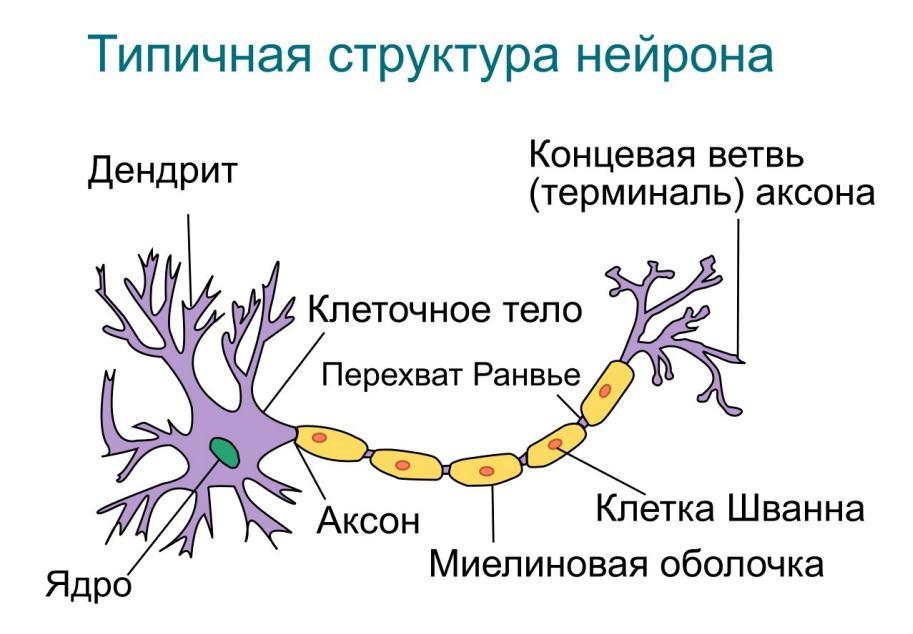
Нейронные сети

Нейронная сеть (Neural network)



Usefulness of Artificial Neural Networks in the Diagnosis and Treatment of Sleep Apnea-Hypopnea Syndrome
http://dx.doi.org/10.5772/66570

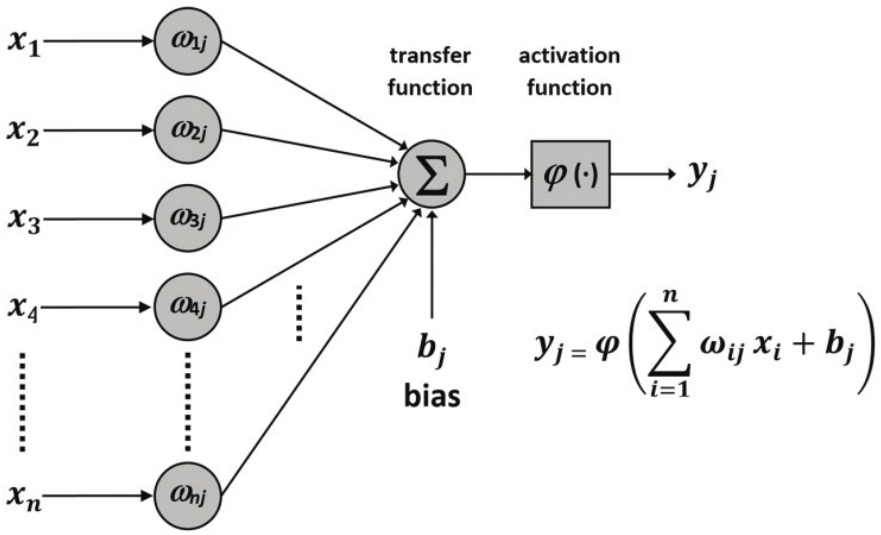
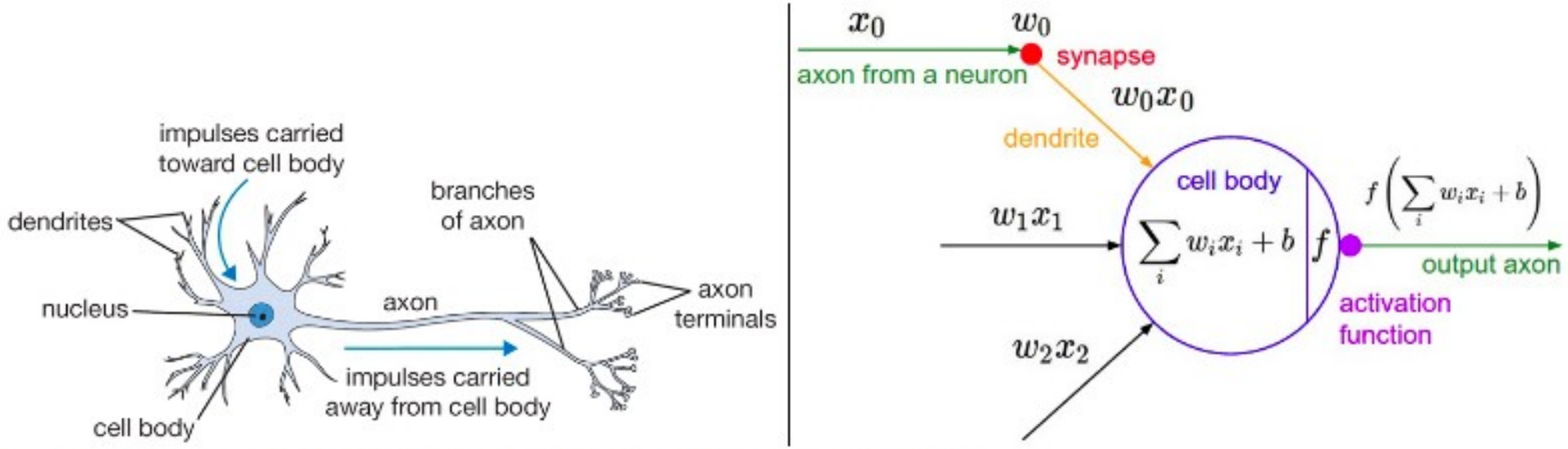


