Improving the Way Neural Networks Learn

October 27, 2019

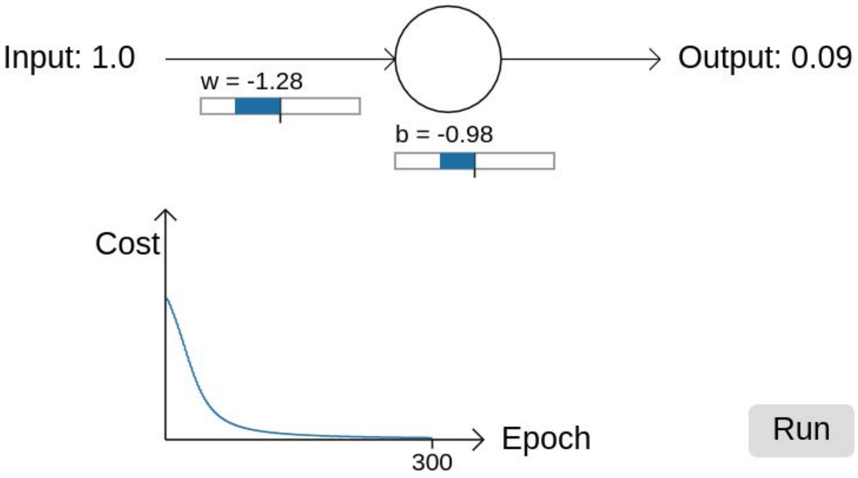
Topics Covered

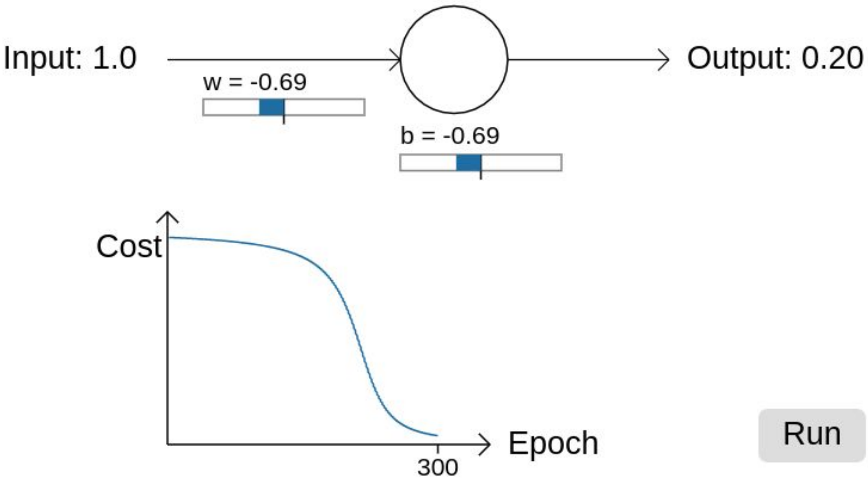
- Cross-entropy cost function
- Softmax
  - Log-likelihood cost function
- Overfitting
  - L2 regularization
  - L1 regularization
  - Dropout
  - Expansion of training data
- Weight initialization
- Hypertuning parameters
- Stochastic gradient descent variations
- Neuron activation variations

Cross-Entropy Cost Function Motivation

Motivation:

- Quadratic cost function learns slowly when the cost is high as ∂C/∂w and ∂C/∂b are small





