DM825

Introduction to Machine Learning

Lecture 5

Perceptron and Neural Networks

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. Decision Theory

2. Perceptron

3. Multilayer Perceptron

Outline

1. Decision Theory

2. Perceptron

3. Multilayer Perceptron

Decision Theory

Optimal decision under uncertainty $\leftarrow$ probability theory + decision theory

Probability theory:

We infer the joint probability $p(\vec{y}, \vec{x})$ then we must decide on $\vec{y}$.

Example:

medical diagnosis:

$\mathcal{C}_1$	has cancer	$y = 1$
$\mathcal{C}_2$	has not cancer	$y = 0$

$p(\mathcal{C}_k, \vec{x})$ not enough to decide optimally.

Decision step

we derive $p(\mathcal{C}_k \mid \vec{x}) = \frac{p(\vec{x}|\mathcal{C}_k)p(\mathcal{C}_k}{p(\vec{x})}$ (all obtainable from $\mathcal{C}_k$, $p(\mathcal{C}_k, \vec{x})$)

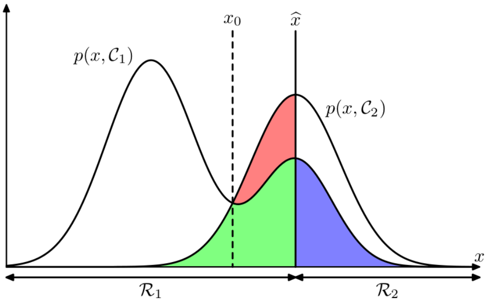
We want to minimize the probability of assigning to the wrong class. We show that the intuition of choosing the class with $p(\mathcal{C}_k, \vec{x})$ is right.

$\mathcal{R}_k$ decision regions of the input space: boundaries are called decision boundaries (or surfaces)

Mistake if $\mathcal{C}_1$ is true but it is predicted $\mathcal{C}_2$.

$$\begin{aligned} p(\text{mistake}) &= p(\vec{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\vec{x} \in \mathcal{R}_2, \mathcal{C}_1) \\ &= \int_{\mathcal{R}_1} p(\vec{x}, \mathcal{C}_2) dx + \int_{\mathcal{R}_2} p(\vec{x}, \mathcal{C}_1) dx \end{aligned}$$

since $p(\mathcal{C}_k, \vec{x}) = p(\mathcal{C}_k \mid \vec{x}) p(\vec{x})$ and $p(\vec{x})$ is common, the solution that minimizes is the one that assigns each $\vec{x}$ to $\mathcal{R}_k$ with largest $p(\mathcal{C}_k \mid \vec{x})$.



Minimizing Expected Loss

L_{kj} true class $\mathcal{C}_k$, predicted $\mathcal{C}_j$

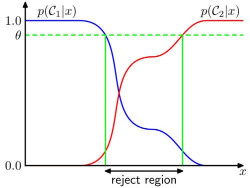
$$L = \begin{array}{ccc} & \text{cancer} & \text{normal} \\ C & 0 & 1000 \\ N & 1 & 0 \end{array}$$

The true class is unknown. We inferred $p(\vec{x}, \mathcal{C}_k)$, we want to choose the boundary regions that minimize expected loss:

$$E[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\vec{x}, \mathcal{C}_k) d\vec{x}$$

For each $\vec{x}$we minimize $\sum_k L_{kj} p(\vec{x}, \mathcal{C}_k)$ that is for each $\vec{x}$we choose j such that $\sum_k L_{kj} p(\mathcal{C}_k \mid \vec{x})$ is minimum.

Possible to introduce a threshold θ and reject those inputs $\vec{x}$for which the largest of the posterior probabilities $p(\mathcal{C}_k \mid \vec{x})$ is less than or equal to θ



Generative vs Discriminative Approaches

Discriminative Approaches:

- construct a discriminative function that directly assigns each vector $\vec{x}$ to a specific class.
- determine conditional probability $p(y \mid \vec{x}, \theta)$ or $p(\mathcal{C}_k \mid \vec{x}, \theta)$ by parameterzing and then determine parameters via MaxLikelihood.

