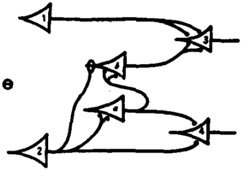
The Math Behind Neural Networks

Justin Sybrandt

Note: I’ve ripped off all images in this presentation.

The Brain and the Machine

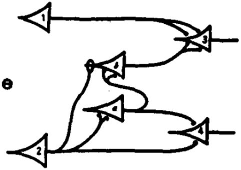
1942 A Logical Calculus of Ideas Immanent in Nervous Activity



- McCulloch & Pitts
- Neurons + Synapses

The Brain and the Machine

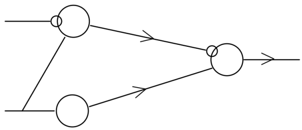
1942 A Logical Calculus of Ideas Immanent in Nervous Activity



- McCulloch & Pitts
- Neurons + Synapses

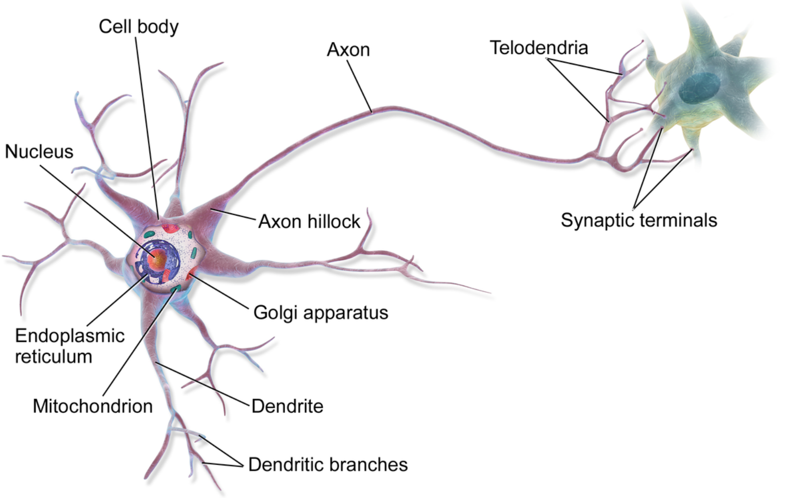
1945 First Draft of a Report on the EDVAC

FIGURE 2



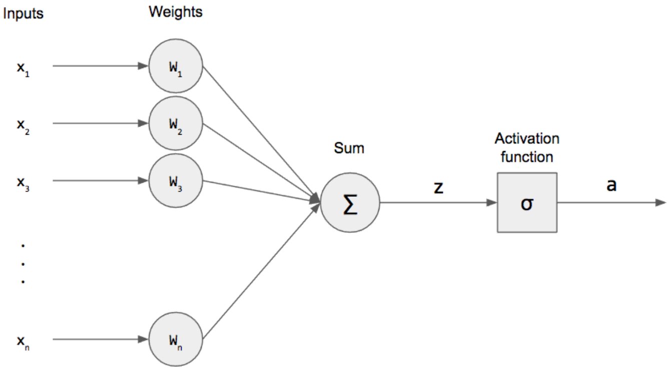
- von Neumann
- Defines E-Elements

Actual Neuron

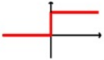
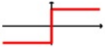


Perceptron

1958 The perceptron: A probabilistic model for information storage and organization in the brain.



Activation Functions

Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z)= \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z)= \begin{cases} 0 & z \leq -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \leq z \leq \frac{1}{2} \\ 1 & z \geq \frac{1}{2} \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z)= \dfrac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z)= \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z)= \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

Perceptron Inference

- Notation

x : input data vector of size n

w : weight vector of size n

α : activation function

o : prediction output

- Inference

$$z = \sum_{i=1}^{n} x_i w_i$$

$$o = \alpha(z)$$

Basic Perceptron Training

- Notation

  t : target label in $\{0, 1\}$

  o : prediction output

  η : learning rate

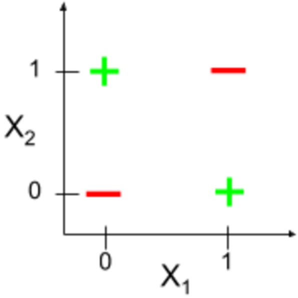
- Update Step: for all (x, t)