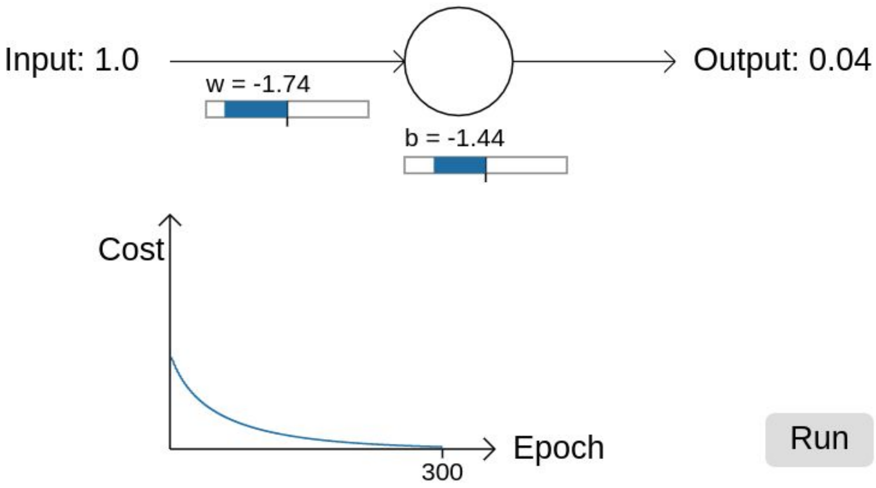
Cross-Entropy Cross Function

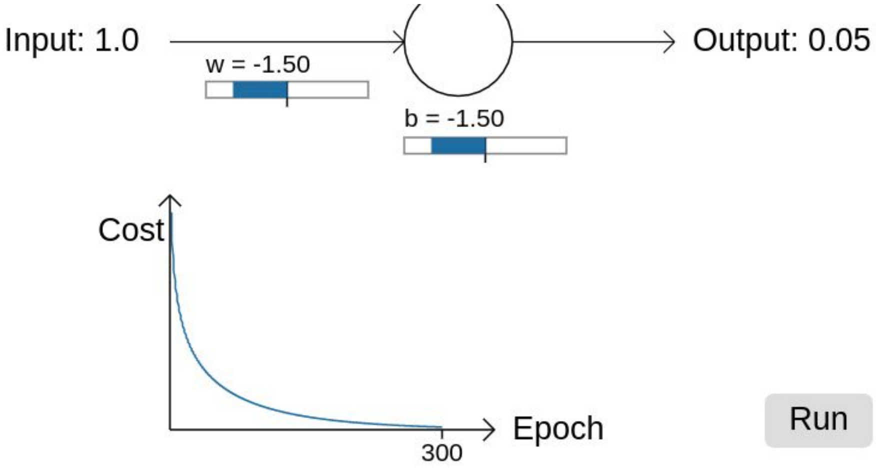
$$C = -\frac{1}{n} \sum_x [y \ln a + (1 - y) \ln(1 - a)]$$

