Deep Learning
Episode 0

Linear Models

Yandex Data Factory

LAMBDA

British Hedgehog Preservation Society

Linear Regression

Model:

$$X \longrightarrow Wx + b \longrightarrow Y^{pred}$$

Objective function:

$$L = \sum_i \left(y_i - y_i^{pred} \right)^2$$

Optimization (exact):

{that formula}

Linear Regression

Model:

$$X \longrightarrow Wx + b \longrightarrow Y^{pred}$$

Objective function:

$$L = \sum_i \left(y_i - y_i^{pred} \right)^2$$

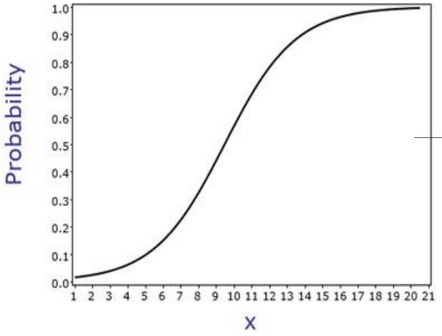
Optimization (iterative):

$$w_0 \leftarrow 0$$

$$w_{i+1} \leftarrow w_i - \alpha \frac{\delta L}{\delta W}$$

$$\frac{\delta L}{\delta W} = \sum_i -2x\left(y_i - \left(wx_i + b\right)\right)$$

Logistic Regression

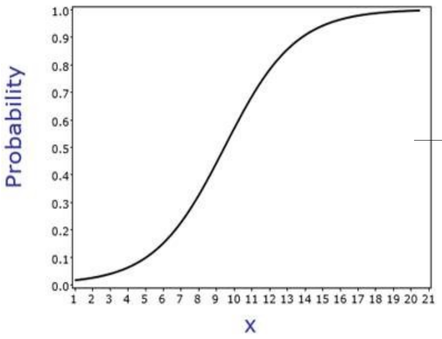
X → Wx +b →  → P(y)

$$P(y)=\sigma(Wx+b)$$

Objective function ?

Logistic Regression

Model:

X ⟶ Wx +b ⟶  ⟶ P(y|X)

Objective function:

$$L = -\sum_i y \log P^{pred}(y) + (1-y)\log(1-P^{pred}(y))$$

Optimization (iterative):

You guessed it!

Logistic Regression

Model:

X

$$a_{[y=a]} = W_a x + b_a$$

$$a_{[y=b]} = W_b x + b_b$$

$$a_{[y=c]} = W_c x + b_c$$

$$\frac{e^{a_{[y=class]}}}{\sum_j e^{a_{[y=j]}}}$$

P(a|X)

P(b|X)

P(c|X)

Objective function:

$$L = -\sum_i \sum_{class} [y_i = class] \log P^{pred}(class \mid X)$$

Gradient descent

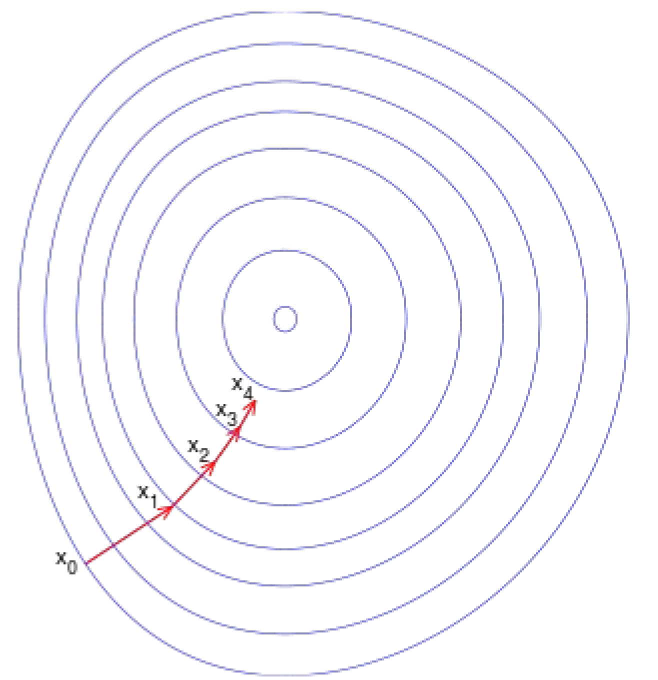
Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$(\text{for } j = 1 \text{ and } j = 0)$$

}



Can we do better?