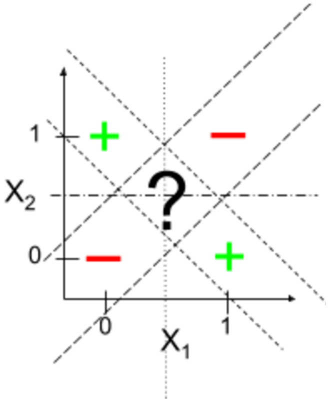
$$w = w + \Delta w$$

$$\Delta w = \eta \, x \, (t - o)$$

- Converges if $x \in X$ is linearly separable

Issue with Basic Training





Perceptron Training with Gradient Descent

- Notation

  $\mathscr{L}(X, w, t)$: loss function

- Update

$$\Delta w = -\eta \frac{\partial \mathscr{L}}{\partial w}$$

Perceptron GD Example

- Mean Squared Error

$$\mathscr{L}_{MSE}(X, w, t) = \frac{1}{2}\,\mathbb{E}_{j=1}^{d}(t_j - o_j)^2$$

- Sigmoid Activation

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

Gradient of MSE

- Gradient Contribution for a single (x, t)

$$\frac{\partial \mathscr{L}}{\partial w_i} = \frac{\partial \mathscr{L}}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial w_i}$$

Gradient of MSE

- Gradient Contribution for a single (x, t)

$$\frac{\partial \mathscr{L}}{\partial w_i} = \frac{\partial \mathscr{L}}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial w_i}$$

$$\frac{\partial \mathscr{L}}{\partial o} = \frac{\partial}{\partial o} \frac{(t-o)^2}{2} = -(t-o)$$

Gradient of MSE

- Gradient Contribution for a single (x, t)

$$\frac{\partial \mathscr{L}}{\partial w_i} = \frac{\partial \mathscr{L}}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial w_i}$$

$$\frac{\partial \mathscr{L}}{\partial o} = \frac{\partial}{\partial o} \frac{(t-o)^2}{2} = -(t-o)$$

$$\frac{\partial o}{\partial z} = \frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))$$

Gradient of MSE

- Gradient Contribution for a single (x, t)

$$\frac{\partial \mathscr{L}}{\partial w_i} = \frac{\partial \mathscr{L}}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial w_i}$$

$$\frac{\partial \mathscr{L}}{\partial o} = \frac{\partial}{\partial o} \frac{(t-o)^2}{2} = -(t-o)$$

$$\frac{\partial o}{\partial z} = \frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))$$

$$\frac{\partial z}{\partial w_i} = \frac{\partial}{\partial w_i} \sum_{j=1}^{n} w_j x_j = x_i$$

Gradient of MSE : Put it All Together

- Gradient Contribution for a single (x, t)

$$\begin{aligned}\frac{\partial \mathscr{L}}{\partial w_i} &= \frac{\partial \mathscr{L}}{\partial o}\frac{\partial o}{\partial z}\frac{\partial z}{\partial w_i} \\ &= -(t-o)\sigma(z)(1-\sigma(z))x_i \\ &= -(t-o)(o-o^2)x_i\end{aligned}$$

- Update Weights

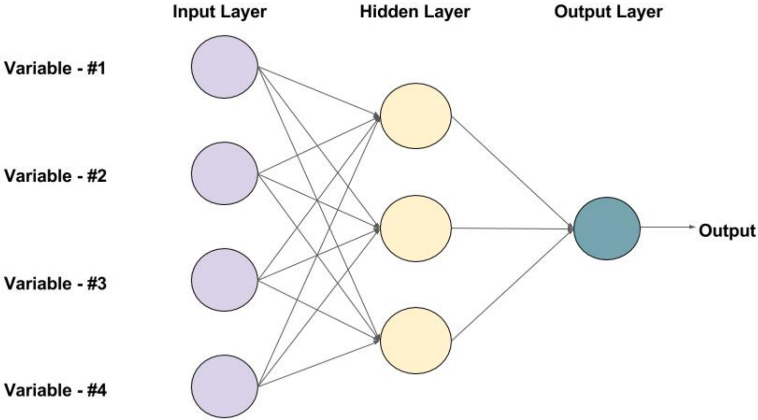
$$\Delta w_i = -\eta\frac{\partial \mathscr{L}}{\partial w_i} = \eta\,\mathbb{E}_{x\in X}(t-o)(o-o^2)x_i$$

