Notes: Neural Network Methods for Natural Language Processing – Part 1 Supervised Classification and Feed-forward Neural Networks

Yingbo Li

01/20/2021

Table of Contents

Three most common types of NNs in NLP

- Feed-forward networks, i.e., multi-layer perceptrons (MLPs), or fully connected layers
  - Allow to work with fixed sized inputs
  - Or with variable length inputs in which we can disregard the order of the elements (continuous bags of words)
- Recurrent neural networks (RNNs)
  - Specialized models for sequential data
  - Produce a fixed size vector that summarizes the sequence
  - Doesn't require fixed sized input (e.g., lengths of input sequence can vary)
- Convolutional feed-forward networks (CNNs)
  - Good at extracting local patterns in the data

About neural networks

- Some of the neural network techniques (e.g., MLP) are simple generalizations of the linear models and can be used as almost drop-in replacements for the linear classifiers
- RNNs and CNNs are rarely used as standalone components. They are used to extract features and being fed into other network components, and trained to work in tandem with them.

Success stories

- Multi-layer feed-forward networks can provide competitive results on sentiment classification and factoid question answering
- Networks with convolutional and pooling layers are useful for classification tasks in which we expect to find strong local clues regarding class membership, but these clues can appear in different places in the input
  - Convolutional and pooling layers allow the model to learn to find such local indicators, regardless of their position

Note

- **In this book, vectors are assumed to be row vectors**

One-hot and dense vector representations (p23)

- The input $\mathbf{x}$ in language classification example contains the normalized bigram counts in the document D
- $D_{[i]}$ is the bigram at document position i
- Each vector $\mathbf{x}^{D_{[i]}} \in \mathbb{R}^d$ is a one-hot vector
- The following $\mathbf{x}$ is an averaged bag of words, or just bag of words:

$$\mathbf{x} = \frac{1}{|D|} \sum_{i=1}^{|D|} \mathbf{x}^{D_{[i]}}$$

- Bag of words doesn't consider orders among words

Minibatch stochastic gradient descent (SGD) algorithm (p32)

- Goal: set the parameters Θ to minimize the total loss

$$\mathcal{L}(\Theta) = \sum_{i=1}^{n} L\left(f(\mathbf{x}_i; \theta), \mathbf{y}_i\right)$$

  over the training set
- Learning rate: η_t
- Minibatch size m, can vary from $m = 1$ to $m = n$
- After the inner loop, $\hat{\mathbf{g}}$ contains the gradient estimate

Algorithm 2.2 Minibatch stochastic gradient descent training.

Input:

- Function $f(\boldsymbol{x}; \Theta)$ parameterized with parameters Θ.
- Training set of inputs $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$ and desired outputs $\boldsymbol{y_1}, \ldots, \boldsymbol{y_n}$.
- Loss function L.

---

while stopping criteria not met **do**
  Sample a minibatch of m examples $\{(\boldsymbol{x_1}, \boldsymbol{y_1}), \ldots, (\boldsymbol{x_m}, \boldsymbol{y_m})\}$
  $\hat{\boldsymbol{g}} \leftarrow 0$
  for $i = 1$ to m **do**
    Compute the loss $L(f(\boldsymbol{x_i}; \Theta), \boldsymbol{y_i})$
    $\hat{\boldsymbol{g}} \leftarrow \hat{\boldsymbol{g}} +$ gradients of $\frac{1}{m} L(f(\boldsymbol{x_i}; \Theta), \boldsymbol{y_i})$ w.r.t Θ
  $\Theta \leftarrow \Theta - \eta_t \hat{\boldsymbol{g}}$
return Θ

A feed-forward neural network with one hidden-layer

$$\mathrm{NN}_{\mathrm{MLP1}}(\boldsymbol{x}) = g(\boldsymbol{x}\boldsymbol{W}^{\mathbf{1}} + \boldsymbol{b}^{\mathbf{1}})\boldsymbol{W}^{\mathbf{2}} + \boldsymbol{b}^{\mathbf{2}} \tag{4.2}$$

$$\boldsymbol{x} \in \mathbb{R}^{d_{in}}, \quad \boldsymbol{W}^{\mathbf{1}} \in \mathbb{R}^{d_{in} \times d_1}, \quad \boldsymbol{b}^{\mathbf{1}} \in \mathbb{R}^{d_1}, \quad \boldsymbol{W}^{\mathbf{2}} \in \mathbb{R}^{d_1 \times d_2}, \quad \boldsymbol{b}^{\mathbf{2}} \in \mathbb{R}^{d_2}.$$

- g is a nonlinear function
- The first layer transforms the data into a good representation, while the second layer applies a linear classifier to that representation
- Layers resulting from linear transformations are called fully connected, or affline

Common nonlinearities (p45)

- Sigmoid (currently considered to be deprecated for use in internal layers of NN)
- Hyperbolic tangent (tanh): common
- ReLU: common

