人工智能原理

第19章 – 样例学习

彭振辉
中山大学人工智能学院
2024年春季

Learning I 至 Learning III 大纲

- 监督学习
  - 决策树
  - 线性回归和分类
- 评估和选择最佳假说
- 人工神经网

- 极大似然参数学习
  - 贝叶斯参数学习

- 19.5, 19.7 – 19.9, 20.3 为课外阅读，第21章待定

随堂习题：决策树

现在我们考虑用决策树对这些样例进行分类。

y	0	0	0	0	0	1	1	1	1	1
x_1	1	0	0	0	0	1	0	0	1	1
x_2	0	0	1	0	0	1	1	1	1	1
x_3	0	0	0	0	0	0	1	1	0	0

(i) 所有决策树都是线性分类器吗（即其不能表示非线性函数）？__（填：是 或 否）（2分）

(ii) x_1, x_2 和 x_3 中哪一个具有最高的信息增益？_____（单选，2分）

A. x_1　B. x_2　C. x_3

随堂习题：决策树

现在我们考虑用决策树对这些样例进行分类。

y	0	0	0	0	0	1	1	1	1	1
x_1	1	0	0	0	0	1	0	0	1	1
x_2	0	0	1	0	0	1	1	1	1	1
x_3	0	0	0	0	0	0	1	1	0	0

(i) 所有决策树都是线性分类器吗（即其不能表示非线性函数）？否 (填：是 或 否)（1分）

(ii) x_1, x_2 和 x_3 中哪一个具有最高的信息增益？__B___(单选，2分)

A. x_1　B. x_2　C. x_3

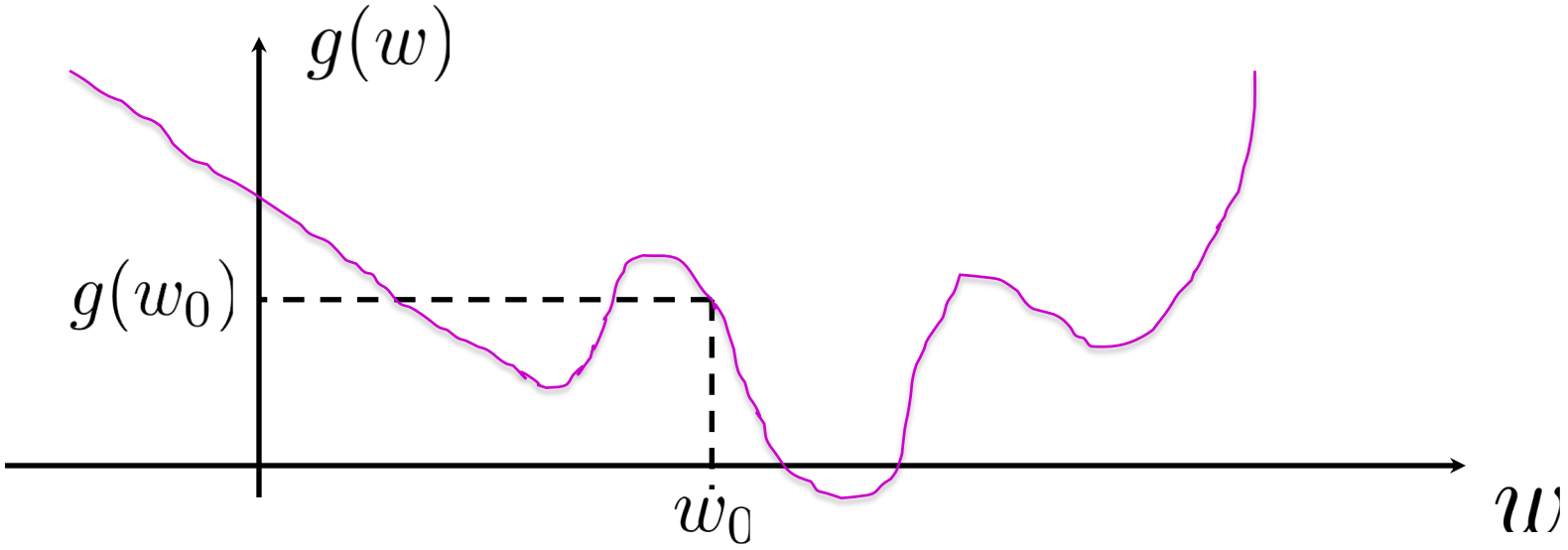
Optimization 优化

知道优化目标了，但具体怎么做优化呢?

