Practical issues for multilayer perceptrons

CS 550: Machine Learning

Practical issues for MLPs

- So far, we have not considered practical issues for the sake of simplicity.
  - This can lead to unsatisfactory results such as
    - very slow convergence or
    - poor performance

- The practical suggestions have been
  - based on plausible heuristics, and
  - found to be useful in many practical applications

- More in Section 6.8 of Duda-Hart-Stork’s book
Online, stochastic or batch training?

- **Stochastic training is preferred** for most applications
  - Especially when datasets are highly redundant
  - It is typically faster than batch training

- **Batch training** allows using some second-order techniques that
  - Cannot be easily incorporated into stochastic learning

- **Online training** is rarely used in practice
  - When the amount of data is so large or
  - When memory costs are so high

When to stop

- When change in the criterion function is smaller than some preset value $\theta$
  $$\|\nabla E(v, w)\| \leq \theta$$

- When a minimum is reached on the validation set

  - **Training error** ultimately reaches an asymptotic value
  - **The error on an independent test set** is virtually always higher
    - While it generally decreases, it can increase or oscillate.
Activation function

- A number of properties we seek for activation function $f(.)$
  - $f(.)$ must be **nonlinear**
    - Otherwise 3-layer networks provide no further computational power
  - $f(.)$ must be **saturate**: it has some max and min output values
    - This will keep the weights and activations bounded and
      - Keep the training time limited
    - It is also desirable for classification
      - when the output is represented by probabilities
    - It may not be desirable in networks used for regression
  - $f(.)$ and $f'(.)$ must be **defined throughout the range of their argument**
    - Backpropagation works when $f(.)$ is continuous and its derivative is met
  - $f(.)$ should be **linear for small values of net activation**
    - This will enable the system to implement a linear model if it is adequate for low error

One class of such functions is sigmoid

- Logarithmic sigmoid function
  
  $f(x) = \frac{1}{1 + \exp(-x)}$

- Hyperbolic tangent sigmoid function
  
  $f(x) = \alpha \tanh(\beta x) = \alpha \frac{\exp(\beta x) - \exp(-\beta x)}{\exp(\beta x) + \exp(-\beta x)}$
Scaling input

- A neural network may prefer some features over the others
  - When the orders of magnitudes of features are different
    - e.g., in fish classification, “mass” is measured in grams and “length” is measured in meters
  - The neural network adjust weights in favor of features with smaller magnitudes
    - e.g., in fish classification, “mass” has larger effects than “length”
      - If “mass” is measured in kilograms and “length” is measured in millimeters, the situation will be reversed
- We normalize the training samples to prevent this problem
  - Samples are shifted so that the average of each feature is 0.0
  - Dataset is normalized so that the variance in each feature is 1.0
- Test samples must be standardized with the same transformation

When the training set is unbalanced

- MLPs may have difficulties in learning the minority class
  - They may favor the majority class
- A common practice for dealing with this is to rebalance such a training set artificially
  - Oversampling: replicate training samples from the minority class(es)
  - Undersampling: ignore training samples from the majority class(es)
When the training set is too small

- Training with noise
  - Virtual training samples can be generated
    - They can be used as if they were normal training samples
  - In the absence of problem-specific information
    - Virtual samples should be generated by
      - Adding \(d\)-dimensional Gaussian noise to true training samples
    - In classification,
      - Noise is added to inputs and class labels should be left unchanged
    - In regression,
      - Noise could be added to both inputs and outputs
  - This method generally does not improve accuracy
    - For highly local techniques
      - such as the nearest neighbor method

- Manufacturing data
  - If we have knowledge about the sources of variation among samples
    - We can “manufacture” training samples that convey more information than uncorrelated noise
  - For example, in optical character recognition, we manufacture data by
    - Rotating the images of training samples
    - Performing simple image processing on the images
      - e.g., to simulate a bold face character
  - Disadvantage
    - Memory requirements may be large
    - Overall training may be slow
Target values