Gradient of MSE : Implications

$$\Delta w_i = \eta \, \mathbb{E}_{x \in X} (t - o)(o - o^2) x_i$$

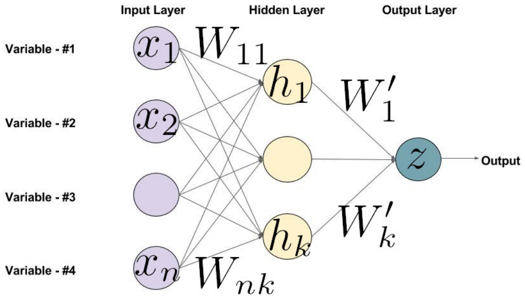
- Weight update is largest when $o = 0.5$, vanishes elsewhere.
- Weight update is proportional to x_i.
- Weight does NOT update if $t = o$.

Feed-Forward Neural Network



An example of a Feed-forward Neural Network with one hidden layer (with 3 neurons)

Feed-Forward Neural Network : Notation



An example of a Feed-forward Neural Network with one hidden layer (with 3 neurons)

$$h_j = \sum_{i=1}^{n} W_{ij} x_i + b_j \qquad z = \sum_{j=1}^{k} W'_j o_j^{(h)} + b'$$

$$o_j^{(h)} = h_j \qquad o = \sigma(z)$$

Back Propagation

- Weight Update:

$$\Delta W = -\eta \frac{\partial \mathscr{L}}{\partial W}$$

Back Propagation

- Weight Update:

$$\Delta W = -\eta \frac{\partial \mathscr{L}}{\partial W}$$

- Derivative in last layer (look familiar):

$$\frac{\partial \mathscr{L}}{\partial W'_j} = \frac{\partial \mathscr{L}}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial W'_j}$$

Back Propagation

- Weight Update:

$$\Delta W = -\eta \frac{\partial \mathscr{L}}{\partial W}$$

- Derivative in last layer (look familiar):

$$\frac{\partial \mathscr{L}}{\partial W'_j} = \frac{\partial \mathscr{L}}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial W'_j}$$

- Derivative in hidden layer:

$$\frac{\partial \mathscr{L}}{\partial W_{ij}} = \frac{\partial \mathscr{L}}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial o_j^{(h)}} \frac{\partial o_j^{(h)}}{\partial h_j} \frac{\partial h_j}{\partial W_{ij}}$$

Gradient at Hidden Layer

- Gradient contribution for a single (x, t) on weight W_{ij}.

$$\frac{\partial \mathscr{L}}{\partial W_{ij}} = \frac{\partial \mathscr{L}}{\partial o} \frac{\partial o}{\partial z} \frac{\partial z}{\partial o_j^{(h)}} \frac{\partial o_j^{(h)}}{\partial h_j} \frac{\partial h_j}{\partial W_{ij}}$$

Gradient at Hidden Layer

- Gradient contribution for a single (x, t) on weight W_{ij}.

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} = \frac{\partial \mathcal{L}}{\partial o}\frac{\partial o}{\partial z}\frac{\partial z}{\partial o_j^{(h)}}\frac{\partial o_j^{(h)}}{\partial h_j}\frac{\partial h_j}{\partial W_{ij}}$$

- Same as before

$$\frac{\partial \mathcal{L}}{\partial o}\frac{\partial o}{\partial z} = -(t-o)(o-o^2)$$

- Hidden Propagation

$$\frac{\partial z}{\partial o_j^{(h)}} = W'_j \quad \frac{\partial o_j^{(h)}}{\partial h_j} = 1 \quad \frac{\partial h_j}{\partial W_{ij}} = x_i$$