Figure 3. Scheme of a perceptron. A nonlinear activation function $\phi(\cdot)$ is applied to the weighted sum of the input features (x_n) and the bias term (b_j) in order to compute the output (y_k).

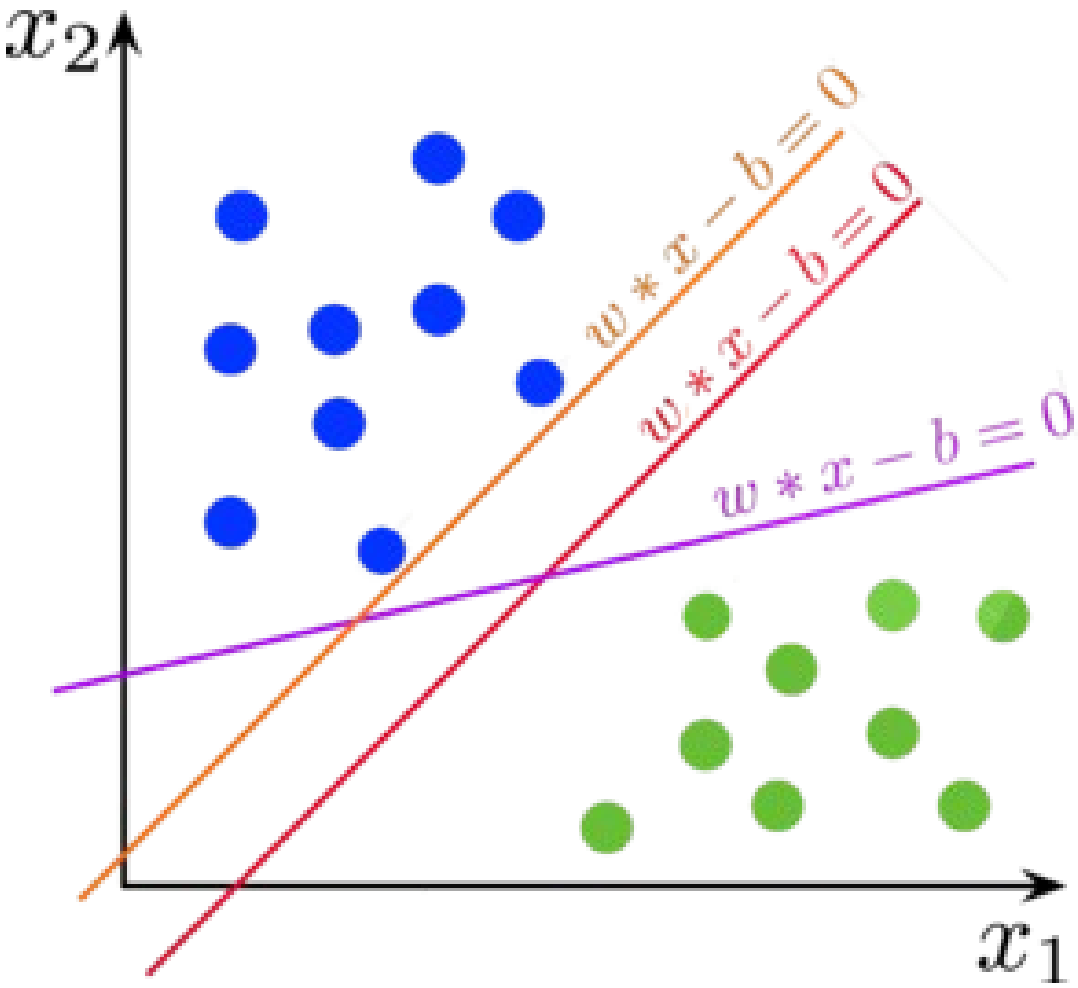


