Hidden Units

Sargur N. Srihari
srihari@cedar.buffalo.edu

Topics in Deep Feedforward Networks

- Overview

1. Example: Learning XOR
2. Gradient-Based Learning
3. Hidden Units
4. Architecture Design
5. Backpropagation and Other Differentiation
6. Historical Notes

Topics in Hidden Units

1. ReLU and their generalizations
2. Logistic sigmoid and Hyperbolic tangent
3. Other hidden units

Choice of hidden unit

- Previously discussed design choices for neural networks that are common to most parametric learning models trained with gradient optimization
- We now look at how to choose the type of hidden unit in the hidden layers of the model
- Design of hidden units is an active research area that does not have many definitive guiding theoretical principles

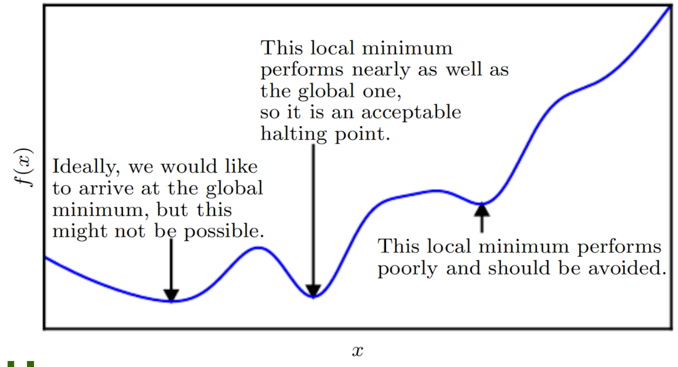
Choice of hidden unit

- ReLU is an excellent default choice
- But there are many other types of hidden units available
- When to use which kind (though ReLU is usually an acceptable choice)?
- We discuss motivations behind choice of hidden unit
  - Impossible to predict in advance which will work best
  - Design process is trial and error
    - Evaluate performance on a validation set

Is Differentiability necessary?

- Some hidden units are not differentiable at all input points
  - Rectified Linear Function $g(z)=\max\{0,z\}$ is not differentiable at $z=0$
- May seem like it invalidates for use in gradient-based learning
- In practice gradient descent still performs well enough for these models to be used in ML tasks

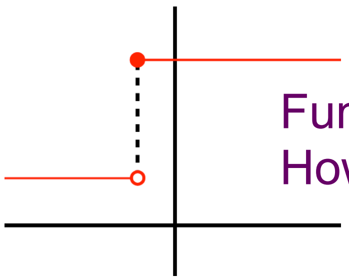
Differentiability ignored



- Neural network training
  - not usually arrives at a local minimum of cost function
  - Instead reduces value significantly
- Not expecting training to reach a point where gradient is 0,
  - Accept minima to correspond to points of undefined gradient
- Hidden units not differentiable are usually non-differentiable at only a small no. of points

Left and Right Differentiability

- A function $g(z)$ has a left derivative defined by the slope immediately to the left of z
- A right derivative defined by the slope of the function immediately to the right of z
- A function is differentiable at $z=a$ only if both
  - the left derivative $\boxed{\partial_+ f(a) := \lim_{\substack{x \to a+ \\ x \in I}} \frac{f(x)-f(a)}{x-a}}$ and
  - The right derivative $\boxed{\partial_- f(a) := \lim_{\substack{x \to a- \\ x \in I}} \frac{f(x)-f(a)}{x-a}}$ are equal



Function is not continuous: No derivative at marked point
However it has a right derivative at all points with $\delta_+ f(a)=0$ at all points

Software Reporting of Non-differentiability

- In the case of $\mathrm{g}(z)=\max\{0,z\}$, the left derivative at $z=0$ is 0 and right derivative is 1
- Software implementations of neural network training usually return:
  - one of the one-sided derivatives rather than reporting that derivative is undefined or an error
    - Justified in that gradient-based optimization is subject to numerical anyway
    - When a function is asked to evaluate $g(0)$, it is very unlikely that the underlying value was truly 0, instead it was a small value ε that was rounded to 0

