

Contents

1 资料

2 结构

| | |
|---|---|
| 2.1 历史与现况 | 1 |
| 2.1.1 历史 | 1 |
| 2.1.2 现况 | 1 |
| 2.2 监督机器学习结构 | 1 |
| 2.3 线性回归/感知机 | 1 |
| 2.3.1 介绍与分类原理 | 1 |
| 2.3.2 特征选择问题 (多项式) | 2 |
| 2.3.3 正则/Overfitting | 2 |
| 2.3.4 训练方法问题 | 2 |
| 2.3.5 感知机的局限 (异或问题) | 3 |
| 2.4 多层感知机/ANN | 3 |
| 2.4.1 网络结构 (linear model + activity function) | 3 |
| 2.4.2 激活函数 (Sigmoid/ Tanh / ReLU etc.) | 3 |
| 2.4.3 前向算法 | 3 |
| 2.4.4 后向算法 | 3 |
| 2.5 无监督学习 | 3 |
| 2.5.1 Autoencoder | 3 |
| 2.6 卷积网络 | 3 |
| 2.6.1 理解 Sobel 边界检测 | 3 |
| 2.7 RNN | 3 |
| 2.8 如何构造与应用 | 3 |

3 Refs

1 资料

1. *Deep Learning* PDF on github
2. CS231n Convolutional Neural Networks for Visual Recognition
3. Neural Networks for Machine Learning
4. UFLDL Tutorial

2 结构

2.1 历史与现况

2.1.1 历史

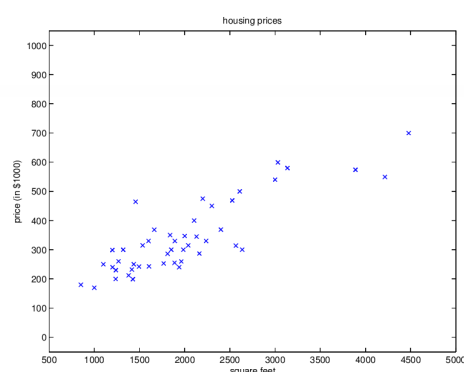
从 1940s 到 2014: <http://people.idsia.ch/~juergen/DeepLearning15May2014.pdf>

2.1.2 现况

1. Andrew Ng, Highway
2. NLP
3. Imagenet
- 4.

2.2 监督机器学习结构

假设我们获得了一部分的房价与房屋面积数据, 如



下:

对应的一条数据的格式是 $\{x, y\}$, 其中 x 为面积, y 为房价. 这些数据的 (一部分) 集合 $\{(x_i, y_i); i = 1, \dots, m\}$ 可以叫做训练集 (training set)

那么, 监督机器学习可以如下理解:

- 3 利用已知的训练集, 学习一个函数 $h: X \rightarrow Y$, $h(x)$ 的结果对 y 的预测达到我们预设的 足够好 的标准.

如果我们需要根据房屋面积预测房价, 那么我们需要学习一个函数 $h(x)$, 输入 x 为面积, 输出 y 为房价. 最简单的一个可以学习的函数就是线性函数, 对于 y 的预测, 属于回归问题.

2.3 线性回归/感知机

2.3.1 介绍与分类原理

1. 线性回归的形式

$$y = W * x + b \quad (1)$$

其中, $W = w_0, w_1, \dots, w_n$, $x = x_0, x_1, \dots, x_n$, 如果将向量形式展开, 变成标量的格式, 则表达式为:

$$y = \sum_0^n w_i * x_i + b \quad (2)$$

模型的参数 W 需要通过学习来获得. 那么, 如何学习 W 的具体值? 根据我们之前对于监督机器学习的理解, 我们需要一个预设的”足够好”的标准. 这一标准是需要人工选择的, 机器无法判断 $h(x)$ 的结果与对应的 y 需要符合什么样的标准才是”好”.

2. Cost Function 对于上面的例子, 我们可以这样考虑:

$h(x)$ 的预测与实际 y 值之间的差别越小越好. 特别是, 如果在理想情况下, 应该是一模一样.