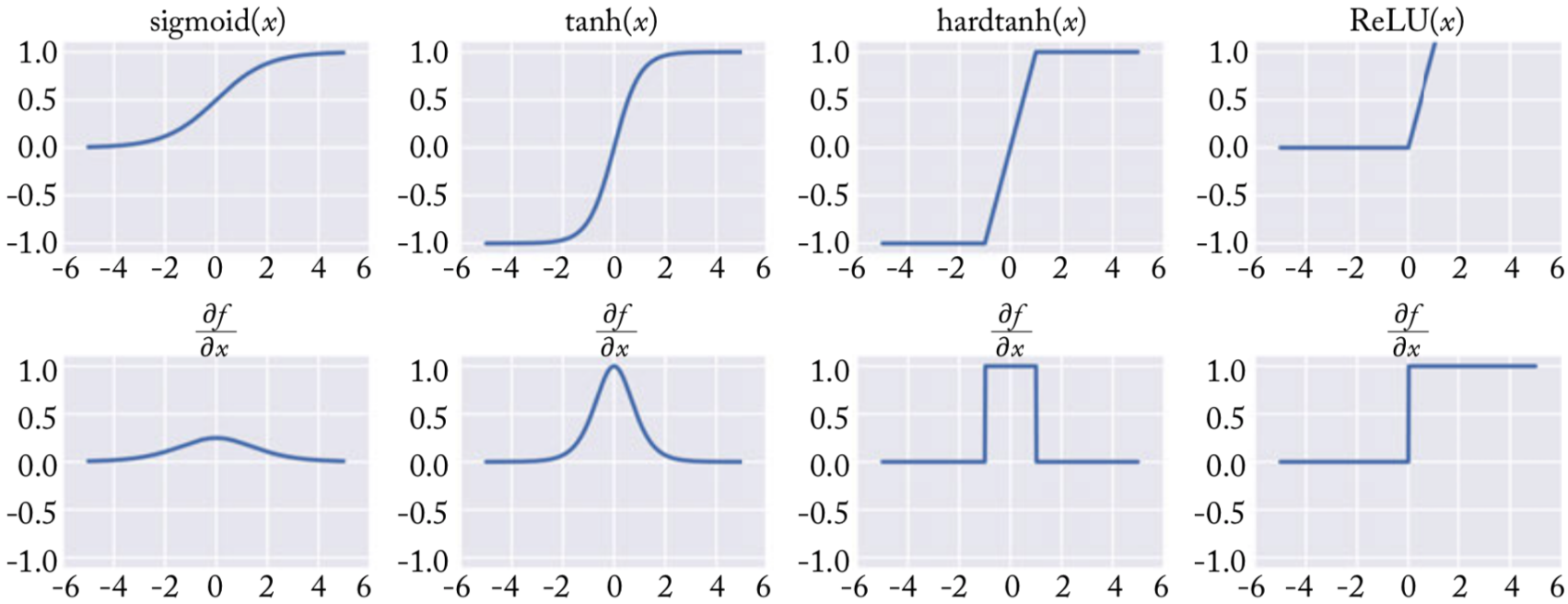

Figure 4.3: Activation functions (top) and their derivatives (bottom).

Regularization and dropout

- L2 regularization, also called weight decay is effective, and tuning the regularization strength λ is advisable
- Dropout training: randomly dropping (setting to zero) half of the neurons in the network in each training example in the stochastic-gradient training
- Dropout is effective in NLP applications of NNs

Comment

- The objective function for nonlinear neural networks is not convex, and gradient-based methods may get stuck in a local minima
- Still, gradient-based methods produce good results in practice
- Choice of optimization algorithm: the book author likes Adam, as it is very effective and relatively robust to the choice of learning rate

Initialization

- It is advised to run several restarts of the training starting at different random initializations, and choosing the best one based on a development set
- Xavier initialization:

first name, suggests initializing a weight matrix $\boldsymbol{W} \in \mathbb{R}^{d_{in} \times d_{out}}$ as:

$$\boldsymbol{W} \sim U\left[-\frac{\sqrt{6}}{\sqrt{d_{in}+d_{out}}}, +\frac{\sqrt{6}}{\sqrt{d_{in}+d_{out}}}\right], \tag{5.1}$$

where $U[a,b]$ is a uniformly sampled random value in the range $[a,b]$. The suggestion is based

- When using ReLU nonlinearities, the following Gaussian initialization may work better than Xavier. The weights should be initialized by sampling from a zero-mean Gaussian distribution whose sd is $\sqrt{2/d_{\text{in}}}$

Vanishing and exploding gradients

- Vanishing gradients
  - batch-normalization, i.e., for every minibatch, normalize the inputs to each of the network layers to have zero mean and unit variance
  - Or use specialized architectures that are designed to assist in gradient flow (i.e., LSTM and GRU)
- Exploding gradients: clipping the gradients if their norm exceeds a given threshold

Learning rate

- Experiment with a range of initial learning rates in range $[0, 1]$, e.g., 0.001, 0.01, 0.1, 1.
- Learning rate scheduling decreases the rate as a function of the number of observed minibatches
  - A common schedule is dividing the initial learning rate by the iteration number
  - Bottou's recommendation:

$$\eta_t = \frac{\eta_0}{1 + \eta_0 \lambda t}$$

References

- Goldberg, Yoav. (2017). Neural Network Methods for Natural Language Processing, Morgan & Claypool