What a Hidden unit does

- Accepts a vector of inputs $\boldsymbol{x}$ and computes an affine transformation $\boldsymbol{z} = W^T\boldsymbol{x}+\boldsymbol{b}$
- Computes an element-wise non-linear function $g(\boldsymbol{z})$
- Most hidden units are distinguished from each other by the choice of activation function $g(\boldsymbol{z})$
  - We look at: ReLU, Sigmoid and tanh, and other hidden units

Rectified Linear Unit & Generalizations

- Rectified linear units use the activation function $g(z) = \max\{0, z\}$
  - They are easy to optimize due to similarity with linear units
    - Only difference with linear units that they output 0 across half its domain
    - Derivative is 1 everywhere that the unit is active
    - Thus gradient direction is far more useful than with activation functions with second-order effects

Use of ReLU

- Usually used on top of an affine transformation

$$\boldsymbol{h}=g(W^T\boldsymbol{x}+\boldsymbol{b})$$

- Good practice to set all elements of $\boldsymbol{b}$ to a small value such as 0.1
  - This makes it likely that ReLU will be initially active for most training samples and allow derivatives to pass through

Generalizations of ReLU

- Perform comparably to ReLU and occasionally perform better
- ReLU cannot learn on examples for which the activation is zero.
- Generalizations guarantee that they receive gradient everywhere

Three generalizations of ReLU

- Three methods based on using a non-zero slope α_i when $z_i<0$:

$$h_i = g(\boldsymbol{z},\alpha)_i = \max(0,z_i) + \alpha_i \min(0,z_i)$$

1. Absolute-value rectification:
   - fixes α_i=-1 to obtain $g(z)=|z|$
2. Leaky ReLU:
   - fixes α_i to a small value like 0.01
3. Parametric ReLU or PReLU:
   - treats α_i as a parameter

Maxout Units

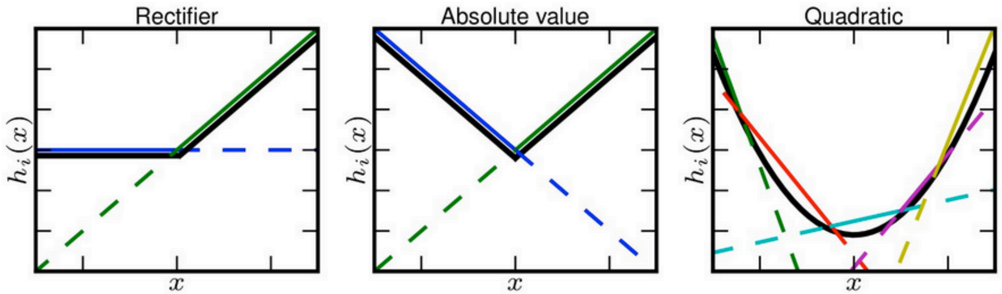
- Maxout units further generalize ReLUs
- Instead of applying element-wise function $g(z)$, maxout units divide z into groups of k values
- Each maxout unit then outputs the maximum element of one of these groups:

  $\mathrm{g(z)}_i = \max_{j \in \mathrm{G}(i)} z_j$

  – where $\mathrm{G}(i)$ is the set of indices into the inputs for group i, $\{(i\text{-}1)k+1,..,ik\}$
- This provides a way of learning a piecewise linear function that responds to multiple directions in the input $\boldsymbol{x}$ space

Maxout as Learning Activation

- A maxout unit can learn piecewise linear, convex function with upto k pieces
  - Thus seen as learning the activation function itself rather than just the relationship between units
    - With large enough k, approximate any convex function



  - A maxout layer with two pieces can learn to implement the same function of the input $\boldsymbol{x}$ as a traditional layer using ReLU or its generalizations