- In classification, a target value can be represented by
  - 0/1 target values
    - Outputs represent posterior probabilities
      - Using softmax function
        » maximum output is transformed to 1.0
        » all others reduced to 0.0
  - -1/+1 target values
    - Outputs do not represent posterior properties

- A proper activation function should be used in the output layer
  - Depending on the selected target value representation

Network topology

- The number of hidden units
  - It controls the expressive power of the network
    - Thus, the complexity of the decision boundary

  - There is no foolproof method to set the number of hidden units before training
    - If samples are well-separated
      - few hidden units are enough
    - If samples have complicated densities
      - more hidden units may be necessary
Network topology

- If too much hidden units,
  - The network is tuned to the particular dataset (overfitting)
    - Training error can become small, but test error is unacceptably high
- If too few hidden units,
  - The network does not have enough free parameters to fit the training data well
    - Training and test errors are high

Network topology

- The number of hidden layers
  - Three layers are enough to implement any arbitrary function
    - Thus the use of more than three layers is only recommended if there are special problem conditions and requirements
  - For example, in optical character recognition,
    - It is desirable to have systems that are invariant with respect to transformations such as translation and rotation
    - It may be easier to learn these transformations with a four-layer neural network
      - Each layer learns a different invariance within a limited range of parameters
      - Multiple layers are stacked to allow the full neural network to learn the full invariance task
Initializing weights

- We cannot initialize the weights to zero
- We want to have uniform learning
  - All weights reach their final equilibrium at about the same time
  - For that, with standardized data, we choose weights randomly from a uniform distribution \( \hat{w} < w < \hat{w} \)
  - If \( \hat{w} \) is chosen too small
    - The net activation of a hidden unit will be small
  - If \( \hat{w} \) is chosen too large
    - The hidden unit may saturate even before learning begins
  - We set \( \hat{w} \) such that the net activation function is in its linear range

Learning rates

- In principle, if the learning rate is small enough to ensure convergence
  - Its value determines only the speed
  - Not the final weight values

- In practice, the learning rate can indeed affect the quality of the final network
  - Since networks are not fully trained most of the time
Learning rates

- The optimal learning rate leads to the local minimum in one step
- The optimal rate is found as
  \[
  \eta_{opt} = \left( \frac{\partial^2 E}{\partial w^2} \right)^{-1}
  \]
- The system converges for \( \eta < \eta_{opt} \) and \( \eta_{opt} < \eta < 2 \eta_{opt} \)
  - But the training is needlessly slow
- It is found that the system diverge if \( \eta > 2 \eta_{opt} \)

Learning rates

- Thus, in order to have rapid and uniform learning
  - For each weight
    - Calculate \( \frac{\partial^2 E}{\partial w^2} \) and
    - Set the optimal learning rate separately
  - For typical networks that use sigmoids
    - It is found that \( \eta \approx 0.1 \) is adequate as a first choice
      - \( \eta \) should be lowered if the criterion function diverges
      - \( \eta \) should be raised if learning seems unduly slow
  - During training, you may also change \( \eta \) as a function of time
Momentum

- Error surfaces often have plateaus
  - regions in which $\frac{\partial E}{\partial w}$ is very small
  - can usually arise when
    - too many weights such that
      - the error only weakly depends on any of them

- Momentum allows to learn more quickly
  - When there are such plateaus

---

Momentum

- In stochastic learning, we include some fraction of the previous weight update into the learning rule
  \[ w^{(t+1)} = w^{(t)} + (1 - \alpha) \Delta w^{(t)} + \alpha \Delta w^{(t-1)} \]
  - Parameter $\alpha$ should be nonnegative and less than 1
  - If $\alpha = 0$, it is the same as standard backpropagation
  - If $\alpha = 1$, the weight vector moves with constant velocity
  - Values typically used are $\alpha \approx 0.9$

- The use of momentum increases stability
  - Thus, it can speed the learning process