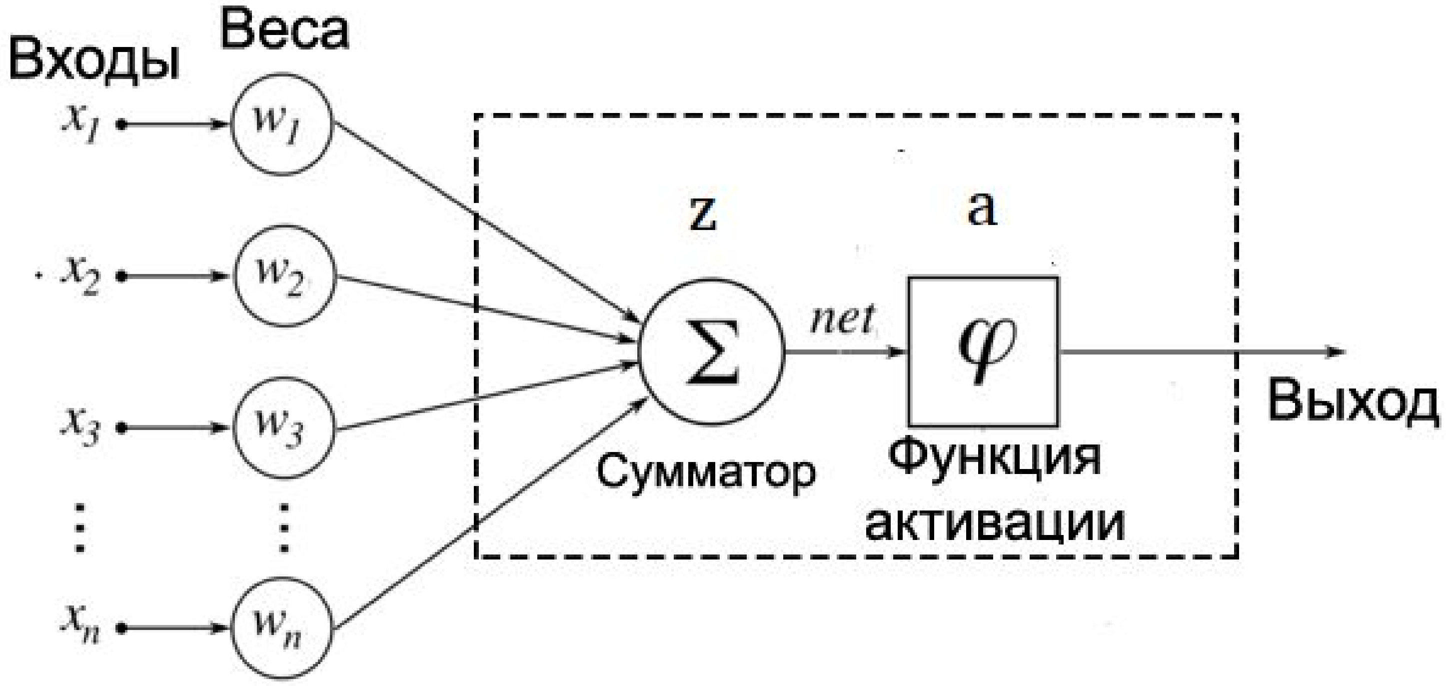
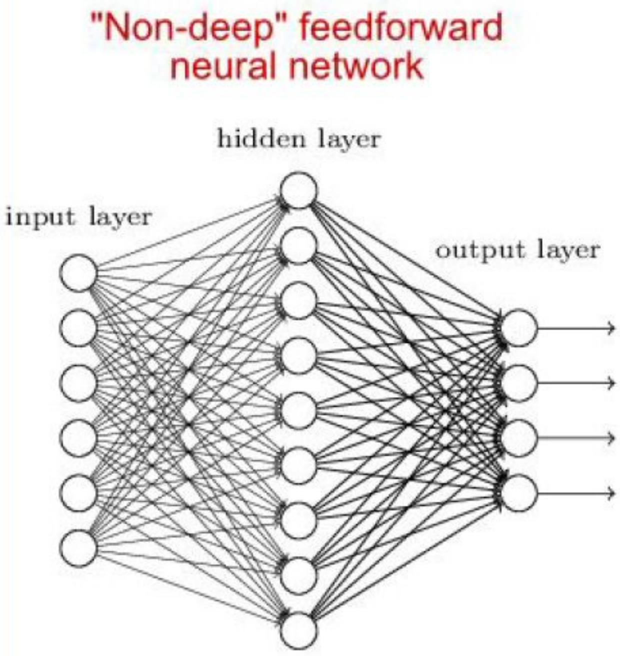
Введение в нейронные сети