One Hidden Layer : Putting it All Together

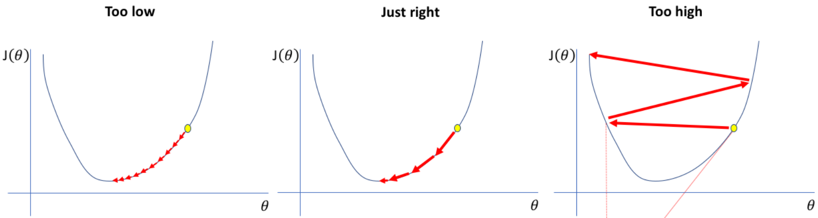
- Last-Layer Weight

$$\frac{\partial \mathscr{L}}{\partial W'_j} = \frac{\partial \mathscr{L}}{\partial o}\frac{\partial o}{\partial z}\frac{\partial z}{\partial W'_j}$$
$$= -(t-o)(o-o^2)h_j$$

- Inner-Layer Weight

$$\frac{\partial \mathscr{L}}{\partial W_{ij}} = \frac{\partial \mathscr{L}}{\partial o}\frac{\partial o}{\partial z}\frac{\partial z}{\partial o_j^{(h)}}\frac{\partial o_j^{(h)}}{\partial h_j}\frac{\partial h_j}{\partial W_{ij}}$$
$$= -(t-o)(o-o^2)W'_j x_i$$

Effect of Learning Rate

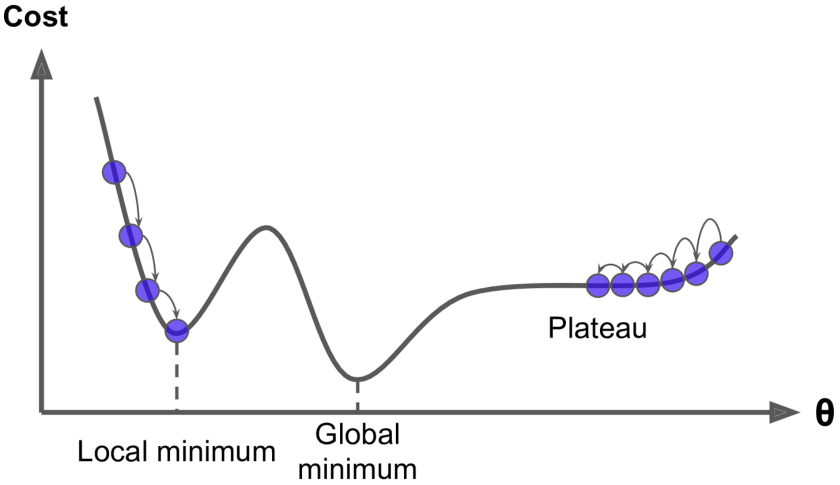


A small learning rate requires many updates before reaching the minimum point

The optimal learning rate swiftly reaches the minimum point

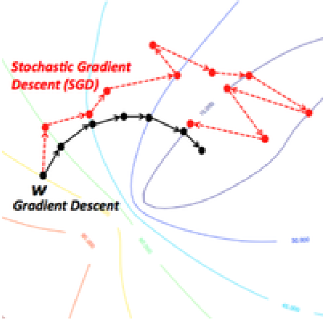
Too large of a learning rate causes drastic updates which lead to divergent behaviors

Local Minima



Stochastic Gradient Descent

- Sample many batches of size b from $X^{d \times n}$.
- $b << n$.
- Allows small incorrect steps during training.
- Better overcomes local minima.



Gradient Descent Modifications

- Momentum
- Nesterov

Momentum

- Original

$$\Delta W = -\eta \frac{\partial \mathcal{L}}{\partial W}$$

- With Momentum

$$M_k = \beta M_{k-1} + \alpha \frac{\partial \mathcal{L}}{\partial W}$$

$$\Delta W = -\eta M_k$$

Nesterov

- Original

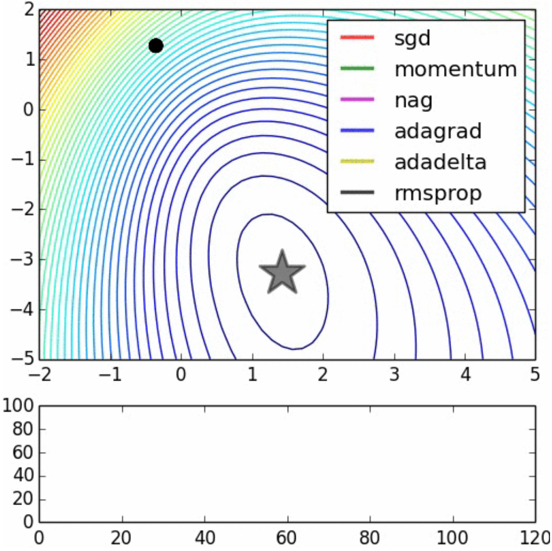
$$\Delta W = -\eta \frac{\partial \mathscr{L}}{\partial W}$$