A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Линейный классификатор

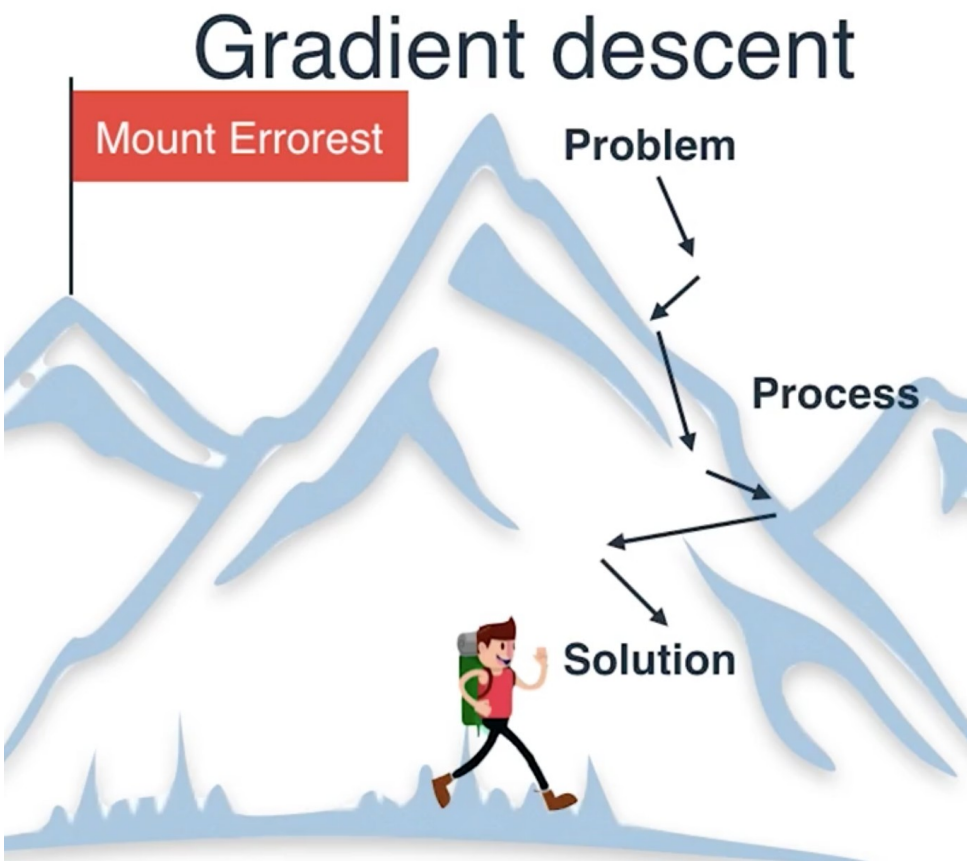
$$\begin{bmatrix} 0.5 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0.4 \\ -1 & 0.2 \\ 0.1 & 0 \\ 2 & -1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 3.2 \end{bmatrix}$$