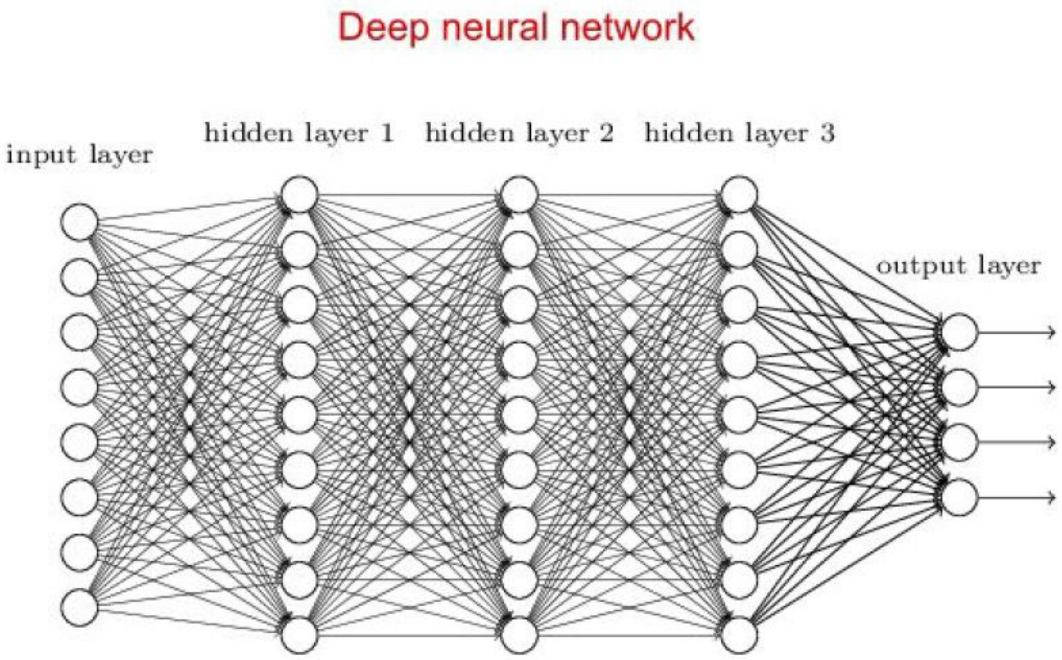
Полносвязные нс

Искусственный нейрон



Нейросеть


"Non-deep" feedforward
neural network
hidden layer
input layer
output layer


Deep neural network
input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer

Feed forward

y[n, i] - активация i-го нейрона уровня n

$$y_n^i = \sum_k w_n^i k$$

w - матрица весов

$$x_n^i = F(y_n^i)$$

Векторизованно:

$$Y_n = W_n X_{n-1}$$
$$X_n = F(Y_n)$$

Backpropagation: интуиция

Будем опираться на $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$

Рассмотрим функцию $f(x, y, z) = (x + y)z \rightarrow q = x + y; f = q \cdot z.$

Мы могли сразу посчитать все производные но это слишком просто для нас.