- With Nesterov

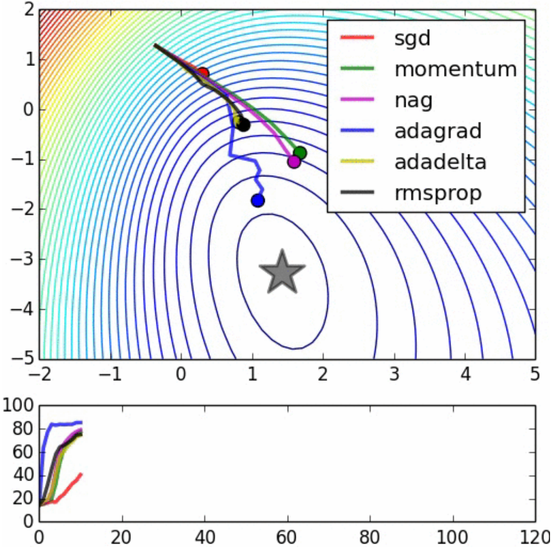
$$N_k = \beta N_{k-1} + \alpha \frac{\partial}{\partial W} \mathscr{L}\left(X, (W - \beta N_{k-1}), t\right)$$

$$\Delta W = -\eta N_k$$

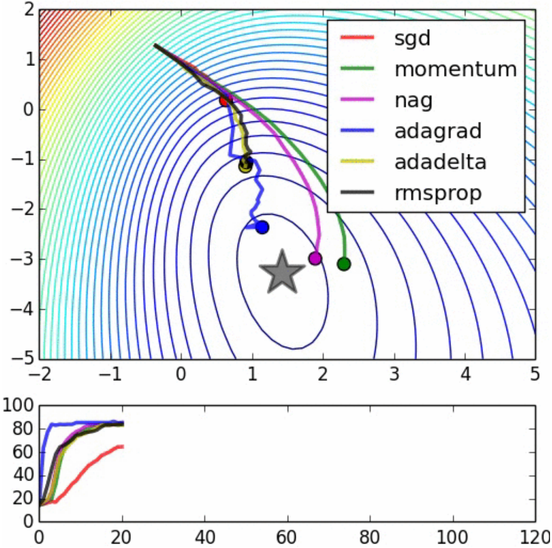
GD Modifications Visualization



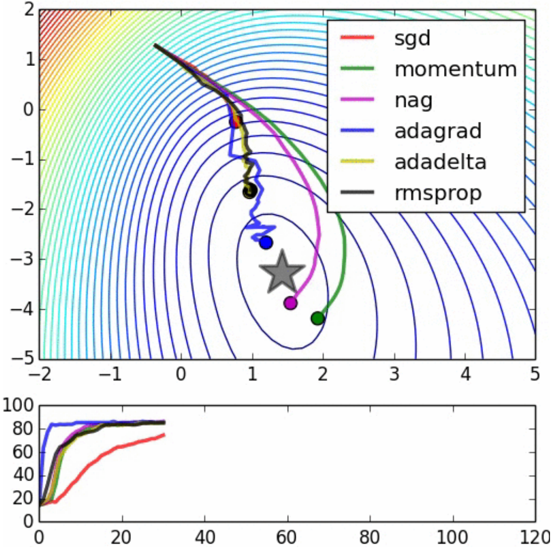
GD Modifications Visualization



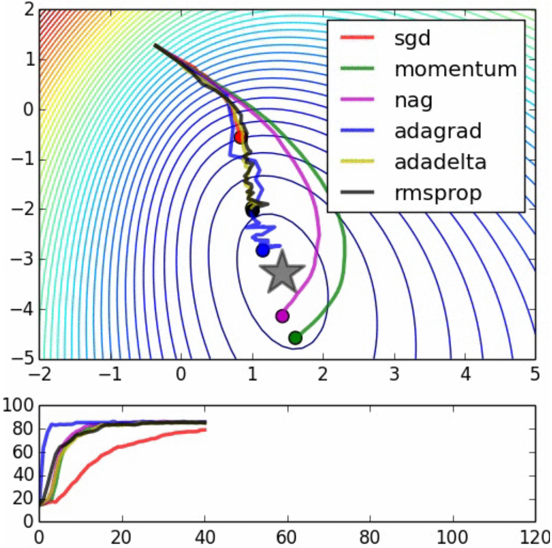
GD Modifications Visualization



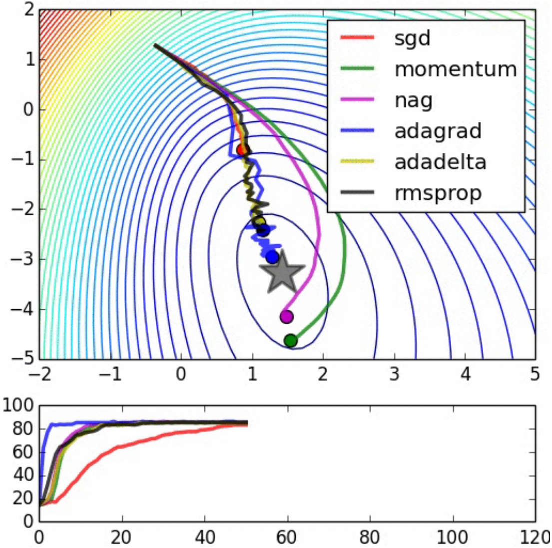
GD Modifications Visualization



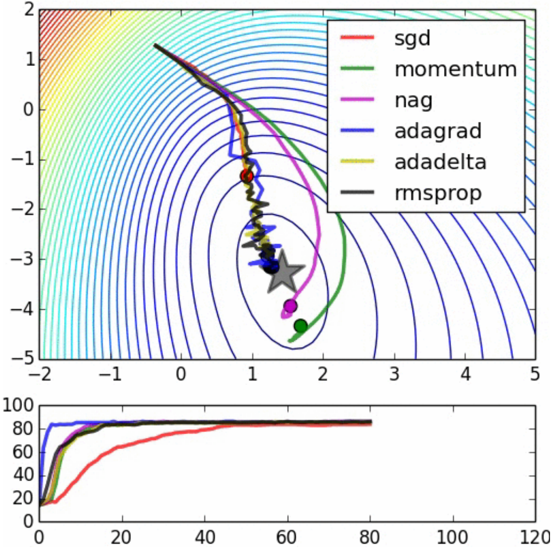
GD Modifications Visualization