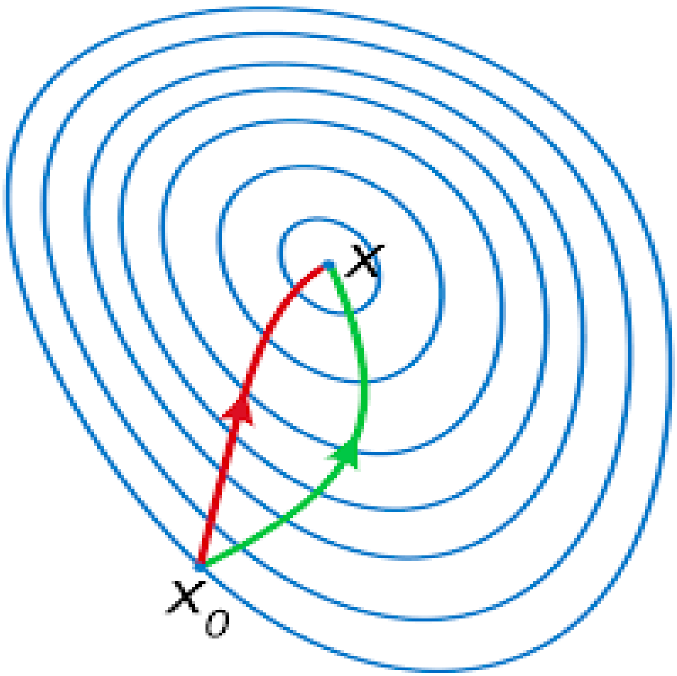
Newton-Raphson

Parameter update

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma\left[\mathbf{H}f(\mathbf{x}_n)\right]^{-1}\nabla f(\mathbf{x}_n).$$

Hessian:

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\[2.2ex] \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\[2.2ex] \vdots & \vdots & \ddots & \vdots \\[2.2ex] \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$



Red: Newton-Raphson
Green: gradient descent

Any drawbacks?

SGD with momentum

Idea: move towards “overall gradient direction”,
Not just current gradient.

$$\Delta w := \eta \nabla Q_i(w) + \alpha \Delta w$$

$$w := w - \Delta w$$

AdaGrad

Idea: decrease learning rate individually for each parameter in proportion to sum of it's gradients so far.