Learning Dynamics of Maxout

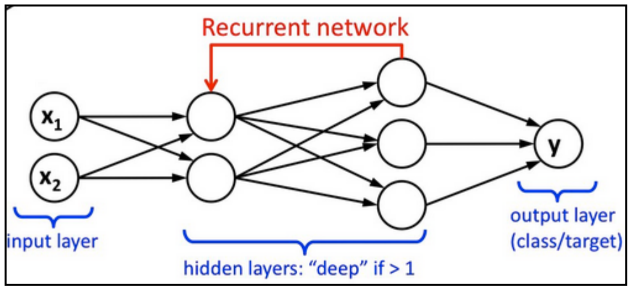
- Parameterized differently
- Learning dynamics different even in case of implementing same function of $\boldsymbol{x}$ as one of the other layer types
  - Each maxout unit parameterized by k weight vectors instead of one
    - So Requires more regularization than ReLU
    - Can work well without regularization if training set is large and no. of pieces per unit is kept low

Other benefits of maxout

- Can gain statistical and computational advantages by requiring fewer parameters
- If the features captured by n different linear filters can be summarized without losing information by taking max over each group of k features, then next layer can get by with k times fewer weights
- Because of multiple filters, their redundancy helps them avoid *catastrophic forgetting*
  - Where network forgets how to perform tasks they were trained to perform

Principle of Linearity

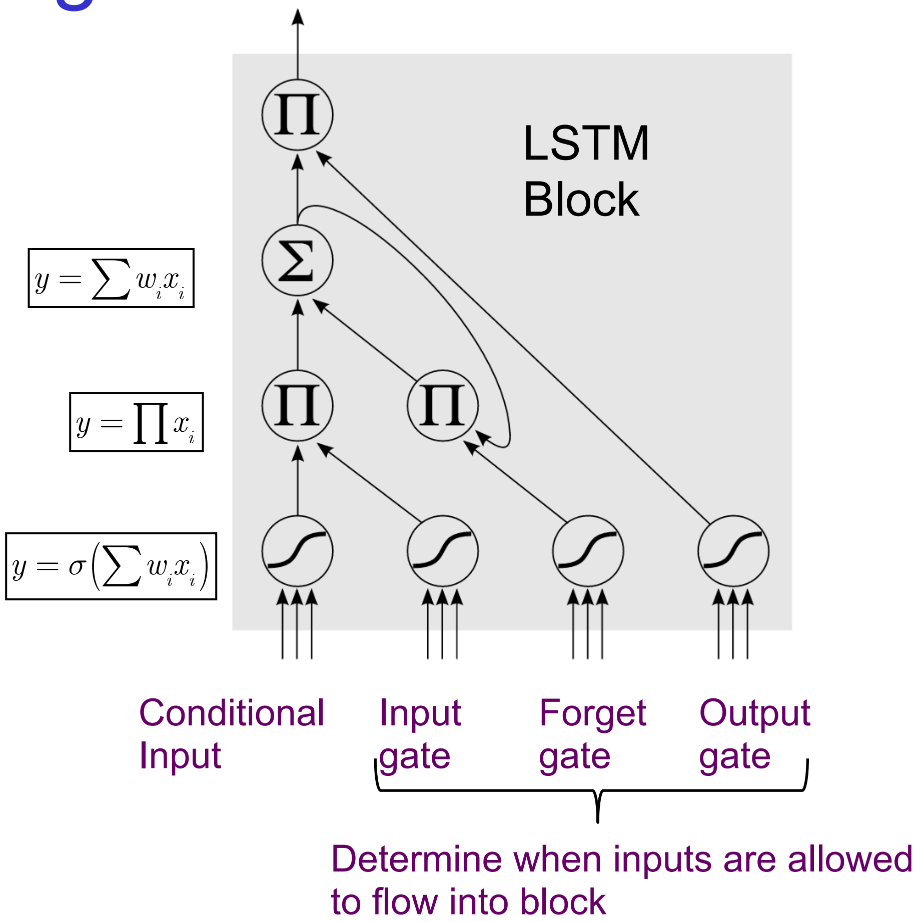
- ReLU based on principle that models are easier to optimize if behavior closer to linear
  – Principle applies besides deep linear networks
    - Recurrent networks can learn from sequences and produce a sequence of states and outputs



    - When training them need to propagate information through several steps
      – Which is much easier when some linear computations (with some directional derivatives being of magnitude near 1) are involves

Linearity in LSTM

- LSTM: best performing recurrent architecture
  - Propagates information through time via summation
- A straightforward kind of linear activation