It learns a decision boundary in the space of inputs, then maps a new input to the response.

Generative Approaches: determine $p(\vec{x} \mid \mathcal{C}_k)$ and $P(\mathcal{C}_k)$ and then compute $p(\mathcal{C}_k \mid \vec{x})$ via Bayes rule
It learns the distribution of the class features, then assign new input according to the class that gives highest probability

Resume

- $\vec{x}$ feature vector
- $\vec{\phi}(\vec{x})$ non linear transformation of $\vec{x}$
- $h(\vec{x}) = f(\vec{\theta}^T \cdot \vec{\phi}(\vec{x}))$ generalized linear model
- under the first discriminative approach $f(\cdot)$ must map to one of the responses

$f(\cdot)$:

- in regression was $I(\cdot)$
- in classification
  - two-classes $\rightsquigarrow$ $\vec{y} \in \{0, 1\}$, logistic
  - k-classes $\rightsquigarrow$ 1-of-k, y is vector of size k, softmax

$f(\cdot)$ is called activation function in ML and its inverse link function in statistics

- In linear regression, since $f \equiv I$ then $f(\vec{\theta}, \vec{\phi}(\vec{x}))$ is linear in $\vec{\theta}$ and in the simplest case also in $\vec{x}$

- In classification, f is non linear in $\vec{\theta}$.
  The decision boundaries are described by $h_{\vec{\theta}}(\vec{x}) = \mathrm{const}$ hence if $\vec{\theta}^T \cdot \vec{x}$ is a linear function then $\vec{\theta}^T \cdot \vec{x} = \mathrm{const}$ and the boundary is linear in $\vec{x}$ (or in any case it is linear in $\vec{\phi}(\vec{x})$)

Outline

1. Decision Theory

2. Perceptron

3. Multilayer Perceptron

Perceptron Algorithm

Classification problem.
$h_{\vec{\theta}}(\vec{x}) = f(\vec{\theta}^T, \vec{\phi}(\vec{x}))$ and $\vec{\phi}(\vec{x})$ includes a bias component $\vec{\phi}_0$

$$f(a) = \begin{cases} +1 & a \geq 0 \\ -1 & a < 0 \end{cases}$$

How to determine $\vec{\theta}$?
Error function minimization:

- misclassification patterns:
  not good because piecewise constant function and gradient methods do not work.

- perceptron criterion:

  we seek $\vec{\theta}$ such that $\begin{array}{ll} \vec{\theta}^T \cdot \vec{\phi}(\vec{x})^i \geq 0 & \text{if } \vec{x}^i \in \mathcal{C}_1 \\ \vec{\theta}^T \cdot \vec{\phi}(\vec{x})^i < 0 & \text{if } \vec{x}^i \in \mathcal{C}_2 \end{array}$

  hence: $\vec{\theta}^T \cdot \vec{\phi}(\vec{x})^i \cdot y^i > 0$ if prediction correct and $\vec{\theta}^T \cdot \vec{\phi}(\vec{x})^i \cdot y^i < 0$ otherwise.
  hence: we minimize: $E_p(\vec{\theta}) = -\sum_{i \in M} \vec{\theta}^T \vec{\phi}(\vec{x})^i y^i$, M set of misclassified

We can use stochastic gradient function:

$$\vec{\theta}_j^{t+1} = \vec{\theta}^t - \alpha \nabla E_p(\vec{\theta}) = \vec{\theta}^t + \alpha \vec{\phi}(\vec{x})^i y^i$$

since $h(\vec{\theta} \cdot \vec{\phi}(\vec{x}))$ stays unchanged if all $\vec{\theta}$ are scaled then $\alpha = 1$ w.l.g.

[Demo]

Perceptron Convergence

Theorem

If the training data set is linearly separable, the perceptron learning algorithm is guaranteed to find an exact solution in a limited number of steps.