$$Let\; g_{\tau,j} = \frac{\delta L}{\delta w_j} \; on\; \tau_{th}\; tick$$

$$G_{j,j} = \sum_{\tau=1}^{t} g_{\tau,j}^2$$

$$w_j := w_j - \frac{\eta}{\sqrt{G_{j,j}}} g_j.$$

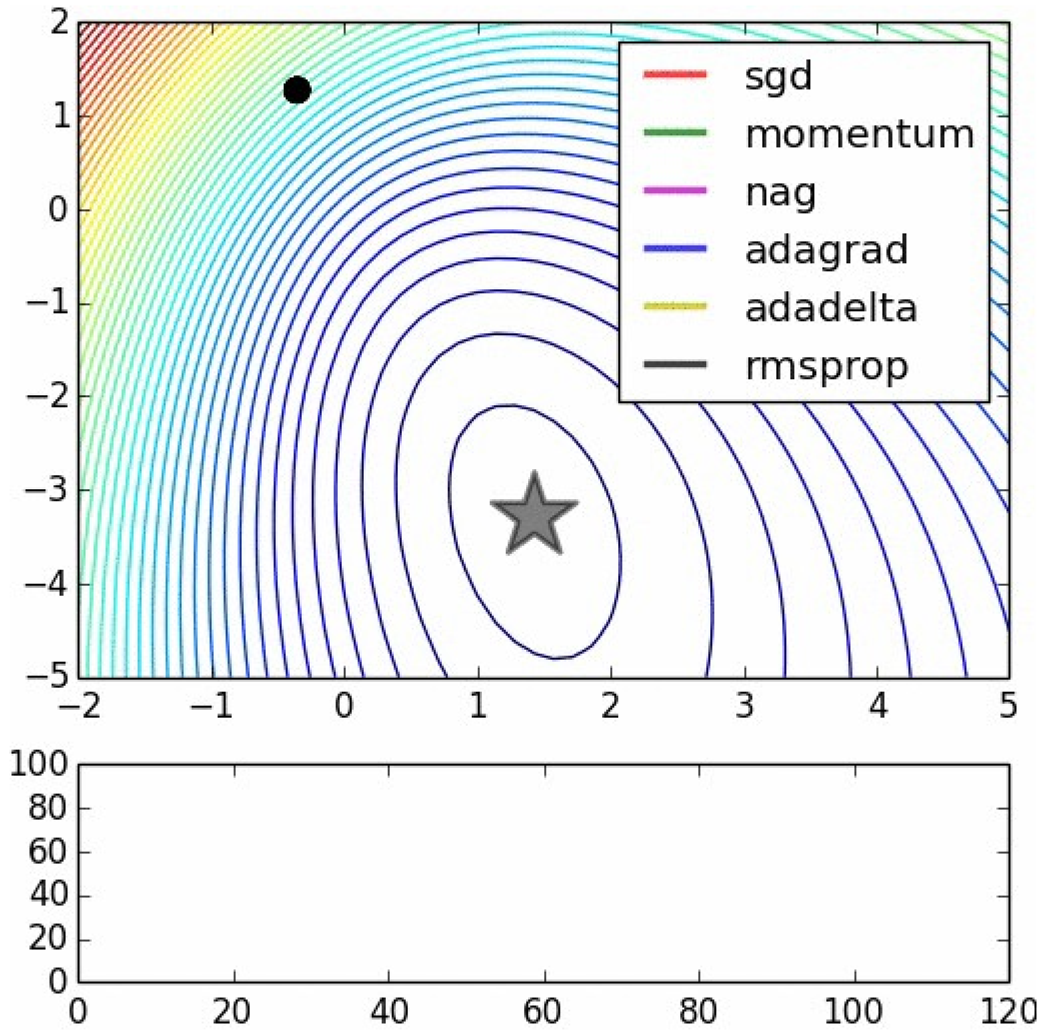
RMSProp

Idea: make sure all gradient steps have approximately same magnitude (by keeping moving average of magnitude)

$$v(w,t) := \gamma v(w,t-1) + (1-\gamma)(\nabla Q_i(w))^2$$

$$w := w - \frac{\eta}{\sqrt{v(w,t)}} \nabla Q_i(w)$$

Alltogether



Moar stuff

Without Hessian

- Adadelta
- Adam
- Adamax
- Nesterov-momentum
- Hessian-free (narrow)
- Conjugate gradients

Estimate inverse Hessian

- BFGS
- L-BFGS
- ****-BFGS

Regularization (weight)

General idea:

$$L_{new} = L + reg$$

performance = how_i_fit_data + how_reasonable_i_am

L2 regularizer

$$L_{new} = L + ||\theta||_2 = L + \sum_i \theta_i^2$$

linear models: theta = {w,b}

- a.k.a. weight decay
- a.k.a. Tikhonov regularizer
- a.k.a. normal prior on params

Regularization (weight)

L2 regularizer

$$L_{new} = L + \sum_i \theta_i^2$$

L1 regularizer

$$L_{new} = L + \sum_i |\theta_i|$$

Difference between L1, L2?