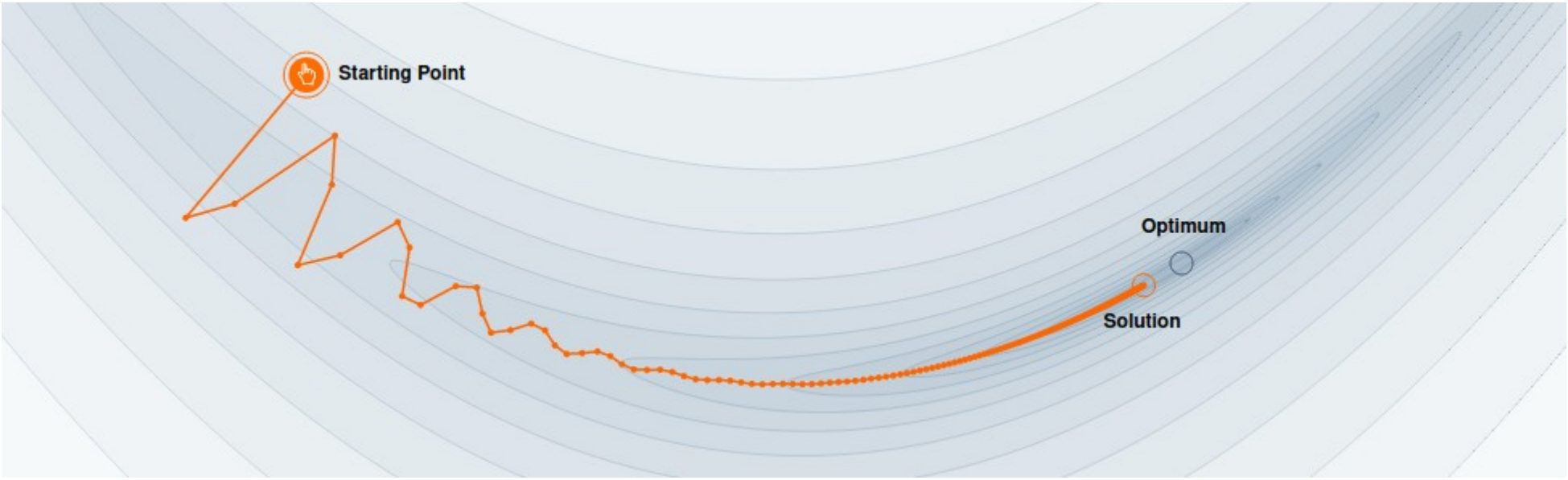
Разделяющие плоскости



https://neerc.ifmo.ru/wiki/index.php?title=Метод_опорных_векторов_(SVM)

Как подобрать “хорошие” параметры?)




