DM825

Introduction to Machine Learning

Lecture 6
Training Neural Networks

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Outline

Training

Deterministic approach: minimize error function:

$$Err(\vec{\theta}) = \frac{1}{2}\sum_{i=1}^{m} \parallel h(\vec{x}_i, \vec{\theta}) - \vec{y}_i \parallel^2$$

Probabilistic approach

- Regression
  assume one single out put y and Gaussian distributed with mean dependent on the NN output:

  $$p(y \mid \vec{x}, \vec{\theta}) = \mathcal{N}(y \mid h(\vec{x}, \vec{\theta}), \beta^1)$$

  assume h to be $I(\cdot)$
  $(\vec{x}, \vec{y}) = \{(\vec{x}^1, y^1) \dots (\vec{x}^m, y^m)\}$
  likelihood $\mathcal{L}(\vec{\theta}) = p(\vec{y}|\vec{x}, \vec{\theta}, \beta) = \prod_{i=1}^{m} p(y^i \mid \vec{x}^i, \vec{\theta}, \beta)$

  $$-\log \mathcal{L}(\vec{\theta}) = \frac{\beta}{2}\sum_{i=1}^{m}\{h(\vec{x}^i, \vec{\theta}) - y^i\}^2 - \frac{m}{2}\ln\beta + \frac{\alpha}{2}\ln(2\pi)$$

  We minimize in $\vec{\theta}$ and β

find $\vec{\theta}_{ML}$ minimizing: $E(\vec{\theta}) = \frac{\beta}{2}\sum_{i=1}^{m}\{h(\vec{x}^i, \vec{\theta}) - y^i\}^2$ (nonconvex)
find β_{ML} substituting $\vec{\theta}_{ML}$ in $\frac{1}{\beta_{ML}} = \sum_{i=1}^{m}\frac{1}{m}\{h(\vec{x}^i, \vec{\theta}) - y^i\}^2$
If multiple independent outputs, ie, $\vec{y}$: $E(\vec{\theta}) = \frac{\beta}{2}\sum_{i=1}^{m}||h(\vec{x}^i, \vec{\theta}) - y^i||^2$
Note that for $\hat{y}_k = a_k$: $\frac{\partial E}{\partial a_k} = \hat{y}_k - y_k$

- Binary classification

Back Propagation Algorithm

Goal: finding efficient technique to evaluate the gradient of $E(\vec{\theta})$, ie, computing derivatives of $E(\vec{\theta})$

Assumption on activation functions: arbitrary differentiable (eg, sigmoidal hidden units)

$$E(\vec{\theta}) = \sum_{i=1}^{m} E_i(\vec{\theta})$$

we evaluate $\nabla_i E(\vec{\theta})$ (accumulated in the batch case).

For a simple linear model $y_k = \sum_j \theta_{jk} x_j$

Let $\hat{y}_k$ be the output at the kth node of output units thus $\hat{y}_k = h(\vec{x}, \vec{\theta})$

$$E_i = \frac{1}{2} \sum_k (\hat{y}_k^i - y_k^i)^2$$

$$\frac{\partial E_i}{\partial \theta_{jk}} = (\hat{y}_{jk}^i - y_{jk}^i) x_{jk}^i$$

With more layers

each unit computes:

$$a_l = \sum_j \theta_{jl} z_j$$

$$z_l = f(a_l)$$ non linear activation function

For chain rule of partial derivative:

$$\frac{\partial E}{\partial \theta_{jl}} = \frac{\partial E}{\partial a_l} \frac{\partial a_l}{\partial \theta_{jl}}$$

that is, the error depends on θ_{jl} only via a_l

We define the errors:

$$\delta_l \equiv \frac{\partial E}{\partial a_l}$$

thus

$$\boxed{\frac{\partial E}{\partial \theta_{jl}} = \delta_l z_j}$$

which resembles $(\hat{y}^i_{jk} - y^i_{jk}) x^i_{jk}$.
δ_l from head node and z_j output from tail node

How do we calculate δ?

$\delta_k = \hat{y} - y_k$ at the output, saw earlier

At other nodes: variations in a_l have effect in E via a_k

$$\delta_l = \frac{\partial E}{\partial a_l} = \sum \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial a_l}$$

$$a_k = \sum_k \theta_{lk} z_l$$

$$z_l = f(a_l)$$

$$\boxed{\delta_l = f'(a_l) \sum_k \theta_{lk} \delta_k}$$

backward propagation formula

Backward Propagation Algorithm

1. for an observation i, apply forward propagation to $\vec{x}^i$ to find activations

$$a_l = \sum_j \theta_{jl} z_j$$

$$z_l = f(a_l)$$

non linear activation function

2. evaluate δ_k $\forall$ output units
3. backpropagate δs to obtain δ_l for each hidden unit
4. calculate derivatives

$$\frac{\partial E^i}{\partial \theta_{jl}} = \delta_l z_j$$

5. apply update rule:

$$\theta_{jl}^{t+1} \equiv \theta_{jl}^t - \alpha \nabla E(\vec{\theta}^t) = \theta_{jl}^t - \alpha \frac{\partial E^i(\vec{\theta}^t)}{\partial \theta_{jl}}$$

- if batch implementation then

$$\frac{\partial E(\vec{\theta}^t)}{\partial \theta_{jl}} = \sum_{i=1}^{k} \frac{\partial E^i(\vec{\theta}^t)}{\partial \theta_{jl}}$$

- computation time: $O(|E|)$ for forward propagation and $O(|E|)$ for backward propagation
- alternative approach: numerical differentiation: it can be used if $f'(a)$ is not known and to verify implementation.

$$\frac{\partial E(\vec{\theta}^t)}{\partial \theta_{jl}} = \frac{E^i(\theta_{jl} + \epsilon) - E^i(\theta_{jl})}{\epsilon}, \qquad \epsilon \ll 1$$

but this takes $O(|E|^2)$

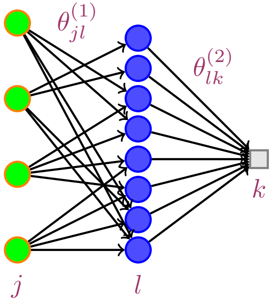
The number of hidden nodes, M, is a parameter to tune via validation. To avoid overfitting:

- regularized error

$$\tilde{E}(\vec{\theta}) = E(\vec{\theta}) + \frac{\lambda}{2}\vec{\theta}^T\vec{\theta}$$

- early stopping in gradient descent: use validation set to decide when to stop

Example



output unit: linear activation function:
$y_k = a_k$

hidden units:

$$h(a) = \tanh(a)$$
$$=$$