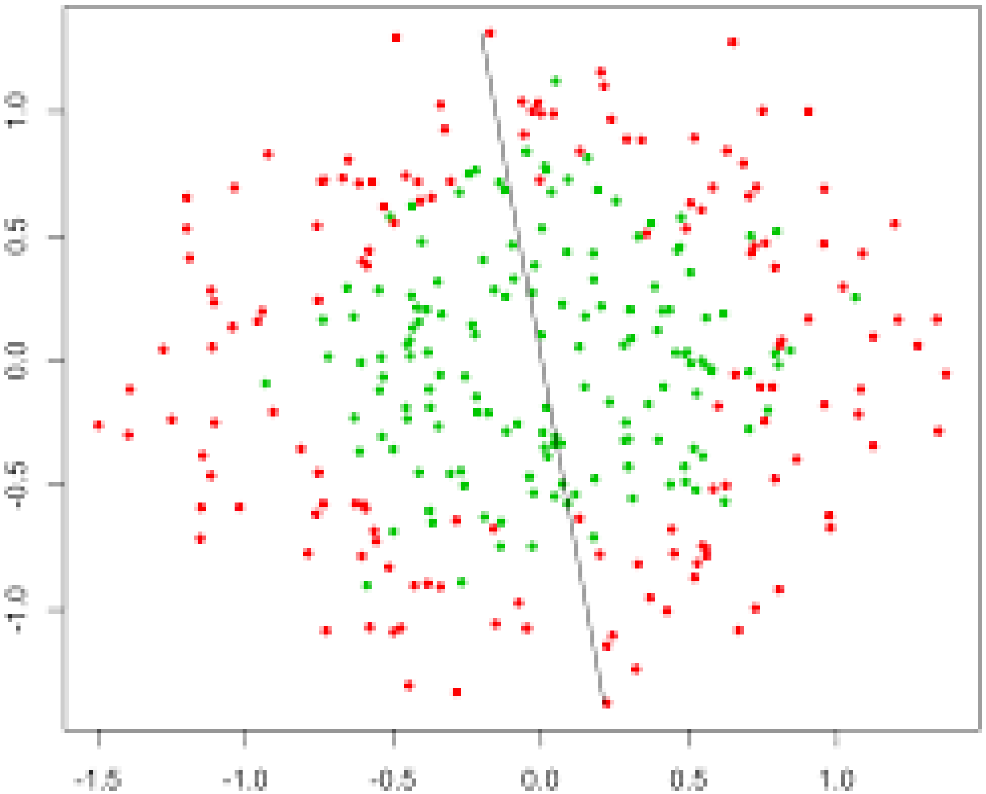
Any other way to regularize?

Regularization(other)

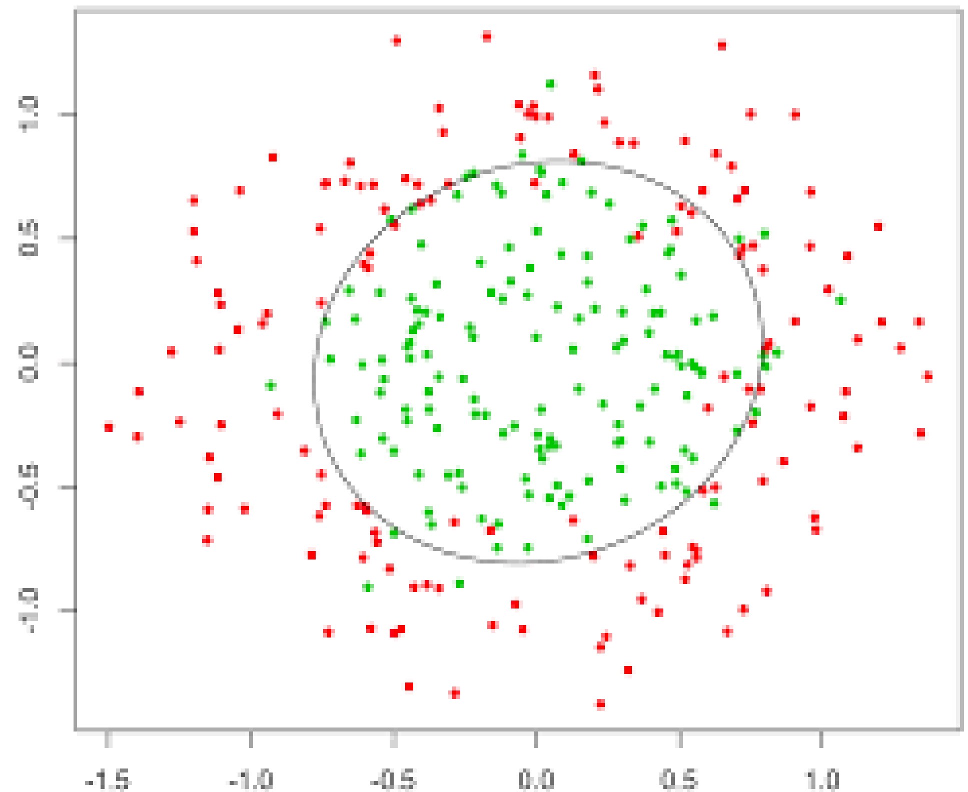
- Distort input
- Distort weights
- Additional objective
- Domain-specific stuff
- Moar data :)
- etc.

Most are domain- or model-specific

Nonlinear dependencies



What we have



What we wan

- How to get that?

Nuff

Go implement that!