$$\frac{\partial C}{\partial w_j} = \frac{1}{n} \sum_x x_j(\sigma(z) - y).$$

$$\frac{\partial C}{\partial b} = \frac{1}{n} \sum_x (\sigma(z) - y)$$

Cross-Entropy Learning





Softmax

$$\sum_j a_j^L = \frac{\sum_j e^{z_j^L}}{\sum_k e^{z_k^L}} = 1$$

- Outputs a probability distribution

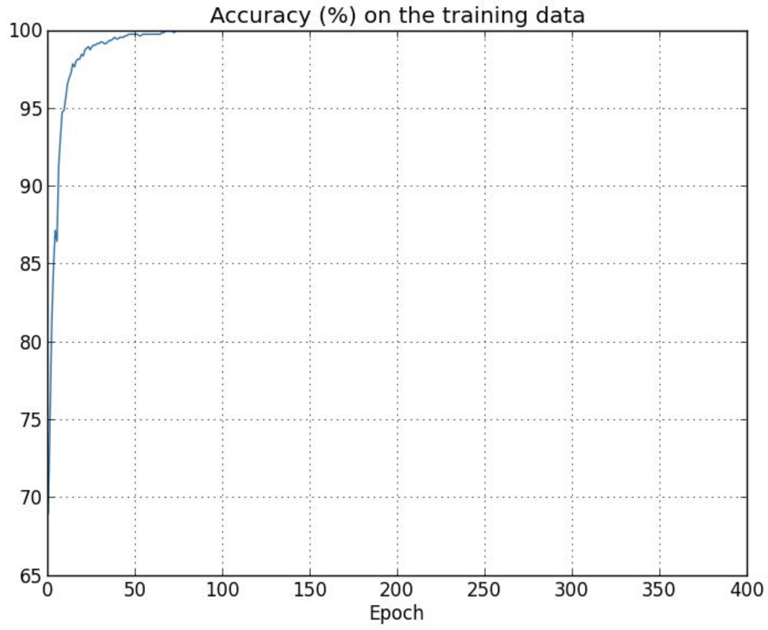
Log-Likelihood Cost

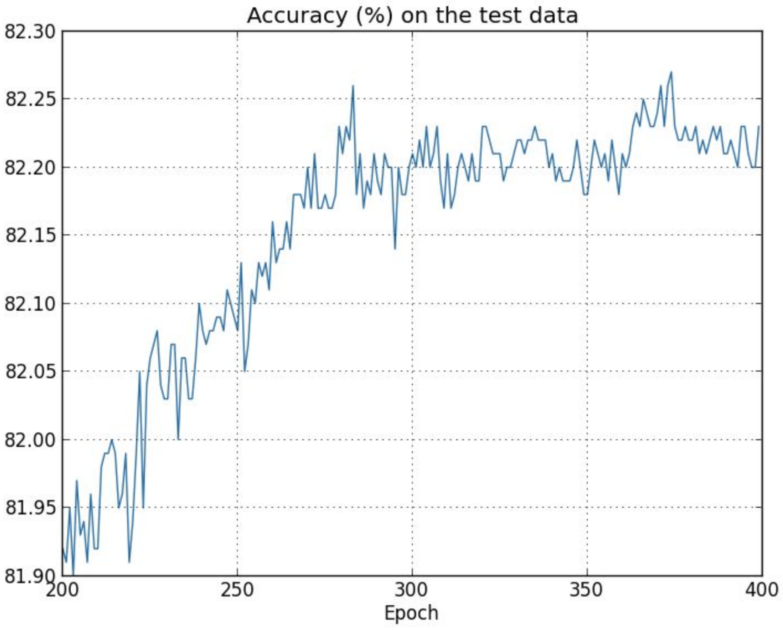
$$C \equiv -\ln a_y^L$$

$$\frac{\partial C}{\partial b_j^L} = a_j^L - y_j$$

$$\frac{\partial C}{\partial w_{jk}^L} = a_k^{L-1}(a_j^L - y_j)$$

Overfitting





L2 Regularization

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2$$

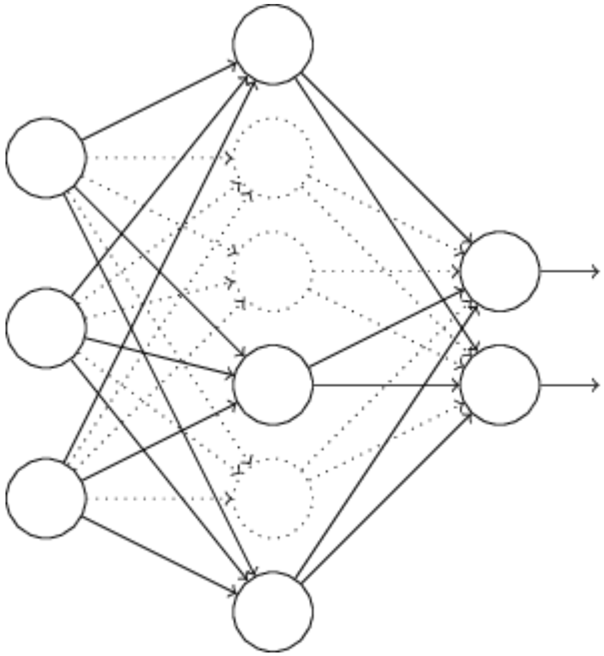
- C_0 is the original cost function
- ƛ is the regularization parameter
- Shrinks weights by an amount proportional to w