$$\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

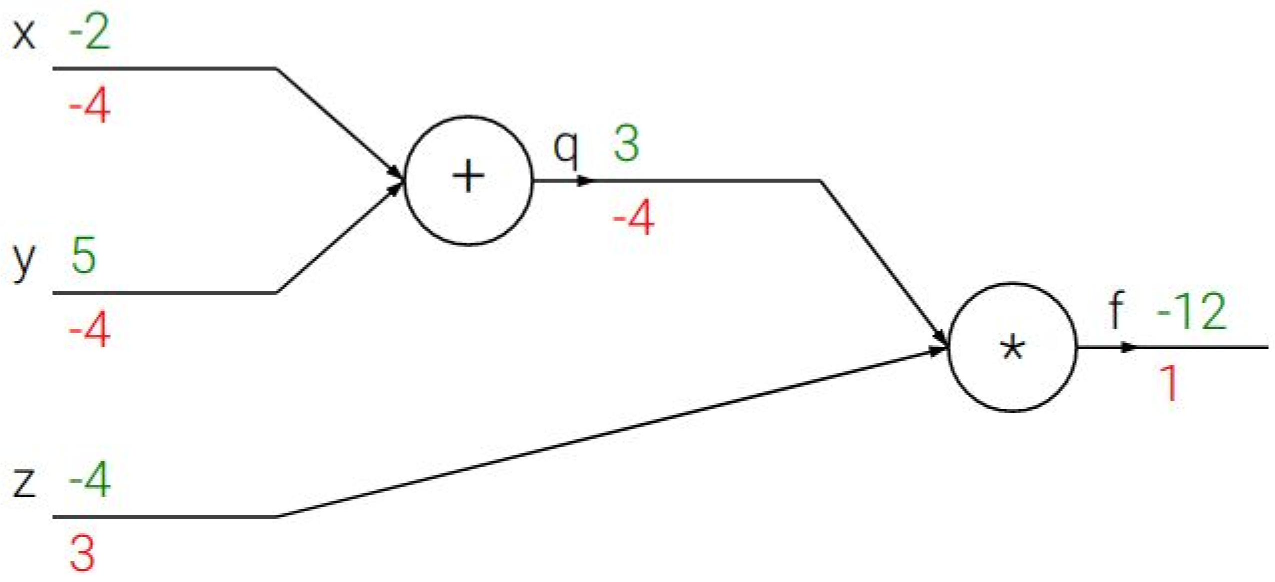
У нас градиент $(1 \cdot z, 1 \cdot z, x + y)$!

(x + y) z

Пусть на входе (-2, 5, -4)

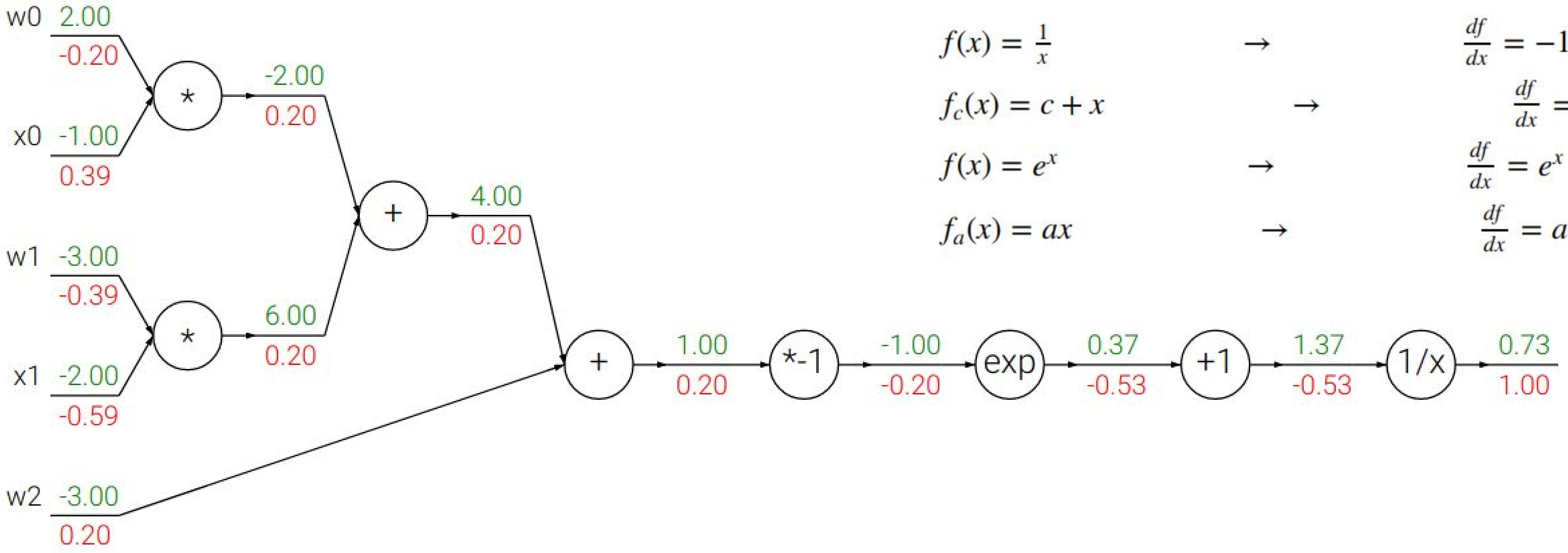
$$\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$



