Neural networks

Deep learning - dropout
Topics: why training is hard

• Depending on the problem, one or the other situation will tend to prevail

• If first hypothesis (underfitting): use better optimization
  ‣ this is an active area of research

• If second hypothesis (overfitting): use better regularization
  ‣ unsupervised learning
  ‣ stochastic «dropout» training
Topics: dropout

- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful

- Could use a different dropout probability, but 0.5 usually works well
Topics: dropout

- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful

- Could use a different dropout probability, but 0.5 usually works well
Topics: dropout

- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful

- Could use a different dropout probability, but 0.5 usually works well
Topics: dropout

- Use random binary masks $m^{(k)}$
  - layer pre-activation for $k > 0$: $h^{(0)}(x) = x$
    $$a^{(k)}(x) = b^{(k)} + W^{(k)}h^{(k-1)}(x)$$
  - hidden layer activation ($k$ from 1 to $L$):
    $$h^{(k)}(x) = g(a^{(k)}(x))$$
  - output layer activation ($k = L + 1$):
    $$h^{(L+1)}(x) = o(a^{(L+1)}(x)) = f(x)$$
Topics: dropout

- Use random binary masks \( m^{(k)} \)
  - layer pre-activation for \( k>0 \) \( h^{(0)}(x) = x \)
    \[ a^{(k)}(x) = b^{(k)} + W^{(k)}h^{(k-1)}(x) \]
  - hidden layer activation \( k \) from 1 to \( L \):
    \[ h^{(k)}(x) = g(a^{(k)}(x)) \circ m^{(k)} \]
  - output layer activation \( k=L+1 \):
    \[ h^{(L+1)}(x) = o(a^{(L+1)}(x)) = f(x) \]
Topics: dropout backpropagation

- This assumes a forward propagation has been made before
  - compute output gradient (before activation)
    \[ \nabla a^{(L+1)}(x) - \log f(x)_y \iff - (e(y) - f(x)) \]
  - for \( k \) from \( L+1 \) to 1
    - compute gradients of hidden layer parameter
      \[ \begin{align*}
      \nabla w^{(k)} - \log f(x)_y & \iff (\nabla a^{(k)}(x) - \log f(x)_y) \cdot h^{(k-1)}(x)^\top \\
      \nabla b^{(k)} - \log f(x)_y & \iff \nabla a^{(k)}(x) - \log f(x)_y
      \end{align*} \]
    - compute gradient of hidden layer below
      \[ \nabla h^{(k-1)}(x) - \log f(x)_y \iff W^{(k)^\top} (\nabla a^{(k)}(x) - \log f(x)_y) \]
    - compute gradient of hidden layer below (before activation)
      \[ \nabla a^{(k-1)}(x) - \log f(x)_y \iff (\nabla h^{(k-1)}(x) - \log f(x)_y) \odot [\ldots, g'(a^{(k-1)}(x)_j), \ldots] \]
Topics: dropout backpropagation

- This assumes a forward propagation has been made before
  - compute output gradient (before activation)
    \[ \nabla_{a^{(L+1)}(x)} \log f(x)_y \iff -(e(y) - f(x)) \nabla_{a^{(L+1)}(x)} - \log f(x)_y \iff -(e(y) - f(x)) \]
  - for \( k \) from \( L+1 \) to 1
    - compute gradients of hidden layer parameter
      \[ \nabla_{W^{(k)}} - \log f(x)_y \iff \left( \nabla_{a^{(k)}(x)} - \log f(x)_y \right) h^{(k-1)}(x)^\top \]
      \[ \nabla_{b^{(k)}} - \log f(x)_y \iff \nabla_{a^{(k)}(x)} - \log f(x)_y \]
    - compute gradient of hidden layer below
      \[ \nabla_{h^{(k-1)}(x)} - \log f(x)_y \iff W^{(k)^\top} \left( \nabla_{a^{(k)}(x)} - \log f(x)_y \right) \]
    - compute gradient of hidden layer below (before activation)
      \[ \nabla_{a^{(k-1)}(x)} - \log f(x)_y \iff \left( \nabla_{h^{(k-1)}(x)} - \log f(x)_y \right) \odot [\ldots, g'(a^{(k-1)}(x)_j), \ldots] \odot m^{(k-1)} \]
**Topics:** test time classification

- At test time, we replace the masks by their expectation
  - this is simply the constant vector 0.5 if dropout probability is 0.5
  - for single hidden layer, can show this is equivalent to taking the geometric average of all neural networks, with all possible binary masks

- Can be combined with unsupervised pre-training

- Beats regular backpropagation on many datasets