L1 Regularization

$$C = C_0 + \frac{\lambda}{n} \sum_w |w|$$

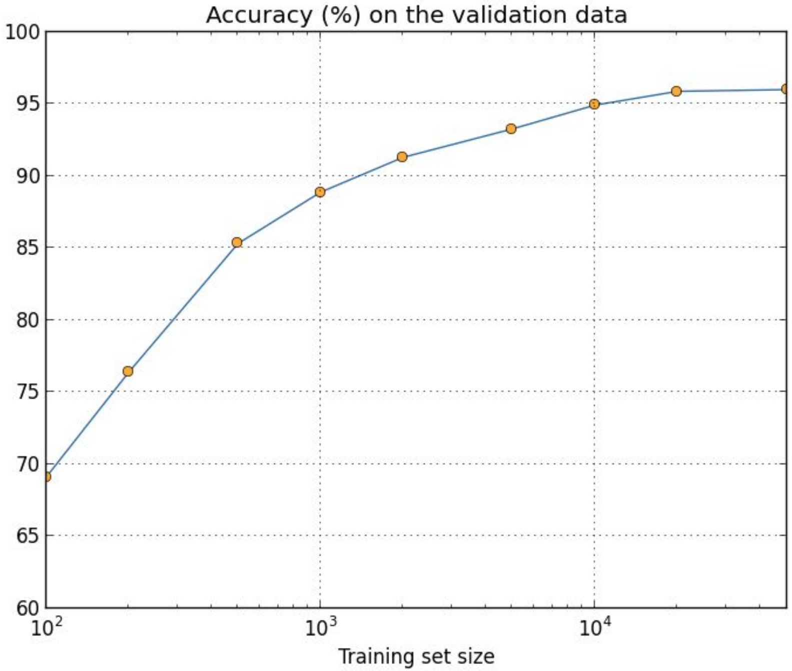
- Shrinks weights by a constant amount

Dropout



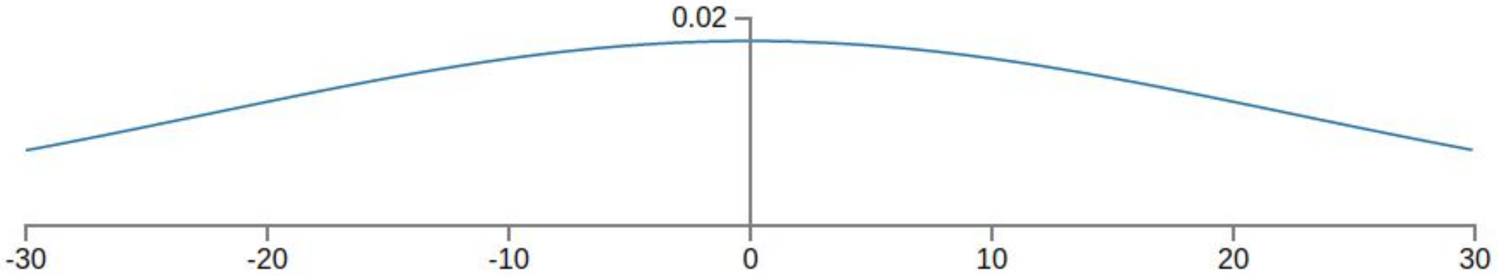
- Each mini-batch, dropout a random subset of neurons
- Produces an effect similar to averaging different networks as each neuron can not rely an another, forcing them to learn more robust features