Пример 2

Пусть теперь у нас функция $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

Backpropagation

Мы хотим получить что-то похожее на градиентный спуск, только для всей матрицы весов

$$W(t) = W(t-1) - \eta \frac{\partial E}{\partial W}$$

где E - стоимостная функция, W - вектор весов

Будем использовать дифференцирование сложной функции по цепочке

$Y_n = W_n X_{n-1}$ где E^p - p-й вектор стоимости из обучающей выборки, f - активационная функция,

$X_n = F(Y_n)$ Y_n, X_n - значения в нейронах уровня n до и после активации, W - матрица весов

$$\frac{\partial E^p}{\partial y_n^i} = f'(y_n^i)\frac{\partial E^p}{\partial x_n^i}$$

$$\frac{\partial E^p}{\partial w_n^{ij}} = x_{n-1}^j \frac{\partial E^p}{\partial y_n^i}$$

$$\frac{\partial E^p}{\partial x_{n-1}^k} = \sum_i w_n^{ik}\frac{\partial E^p}{\partial y_n^i}.$$

$$\frac{\partial E^p}{\partial Y_n} = F'(Y_n)\frac{\partial E^p}{\partial X_n}$$

$$\frac{\partial E^p}{\partial W_n} = X_{n-1}\frac{\partial E^p}{\partial Y_n}$$

$$\frac{\partial E^p}{\partial X_{n-1}} = W_n^T\frac{\partial E^p}{\partial Y_n}.$$

Инициализация

- нулями - плохо
- небольшие случайные числа

MSE

Напомним, что мы такое(M - вывод нейронной сети на входе из Z)