LSTM: an ANN that contains LSTM blocks in addition to regular network units

Input gate: when its output is close to zero, it zeros the input

Forget gate: when close to zero block forgets whatever value it was remembering

Output gate: when unit should output its value

Logistic Sigmoid

- Prior to introduction of ReLU, most neural networks used logistic sigmoid activation

$$g(z)=\sigma(z)$$

- Or the hyperbolic tangent

$$g(z)=\tanh(z)$$

- These activation functions are closely related because

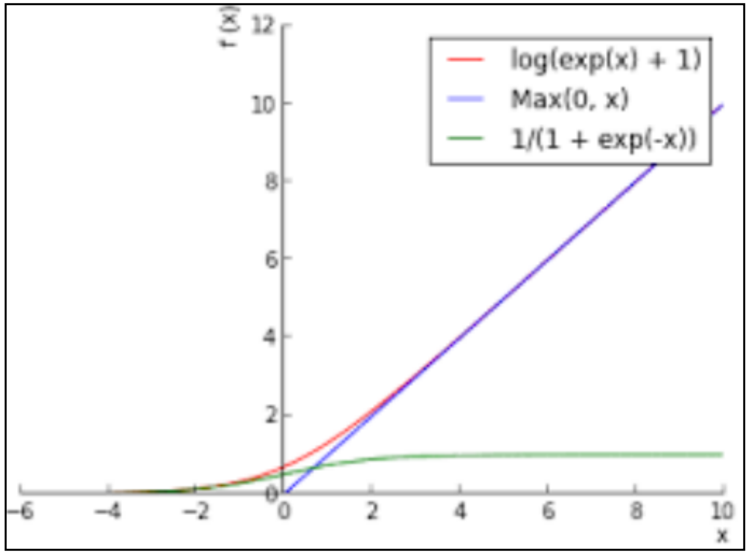
$$\tanh(z)=2\sigma(2z)\text{-}1$$

- Sigmoid units are used to predict probability that a binary variable is 1

Sigmoid Saturation

- Sigmoidals saturate across most of domain
  - Saturate to 1 when z is very positive and 0 when z is very negative
  - Strongly sensitive to input when z is near 0
  - Saturation makes gradient-learning difficult
- ReLU and Softplus increase for $\text{input} > 0$

Sigmoid can still be used
When cost function undoes the
Sigmoid in the output layer



Sigmoid vs $\tanh$ Activation

- Hyperbolic tangent typically performs better than logistic sigmoid
- It resembles the identity function more closely

  $\tanh(0)=0$ while $\sigma(0)=½$

- Because $\tanh$ is similar to identity near 0, training a deep neural network $\boxed{\hat{y} = \boldsymbol{w}^T \tanh\left(U^T \tanh\left(V^T \boldsymbol{x}\right)\right)}$ resembles training a linear model $\boxed{\hat{y} = \boldsymbol{w}^T U^T V^T \boldsymbol{x}}$ so long as the activations can be kept small

Sigmoidal units still useful

- Sigmoidal more common in settings other than feed-forward networks
- Recurrent networks, many probabilistic models and autoencoders have additional requirements that rule out piecewise linear activation functions
- They make sigmoid units appealing despite saturation

Other Hidden Units

- Many other types of hidden units possible, but used less frequently
  - Feed-forward network using $\boldsymbol{h} = \cos(\mathrm{W}\boldsymbol{x} + \boldsymbol{b})$
    - on MNIST obtained error rate of less than 1%
  - Radial Basis $\boxed{h_i = \exp\left(-\frac{1}{\sigma^2} \|W_{:,i} - \boldsymbol{x}\|^2\right)}$
    - Becomes more active as $\boldsymbol{x}$ approaches a template $\mathrm{W}_{:,i}$
  - Softplus $\boxed{g(a) = \zeta(a) = \log(1 + e^a)}$
    - Smooth version of the rectifier
  - Hard $\tanh$
    - Shaped similar to tanh and the rectifier but it is bounded

$$\boxed{g(a) = \max(-1, \min(1, a))}$$