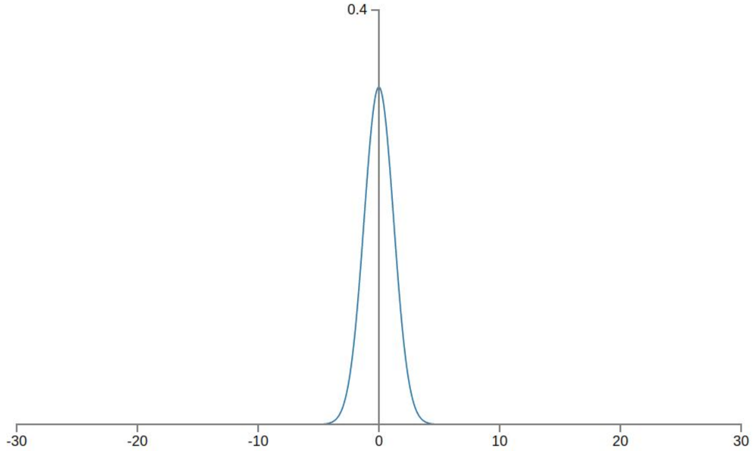
Artificially Expand the Training Data



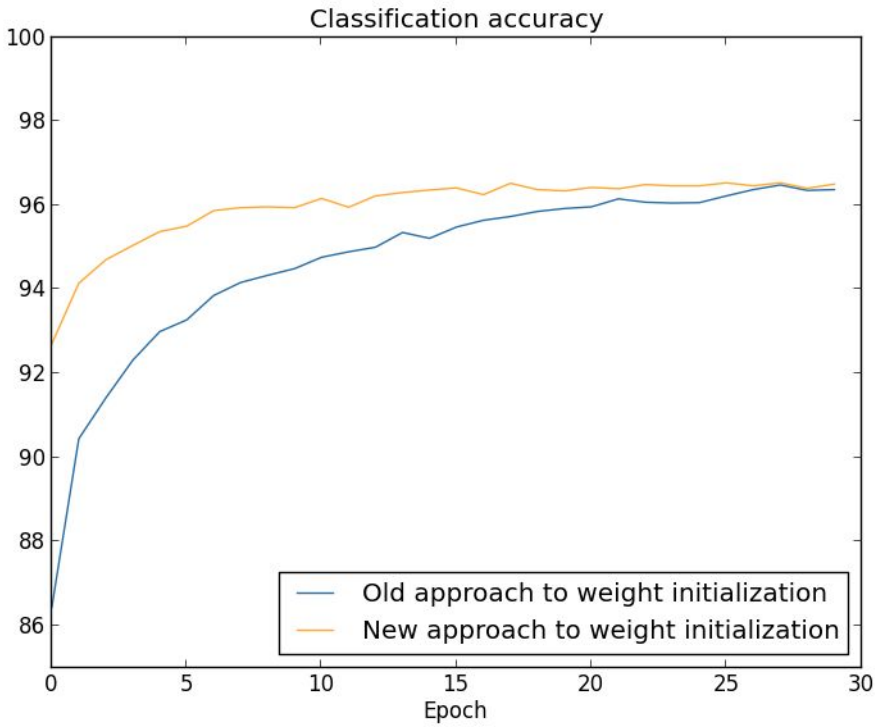
- Model vs data
- "'Our whiz-bang technique gave us an improvement of X percent on standard benchmark Y' is a canonical form of research claim."

Weight Initialization





Weight Initialization Results



- Gaussian distribution with mean 0 and variance of 1 vs variance of 1 / sqrt(x)

Hypertuning Parameters

- Largely heuristic
  - Try magnitudes of 10 on subset of data
- Varying learning rate
- Early stopping
- Gridsearch

Hessian Technique

$$\Delta w = -H^{-1}\nabla C.$$

- Hessian matrix - matrix of partial second derivatives
- Theoretically converges in fewer steps
- Hessian matrix is HUGE, makes computation difficult

