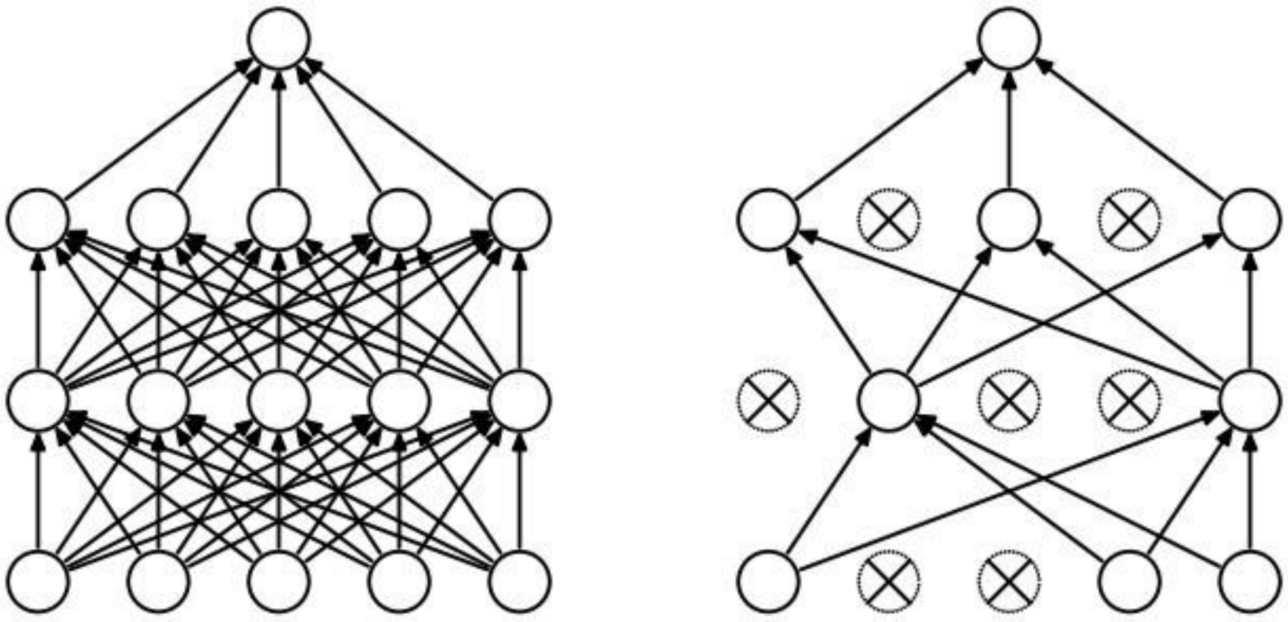


Dropout

Способ борьбы с переобучением

Выбрасывается 1 или несколько случайных нейронов

При прямом дропауте выбрасывается на стадии тестирования, при обратном – на стадии обучения



(a) Standard Neural Net (b) After applying dropout.

Прямой dropout

- Выключает Нейроны с вероятностью p, включает – 1-p
- h(x)=xW+b — линейная проекция входного di-мерного вектора x на dh-мерное пространство выходных значений;
- a(h) – функция активации, тогда применение дропаут – это измененная функция активации: f(h) = D * a(h), D – вектор случайных величин, распределенных по закону Бернулли.
- Применение Дропаут к i-му нейрону: $O_i = X_i a(\sum_{k=1}^{d_i} w_k x_k + b) = \begin{cases} a(\sum_{k=1}^{d_i} w_k x_k + b), & \text{if} \quad X_i = 1 \\ 0, & \text{if} \quad X_i = 0 \end{cases},$
- Таким образом, на этапе обучения: $O_i = X_i a(\sum_{k=1}^{d_i} w_k x_k + b)$
- На этапе тестирования: $O_i = qa(\sum_{k=1}^{d_i} w_k x_k + b)$

Обратный dropout

- Работает на стадии обучения, коэффициент равен 1/(1-p)
- Тогда на этапе обучения: $O_i = \frac{1}{q} X_i a(\sum_{k=1}^{d_i} w_k x_k + b)$
- На этапе тестирования: $O_i = a(\sum_{k=1}^{d_i} w_k x_k + b)$
- Применяется чаще, чем прямой, так как не требует изменять нейронную сеть