$$E^p = \frac{1}{2}(D^p - M(Z^p, W))^2$$

Тогда для последнего уровня

$$\frac{dE^p}{dy_n^i} = f'(y_n^i)(D^p - M)$$

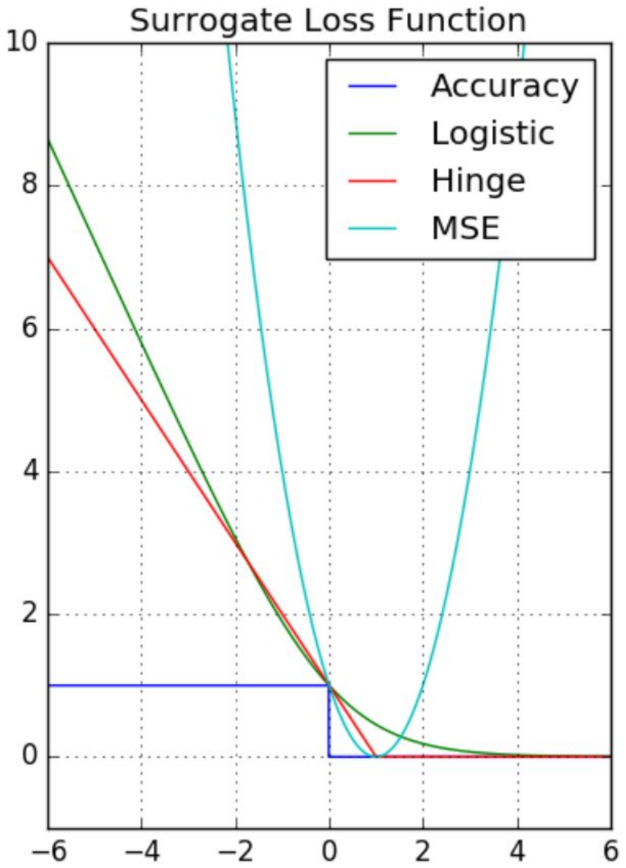
Задача классификации - logloss

Ответ пердставлен в виде вектора с 1 единицей. Надо предсказать вектор вероятностей принадлежности каждому классу. Как оценить?

$$L_i = \sum_j y_{ij} \log(\sigma(f_j)) + (1 - y_{ij}) \log(1 - \sigma(f_j))$$

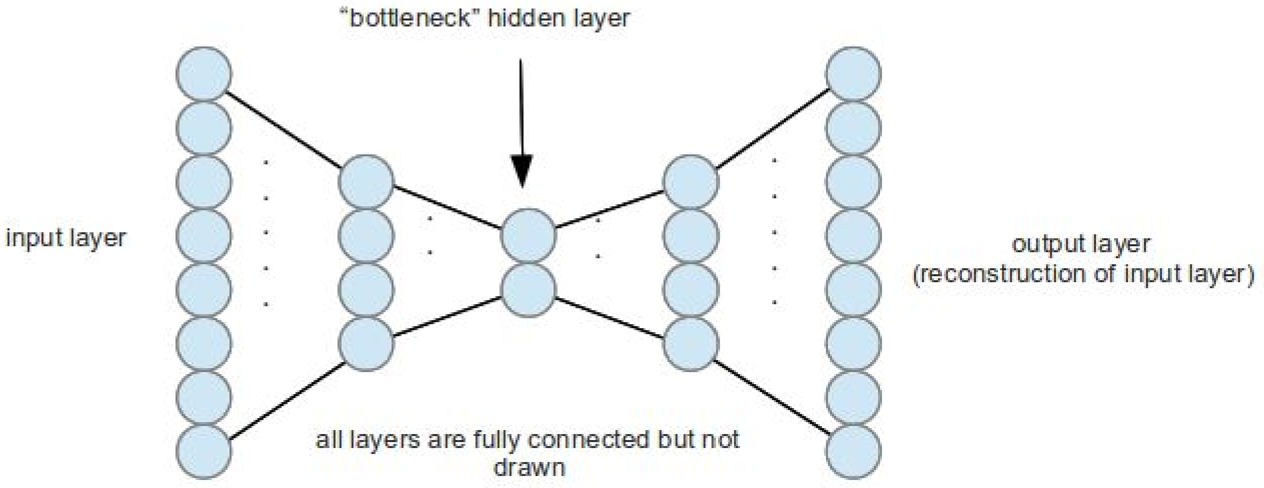
Градиент $\partial L_i / \partial f_j = y_{ij} - \sigma(f_j)$

Logloss vs MSE



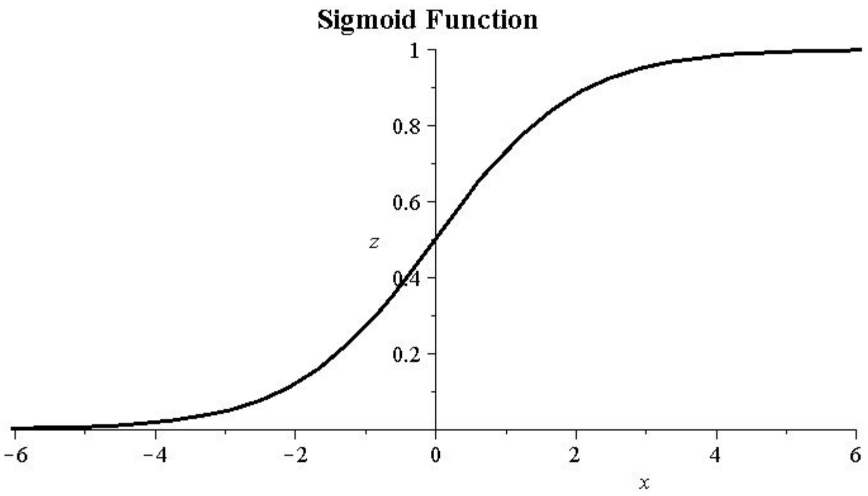
Autoencoder

- задача уменьшения размерности
- имеющееся решение(pca)
- autoencoder - нейросеть, на входе то что на выходе, средний слой очень малого размера



