Gradient Descent Learning

Christian Herta

Oktober, 2013

Outline

Supervised Learning

training data: $\mathcal{D} = \{(\vec{x}^{(1)}, \vec{y}^{(1)}), (\vec{x}^{(2)}, \vec{y}^{(2)}), \ldots, (\vec{x}^{(m)}, \vec{y}^{(m)})\}$

- m training data with n features: $1 \leq j \leq n$
  - Input-Features $\vec{x}^{(i)}$ of the i-th training example $\vec{x}^{(i)} = x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}$
  - i-th target values of the i-th example $\vec{y}^{(i)}$
- $x_j^{(i)}$: values of the feature j of the i-th training example
- Goal: prediction of $\vec{y}$ for a new $\vec{x}$.

Hypothesis (prediction function)

Parametric Model:

$$\vec{h}_{\vec{\Theta}}(\vec{x}) = \vec{h}(\Theta, \vec{x})$$

with

- Parameters Θ
- $\vec{h}$ is a prediction of the $\vec{y}$ for given $\vec{x}$

The parameters have to be adapt by training to make "good" predictions.

Loss

Discrepancy between the desired output $\vec{y}^{(i)}$ and the output of the system $\vec{h}_\Theta(\vec{y}^{(i)})$ for fixed Θ is measured by a loss function:

$$loss(\vec{\Theta}) = loss(h_\Theta(\vec{x}^{(i)}), \vec{y}^{(i)})$$

The total cost of all training examples is given by the mean of the losses:

$$J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} loss(\vec{h}_\Theta(\vec{x}^{(i)}), \vec{y}^{(i)})$$

Usually the loss and the cost (also called error) is considered as a function of the parameters Θ

Example for Losses

- for regression: squared error loss
- for classification: cross entropy loss

Outline

Problem

- Hypothesis: $h_\Theta(\vec{x})$
- k Parameter: Θ
- **Minimization of the cost function $J(\Theta)$**

Gradient Descent for minimization of the cost function J

Repeat until convergence is reached:

$$\Theta_j \leftarrow \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta)$$

- α scalar (or more complex (e.g. approx of inverse Hessian))
- Note for implementation: simultaneous *update* for all Θ_j

Stocastic Gradient Descent

$$\Theta_j \leftarrow \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J_{partial}(\Theta)$$

The cost is computed only for an example or some examples (mini-batch), e.g. randomly selected.

Vector representation of gradient descent

With the definition of the gradient:

$$grad(J(\Theta)) = \nabla J(\Theta) = \begin{pmatrix} \frac{\partial J(\Theta)}{\partial \Theta_0} \\ \frac{\partial J(\Theta)}{\partial \Theta_1} \\ \dots \\ \frac{\partial J(\Theta)}{\partial \Theta_n} \end{pmatrix}$$

$$\Theta^{new} \leftarrow \Theta^{old} - \alpha \cdot grad(J(\Theta^{old}))$$

Regularization

High "capacity" (model complexity) c with respect to the number of examples m $\rightarrow$ Overfitting

$$J_{test} - J_{train} = k(c/m)^{\lambda}$$

with

- $0.5 < \lambda < 1.$
- constant k
- J_{test}

Adding a regularization term to prevent overfitting (formalized in structural risk minimization) for limiting the capacity of the subset of the parameter space. Optimizing of an augmented error $J_{train} + \frac{\lambda}{m}\Omega(\Theta)$