Dropout множества нейронов

- Рассмотрим слой h из n нейронов на отдельном шаге этапа обучения как ансамбль из n экспериментов Бернулли с вероятностью успеха p.
- Таким образом, количество исключенных нейронов $Y = \sum_{i=1}^{d_h}(1 - X_i)$
- Тогда общее число нейронов - биноминальная величина $Y \sim Bi(d_h, p)$
- K успешных событий за n попыток: $f(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k}$

L2-регуляризация

- Еще один способ борьбы с переобучением
- Накладывает штрафы на веса с наибольшими значениями, минимизируя их L2 норму
- Для каждого веса добавляем к целевой функции $\mathcal{L}(\vec{\hat{y}}, \vec{y})$ слагаемое $\frac{\lambda}{2}||\vec{w}||^2 = \frac{\lambda}{2}\sum_{i=1}^{W} w_i^2$
- Важно правильно выбрать лямбду. Если коэффициент слишком мал, то эффект от регуляризации будет ничтожен, если же слишком велик — модель обнулит все веса.

Инициализация сети

- Выбор начальных данных играет большую роль на успешность обучения
- Если установить все 0, то все веса будут неактивны, что послужит серьезным испытанием для обучения
- Рассмотрим два способа инициализации – методы Завьера и Ге

Метод Завьера (Xavier)

- Идея – упростить прохождение сигнала через слой во время как прямого, так и обратного распространения ошибки для линейной функции активации
- Подходит для сигмоидной функции активации
- Она вероятностное распределение (равномерное или нормальное) с дисперсией = $\mathrm{Var}(W) = \dfrac{2}{n_{in} + n_{out}}$

Метод Ге (He)

- Больше подходит для ReLU, так как компенсирует тот факт, что эта функция возвращает нуль для половины области определения
- Все то же самое, но дисперсия равна $\mathrm{Var}(W) = \dfrac{2}{n_{in}}$

Получение дисперсии

- Рассмотрим, что происходит с дисперсией выходных значений линейного нейрона:

$$\mathrm{Var}(\sum_{i=1}^{n_{in}} w_i x_i) = \sum_{i=1}^{n_{in}} \mathrm{Var}(w_i x_i) = \sum_{i=1}^{n_{in}} \mathrm{Var}(W)\mathrm{Var}(X) = n_{in}\mathrm{Var}(W)\mathrm{Var}(X)$$

- Чтобы сохранить дисперсию входных данных после прохождения через слой, надо, чтобы $\mathrm{Var}(W) = \frac{1}{n_{in}}$
- Аналогичная логика для out
- Тогда в качестве ответа берем их среднее $\mathrm{Var}(W) = \frac{2}{n_{in} + n_{out}}$

Batch normalization

- Проблема: по мере распространения сигнала по сети он может сильно исказиться как по матожиднию, так и по дисперсии, что чревато несоответствиями градиента на разных уровнях
- Решение: перед каждый слоем нормализовать входные данные таким образом, чтобы получить нулевое матожидание и единичную дисперсию.
- Имеем $\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i$ и $\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$, тогда $\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$
- Окончательная функция активации $y_i = \gamma \hat{x}_i + \beta$