Сигмоида

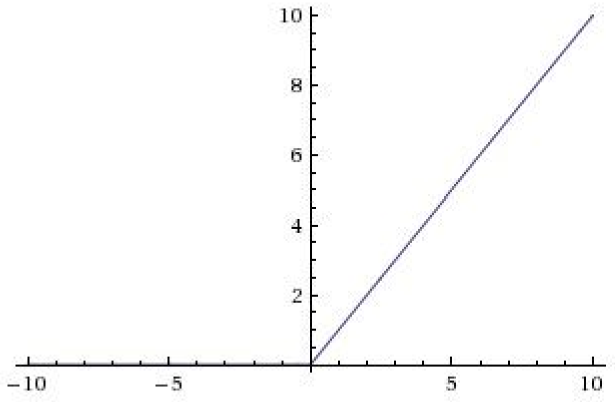
- хороша для получения вероятности
- вызывает затухание градиентов



Rectified Linear Unit

relu = max(0, x)

- проста в вычислении
- не вызывает затухания
- может приводить к необратимому занулению значения в нейроне
- не подходит для вероятности



Softmax

$$P(Y=k|X=x_i)=\frac{e^{s_k}}{\sum_j e^{s_j}} \qquad s=f(x_i;W)$$

$$L_i=-\log P(Y=y_i|X=x_i)$$

Максимизируем ОМП

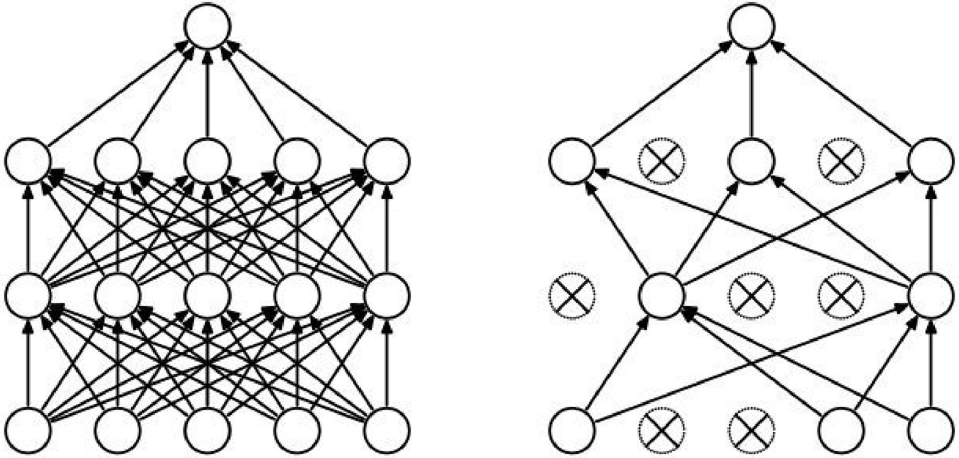
$$L_i=-\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

L2 Регуляризация

- обобщение линейного случая
- к ответу прибавляем $\frac{1}{2}\lambda w^2$
- к каждому градиенту прибавляется - λw

Dropout

- При обучении выкидываем из сети некоторые нейроны
- Нейрон пропускается с вероятностью p
- При оценивании домножаем все активации на p
- Можно делить активации на p при обучении



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

Список источников

http://cs231n.github.io/

https://ml-mipt.github.io/lec/2017/part1/ml-mipt-2017-dnn-lec1/ml-mipt-2017-dnn-lec1.pdf

https://habrahabr.ru/company/oleg-bunin/blog/340184/