Starting Point
Optimum
Solution

Softmax Activation Function

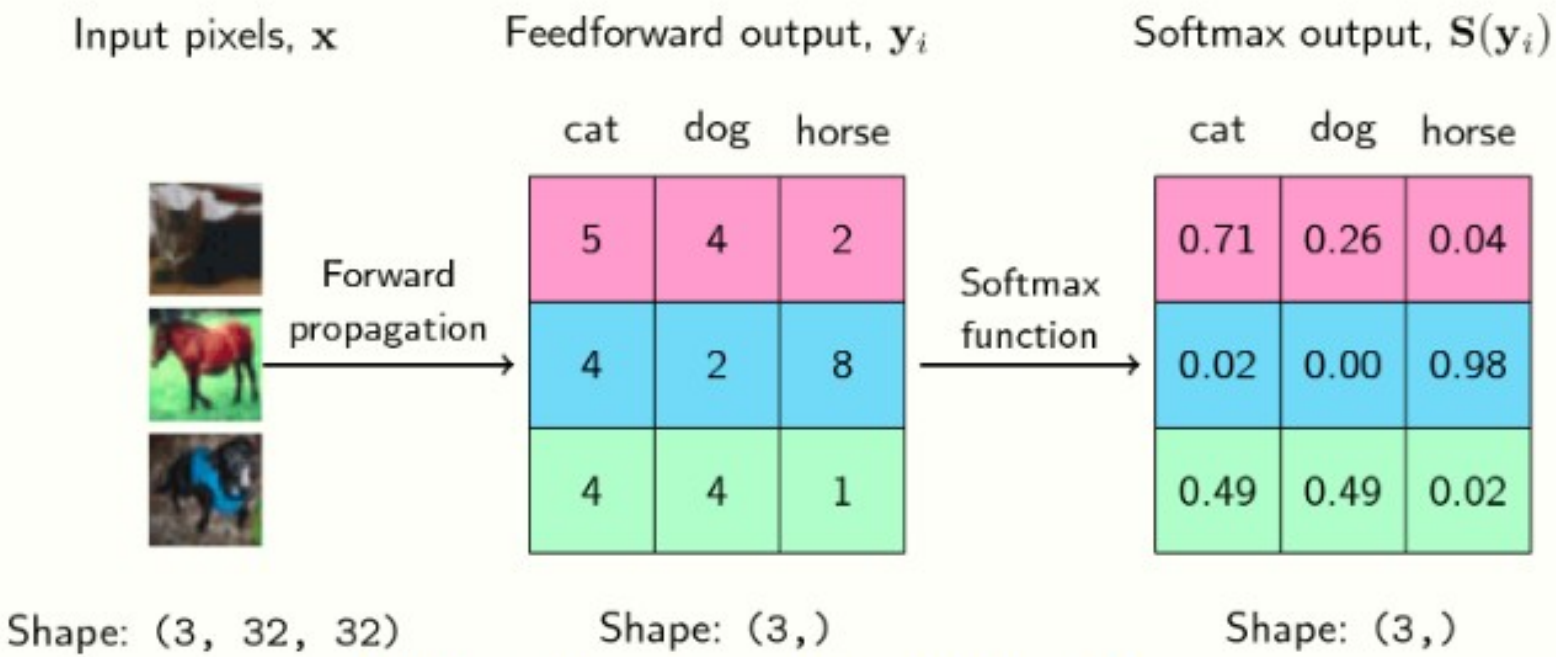


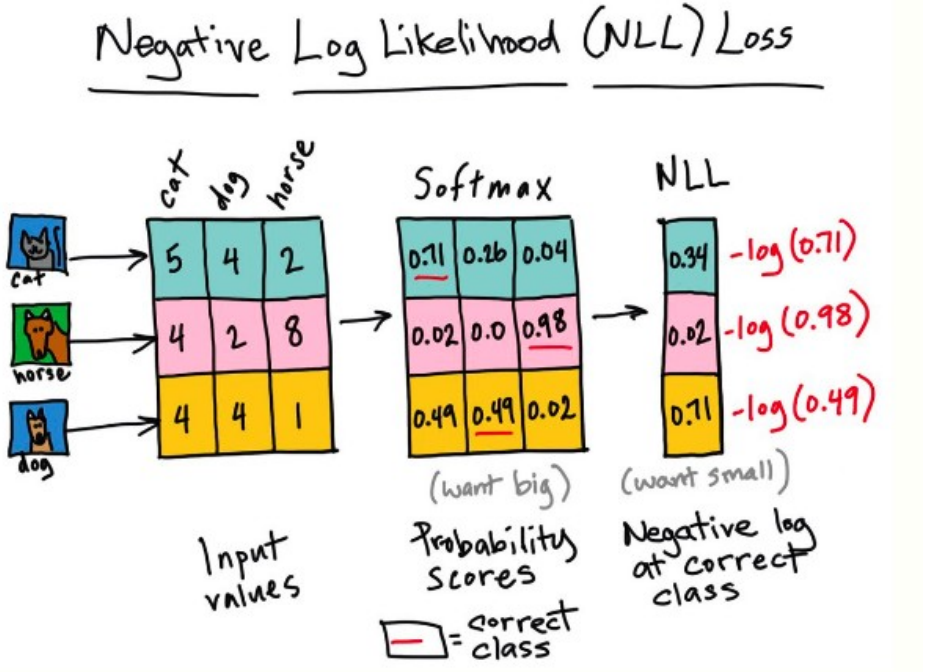
Figure: *Softmax Computation for three classes*

$$p(C=0|x) = \frac{e^{y_0}}{e^{y_0}+e^{y_1}+\cdots+e^{y_n}} = \frac{e^{y_0}}{\sum_i e^{y_i}}$$

$$p(C=1|x) = \frac{e^{y_1}}{e^{y_0}+e^{y_1}+\cdots+e^{y_n}} = \frac{e^{y_1}}{\sum_i e^{y_i}}$$