- maybe large number of iterations required
- may depend on the order in which data are presented
- for non linearly separable points the algorithm will never converge
- it does not provide probabilistic output
- it does not generalize to $k > 2$
- based on linear combination of basis functions

Minsky and Papert (1969) showed that perceptrons do not work on non linearly separable points.

Outline

1. Decision Theory

2. Perceptron

3. Multilayer Perceptron

Multilayer Perceptrons

We saw:

$$h(\vec{x}) = f\left(\sum_{j=1}^{p} \theta_j \phi_j(\vec{x})\right)$$

$f(\cdot)$ in general nonlinear activation function

These models comprised linear combinations of fixed basis functions:

+ analytical properties
– curse of dimensionality

Idea: fix the number of basis functions but allow them to adapt:

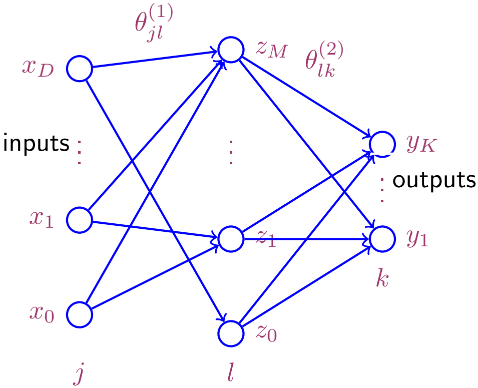
- let $\phi_j(\vec{x})$ depend on parameters
- adjust these parameters along with θ_j during training

Multilayer perceptron: multiple layers of logistic regression
The likelihood function is no longer a convex function of $\vec{\theta}$.

Neural Networks: McCulloch, Pitts (1943), Rosenblatt (1962)
perspective here: statistical pattern recognition
restrict to multilayer perceptrons

Implementation

Each basis function uses the same form, so each basis function is itself a non linear function of a linear combination of inputs.



First Layer

$$a_l = \sum_{i=1}^{D} \theta_{il}^1 x_i + \theta_{0l}^1$$

θ_{il}^1 weights

θ_{0l}^1 biases

a_l activations

Transformed by differentiable nonlinear activation functions $f(\cdot)$:

$z_l = f(a_l)$ output of the hidden units

linearly combined again to give activations to output unit

Second Layer

$$a_k = \sum_{l=1}^{M} \theta_{lk}^2 x_i + \theta_{0k}^2 \qquad k = 1 \ldots K \text{outputs}$$

transformed by activation function to give a set of network outputs $\hat{y}$:

- identity: $\hat{y}_k = a_k$
- multiple binary classification: $\hat{y_k} = \sigma(a_k) = \frac{1}{1+\exp(-a_k)}$ logistic sigmoid
- multiclass: $\hat{y_k} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$ softmax

Combining all together:

$$
\begin{aligned}
y_k &= h_k(\vec{x}, \vec{\theta}) \\
&= \sigma\left(\sum_{l=1}^{M} \theta^2_{lk} f\left(\sum_{i=1}^{D} \theta^1_{il} x_i + \theta^1_{0l}\right) + \theta^2_{0k}\right) \\
&= \sigma\left(\sum_{l=0}^{M} \theta^2_{lk} f\left(\sum_{i=1}^{D} \theta^1_{il} x_i\right)\right) \qquad\qquad x_0 = 1
\end{aligned}
$$

evaluating this is called forward propagation

Note:

- this is not a prob. graphical model because nodes represent deterministic variables, not stochastic.
- perceptrons use step function $\rightsquigarrow$ nonlinarity
  NN uses continuous sigmoidal $\rightsquigarrow$ nonlinear (but differentiable)

- If all hidden layers have linear activation function $\Longrightarrow \exists$ equivalent network without hidden layer
  (composition of linear transformations is itself a linear transformation)

- If number of hidden layers is smaller than input and output nodes $\Longrightarrow$not most general possible linear transformation

- Some confusion in counting layers:
  - 3-layers
  - single hidden layer
  - 2-layers (num. of adaptive weights)

Training

Deterministic approach: minimize error function:

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

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