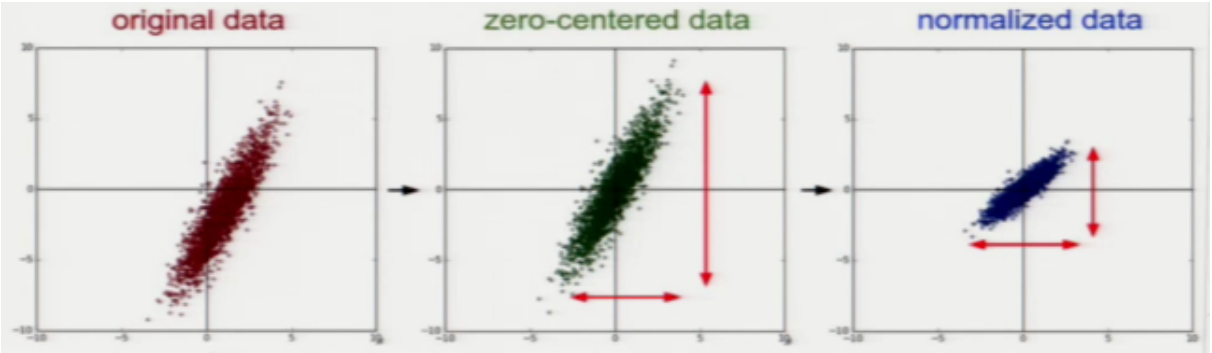
04/06/2016

Heuristic for Avoiding Bad Local Minima (continued)

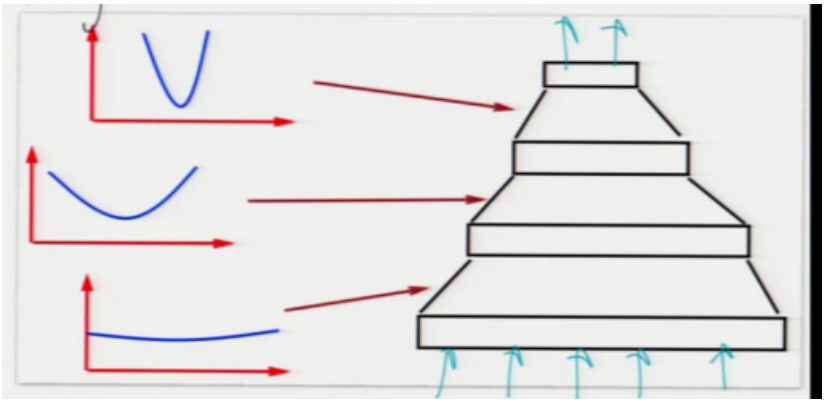
- Train several nets; pick best.
- Layer-wise pre-training

Heuristics for Faster Training

- 1, 2, 3, 4 from last lecture.
- Stochastic gradient descent faster than batch gradient descent on large, redundant data sets. One epoch presents every training example once. Training usually takes many epochs, but if data is huge it can take less than one.
- Normalizing the data:
  1. Center each feature so mean is zero (gets sigmoid into good operating region).
  2. Then scale each feature so variance is constant (~ 1 is good). This makes objective function well conditioned. Condition number of a matrix is the largest eigenvalue divided by smallest eigenvalue gives a measure of how acentric the ellipsoids are which gives an idea how much time we'll spend zig-zagging before getting to the minimum.
- Normalizing data:



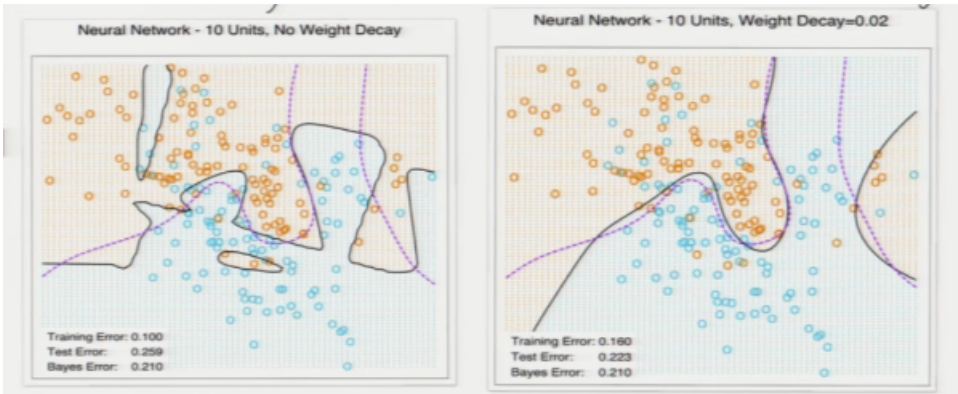
- Centering the hidden units helps too.
  1. Replace sigmoids with $2s(\gamma) - 1$ or with $\tanh \gamma$.
- Use different learning rate for each layer of weights. Gradients are much smaller for early layers than later layers.



- Emphasizing schemes:
  - Shuffle so successive examples are never/rarely same class.
  - Present examples from rare classes more often, or with bigger learning rate ϵ.
  - Warning: can backfire on bad outliers.
- Second-order optimization.
  - No Newton's method: Hessian too expensive.
  - Nonlinear conjugate gradient; for small nets + small data + regression. Batch descent only! $\rightarrow$ too slow with redundant data.
  - Stochastic Levernberg Marquardt; approximates a diagonal Hessian.

Heuristic to Avoid Overfitting

- Ensemble of neural nets. Random weights; bagging. Slow!
- L_2 regularization, aka weight decay. Add $\lambda||w||^2$ to the cost/loss function, where w is vector of all weights. Effect: $\frac{\partial J}{\partial w_i}$ has extra term $-2\lambda w_i$. Weight decays by factor $(1 - 2\lambda)$ if reinforced by training.



- Dropout emulates an ensemble in one network. (Faster training too).

Convolutional Neural Networks (ConvNets)

- Vision: inputs are large images. 200x200 = 40,000 pixels.
- If we connect them all to 40,000 hidden units $\rightarrow$ 1.6 billion connections.
- Neural nets are often over-parametized: too many weights, too little data.
- ConvNet ideas:
  1. Local connectivity: A hidden unit (in early layer) connects only to small patch of units in previous layer.
  2. Shared weights:
     - Groups of hidden units share same set of input weights, called a mask/filter/kernel.
     - We learn several masks.
     - Masks x patches = hiddent unit (in first hidden layer).
     - If one patch learns to detect edges, every patch has an edge detector.
     - ConvNets exploit repeated structure in images, audio.
     - Convolutions: the same linear transformation applied to different parts of image by shifting.