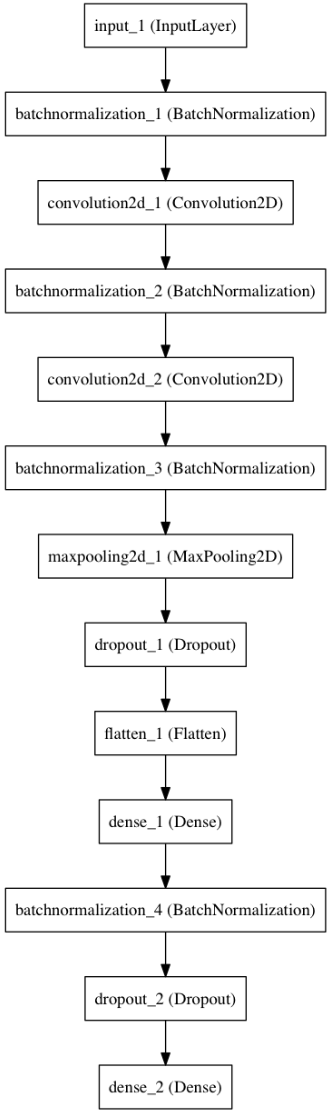
Расширение обучающего множества

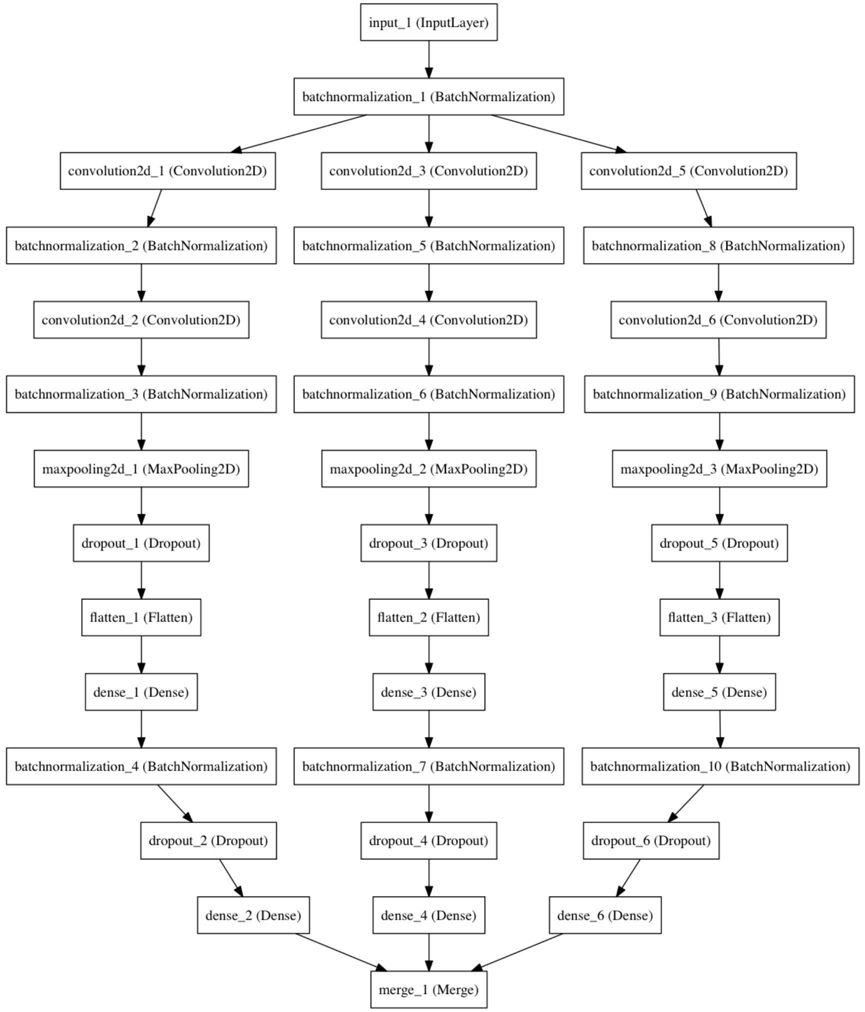
- Проблема: модель обучилась на идеальных символах, а получив слегка искаженные, потеряла в эффективности
- Решение: слегка исказить данные и обучать на большем количестве примеров

Ансамбли

Проблема: при различных начальных условиях обучения им легче дается распределение по одним классам, в то время как другие приводят в замешательство

Решение: вместо одной сети построить несколько копий с разными начальными значениями и вычислить их средний результат на одних и тех же входных данных.





Ранняя остановка

- Сеть прогоняется с различными комбинациями гиперпараметров, а затем решение принимается на основе их производительности на валидационном множестве
- Нельзя использовать тестовое множество до того, как мы определимся с гиперпараметрами
- Ранняя остановка – останавливаем подбор параметров, если за заданное количество эпох потери не начинают уменьшаться

Вопросы

- \- В чем разница между методами инициализации Завьера и Ге? Какая функция активации хорошо подходит каждому методу?
- \- Напишите формулу функции активации после применения к ней батчнорма
- \- Какая разница между прямым и обратным дропаутом? Какой из них предпочтительнее?

Источники

- https://ru.wikipedia.org/wiki/Искусственная_нейронная_сеть
- https://ru.wikipedia.org/wiki/Метод_обратного_распространения_ошибки
- https://habr.com/ru/company/wunderfund/blog/330814/
- https://habr.com/ru/company/wunderfund/blog/315476/