Подробнее тут

Принцип максимального правдоподобия Maximum likelihood



$$p(data) = \prod_s p(c = gt_s | x_s)$$

w, b

Negative Log-likelihood:

$$-\ln p(data) = -\sum_s \ln p(c = gt_s | x_s)$$

w, b

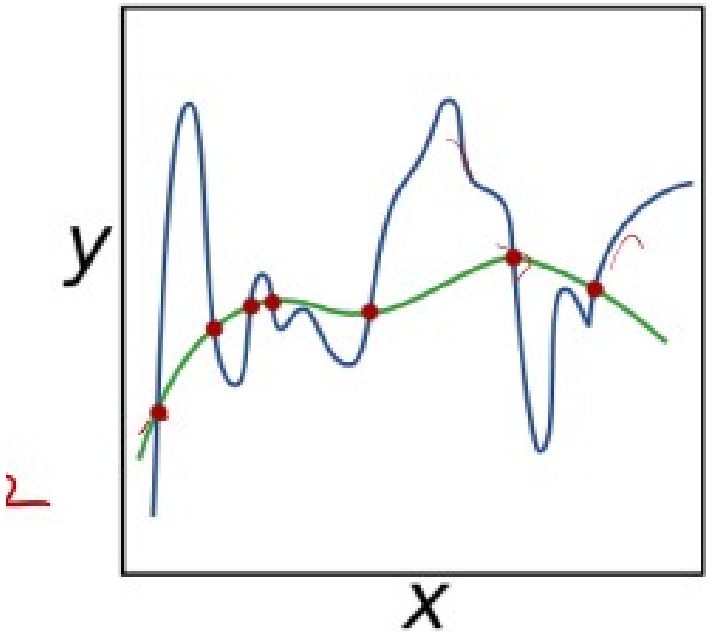
Cross-Entropy loss

Регуляризация
Regularization

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss: Model predictions should match training data}} + \underbrace{\lambda R(W)}_{\text{Regularization: Prevent the model from doing too well on training data}}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data



$$R(w, b) = \|w\|_2^2 + \|b\|_2^2$$

$$\|w\|_2^2 = \left(\sqrt{w_{00}^2 + w_{01}^2 \ldots}\right)^2$$

https://en.wikipedia.org/wiki/Regularization_(mathematics)

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

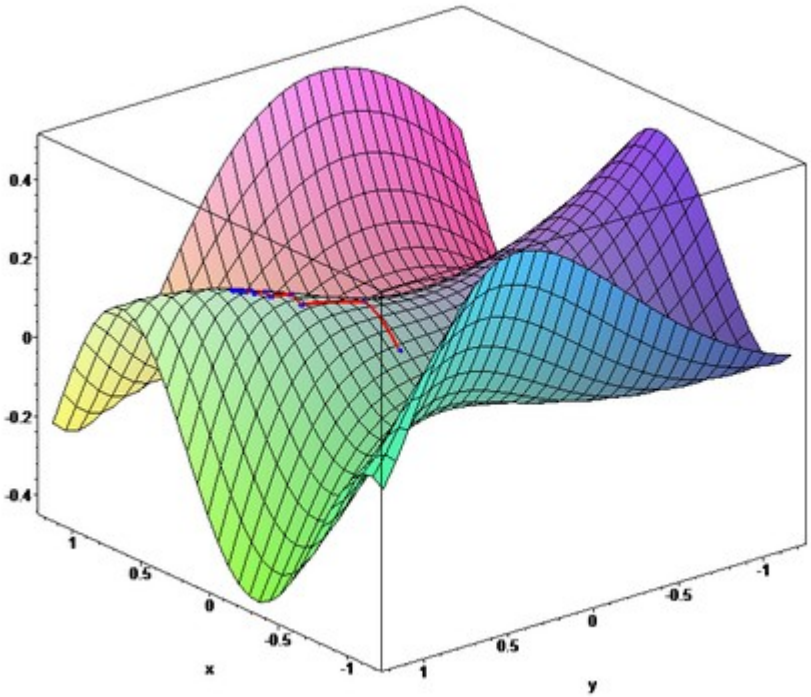
Softmax Function

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

	Unnormalized log-probabilities / logits	exp →	Probabilities must be >= 0: unnormalized probabilities	normalize →	Probabilities must sum to 1: probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

→ L_i = -log(0.13) = **2.04**

Gradient descent
Градиентный спуск



$$\vec{w} = \vec{w} - \eta \vec{\nabla}_w L$$

$$\vec{b} = \vec{b} - \eta \vec{\nabla}_b L$$

$$L'(x) \approx \frac{L(x+\varepsilon) - L(x-\varepsilon)}{2\varepsilon}$$

https://en.wikipedia.org/wiki/Gradient_descent

Стохастический градиентный спуск Stochastic Gradient Descent (SGD)

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(x_i, y_i, W) + \lambda R(W)$$

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

Full sum expensive when N is large!