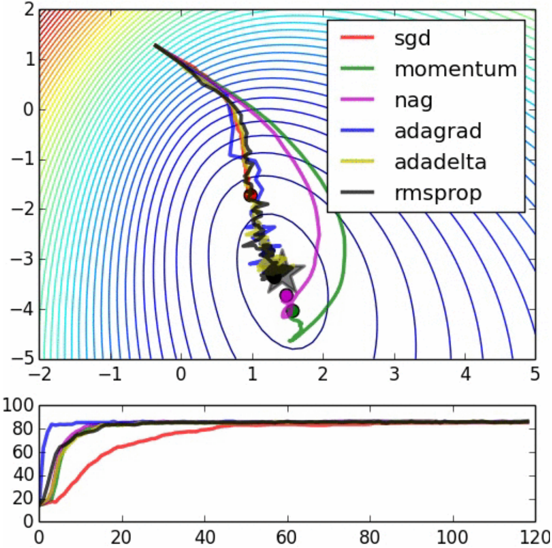
GD Modifications Visualization



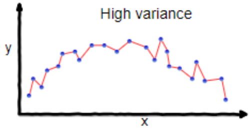
GD Modifications Visualization



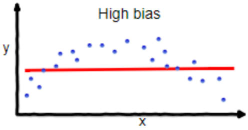
GD Modifications Visualization



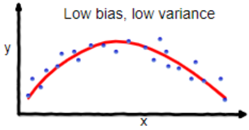
Bias and Variance



overfitting



underfitting



Good balance

Bias and Variance

- In case of bias:
  - Increase model parameters
  - Increase features
  - Lower learning rate
- In case of variance:
  - Increase data
  - Remove features
  - Add regularization terms
  - Raise learning rate