于是我们可以定义: $|h(x) - y|$ 作为评价指标. 考虑到绝对值后续会遇到判断的问题, 更加简单的形式可以定义为 $J = \frac{1}{2} (h(x) - y)^2$. 那么针对这一问题的学习过程就是, 通过改变 W 的值来最小化 J 的过程.

3. 学习过程 为了获得 J 的最小值, 我们可以有如下选择:

- (a) 不断随机 W 值, 找到最小值.
- (b) 在一个随机的 W 值的基础上, 使用随机的修正值.
- (c) 在一个随机的 w 值的基础上, 使用有目的的修正值.

很明显的, 第三种方法是效率上最高的. 那么, 如何获得 有目的 的修正值? J 在最小化的情况下, 其导数是为 0 的, 而恰好 J 是一个理想的凸函数, 只需要沿着导数下降的方向进行修正, 就能保证到达 J 的最小值. 所以, 修正的方法就是:

$$w_j = w_j - \alpha \frac{d}{dw_j} J(W)$$

其中 α 被称为 **learning rate**. 这一方法就是在函数的梯度中, 找到最快速下降的方向.

为了实现这样的更新算法, 我们需要知道 $\frac{d}{dw_j} J(W)$ 这一项. 我们已知 J 的形式, 于是针对每个 w_j 求偏导就可以:

$$\begin{aligned} \frac{d}{dw_j} J(W) &= \frac{d}{dw_j} \frac{1}{2} (h_w(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_w(x) - y) \cdot \frac{d}{dw_j} (h_w(x) - y) \\ &= (h_w(x) - y) \cdot \frac{d}{dw_j} \left(\sum_{i=0}^n w_i x_i - y \right) \\ &= (h_w(x) - y) x_j \end{aligned}$$

于是我们每次更新的方式就是:

$$w_j = w_j - \alpha (h(x) - y) x_j$$

由于我们的训练集 X 是一组数据, 我们可以选择两种方式来更新:

- 每个 x 更新一次

```
while {
  for i = 1 to m {
     $w_j = w_j - \alpha (h(x_i) - y) x_j$  (for every j)
  }
}
```

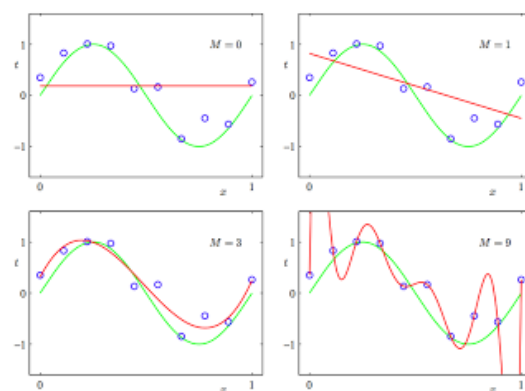
- 遍历所有 x 后统一更新

```
while {
   $w_j = w_j - \alpha \sum_{i=1}^m (h(x_i) - y) x_j$  (for every j)
}
```

另外可以作为折中, 遍历一部分 x 后再更新.

2.3.2 特征选择问题 (多项式)

2.3.3 正则/Overfitting



2.3.4 训练方法问题

1. 牛顿法?
2. 验证集/训练集的解释问题
3. Learning Rate 相关

2.3.5 感知机的局限 (异或问题)

2.4 多层感知机/ANN

2.4.1 网络结构 (linear model + activity function)

2.4.2 激活函数 (Sigmoid/ Tanh / ReLU etc.)

2.4.3 前向算法

2.4.4 后向算法

2.5 无监督学习

2.5.1 Autoencoder

2.6 卷积网络

2.6.1 理解 Sobel 边界检测

2.7 RNN

1. <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

2.8 如何构造与应用

3 Refs

1. How to Layout and Manage Your Machine Learning Project
2. Machine learning in 10 pictures
3. 深度学习-wiki
4. Ufldl tutorial in Python
5. <https://colah.github.io/posts/2014-07-Conv-Nets-Modular/> Github
6. <http://cs229.stanford.edu/materials.html>