Approximate sum using a **minibatch** of examples
32 / 64 / 128 common

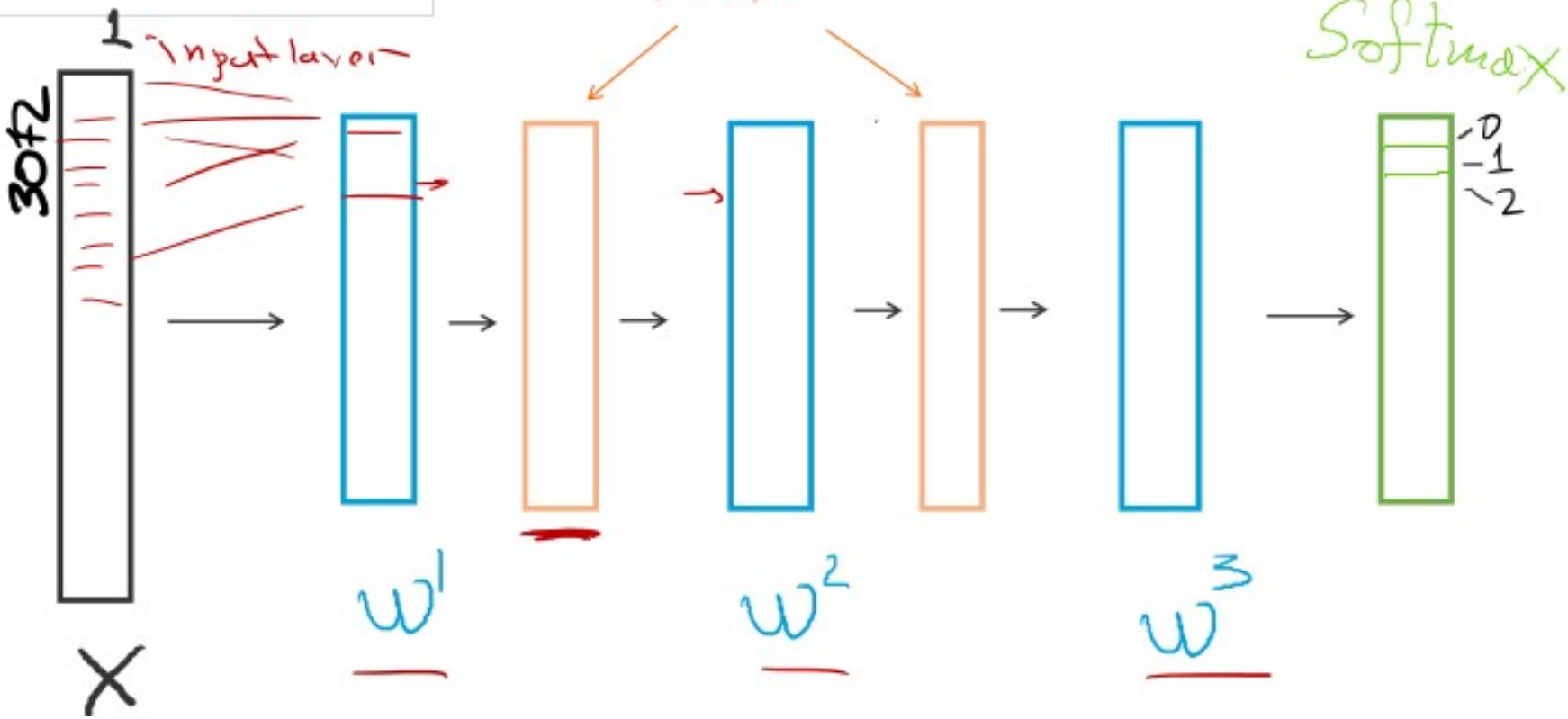
```
# Vanilla Minibatch Gradient Descent

while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Нейронные сети

TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$
Rectified linear unit (ReLU)[10]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

Wikipedia



Что почитать, чтобы разобраться?

- Ian Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning - https://www.deeplearningbook.org - есть перевод на русском языке
- В оригинале на русском – “Глубокое обучение” Николенко, Кадурин
- Курс, на основе которого лекции и семинары - http://cs231n.stanford.edu/
- fast.ai
- Вывод формулы полезной для задания - https://www.youtube.com/watch?v=64nwedrxjPo