Momentum-Based Gradient Descent

$$v \to v' = \mu v - \eta \nabla C$$
$$w \to w' = w + v'.$$

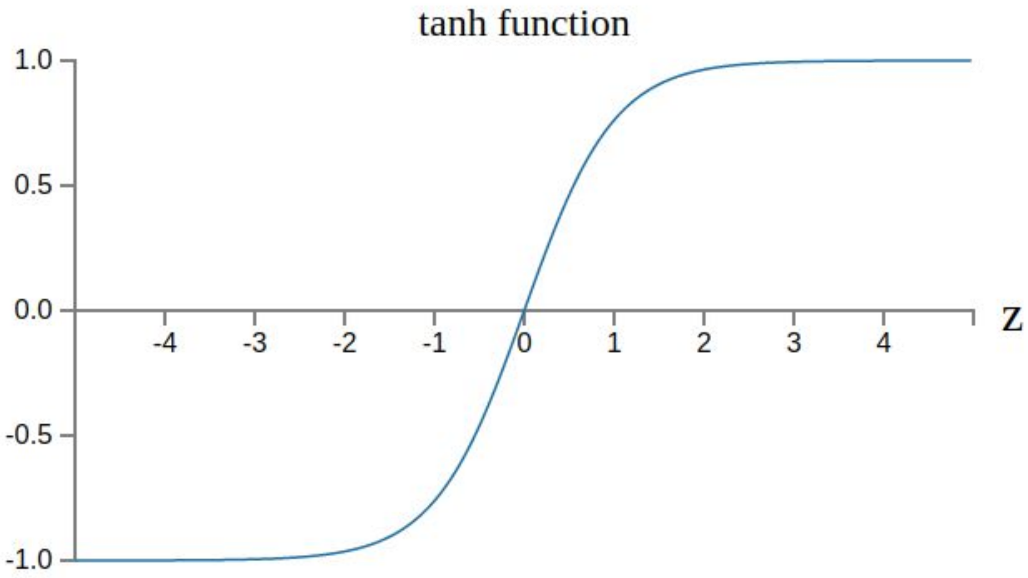
- Uses second derivative information like the Hessian technique
- Allows for faster convergence without overshooting

Other Methods of Minimization

- BFGS
- L-BFGS
- Nesterov’s accelerated gradient technique

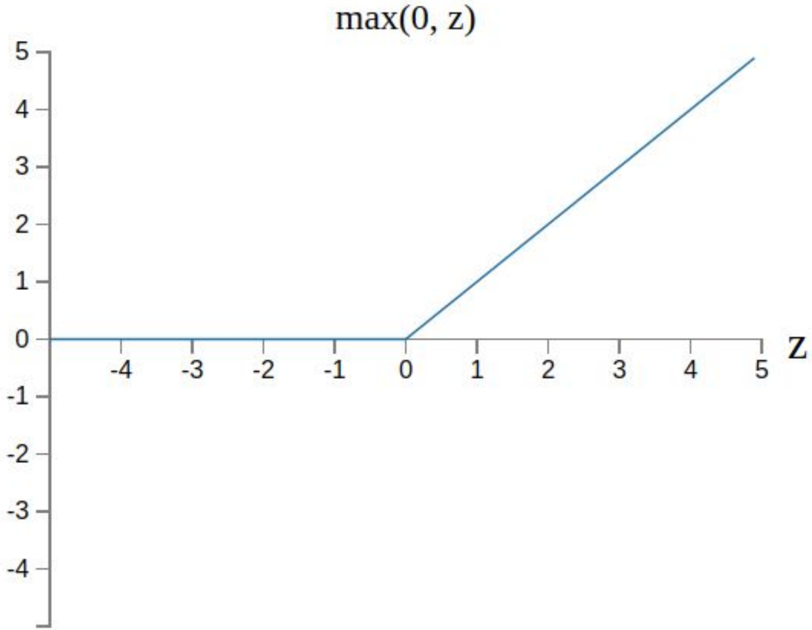
Hyperbolic Tangent Function

- $z = \tanh(w \cdot x + b)$
- Allows for negative and positive weight changes in one pass



Rectified Linear Neuron

$z = \max(0, w \cdot x + b)$



Question: *How do you approach utilizing and researching machine learning techniques that are supported almost entirely empirically, as opposed to mathematically? Also in what situations have you noticed some of these techniques fail?*

Answer: You have to realize that our theoretical tools are very weak. Sometimes, we have good mathematical intuitions for why a particular technique should work. Sometimes our intuition ends up being wrong [...] The questions become: how well does my method work on this particular problem, and how large is the set of problems on which it works well.

\- *Question and answer with neural networks researcher Yann LeCun*