- Optimization 优化
  - 即，如何求解:

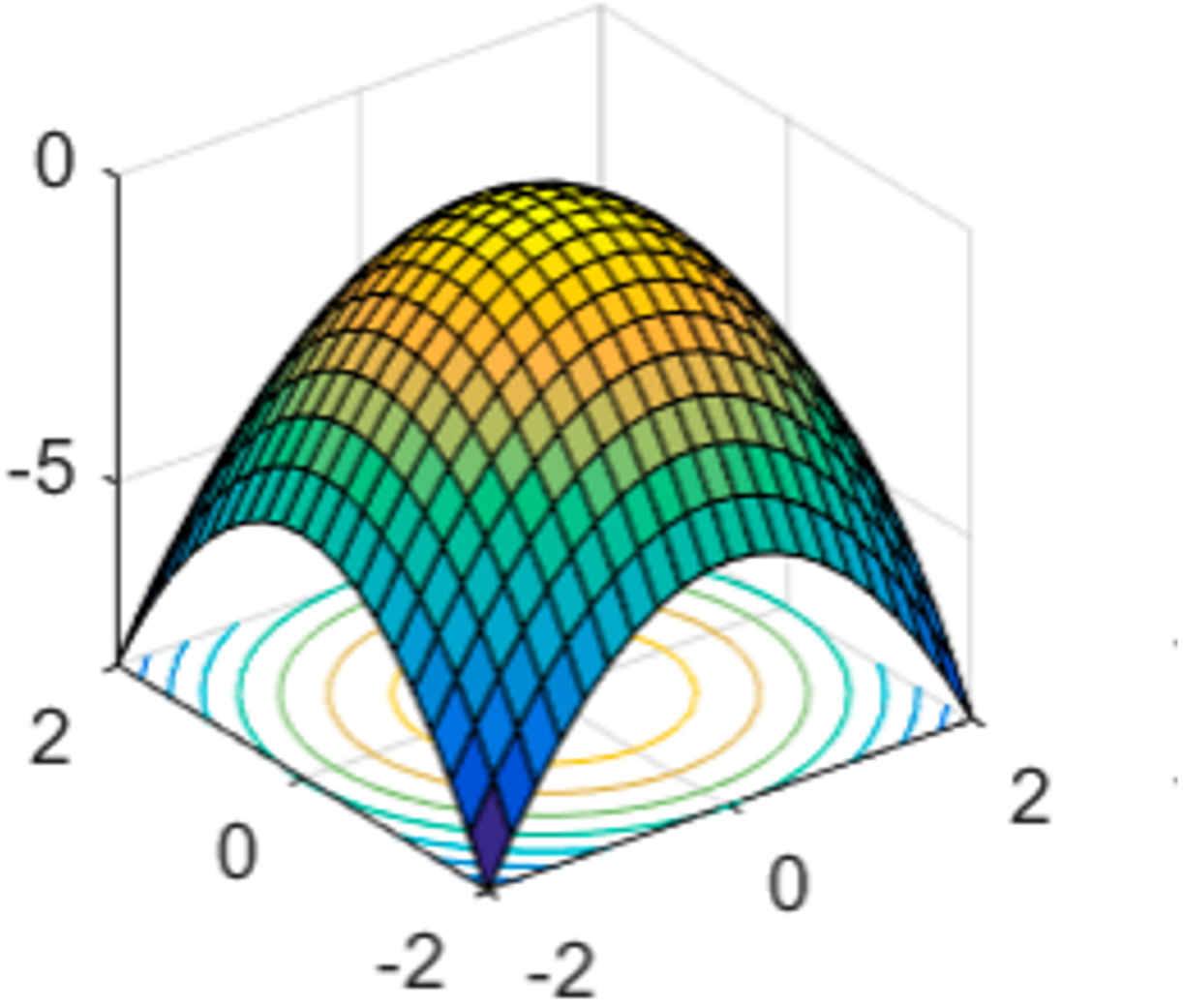
$$\max_{w} \quad ll(w) = \max_{w} \quad \sum_{i} \log P(y^{(i)}|x^{(i)}; w)$$

1-D Optimization



- 可以分析 $g(w_0 + h)$ 和 $g(w_0 - h)$
  - 然后往最好的方向前进
- 或者, 分析导数: $\dfrac{\partial g(w_0)}{\partial w} = \lim_{h \to 0} \dfrac{g(w_0 + h) - g(w_0 - h)}{2h}$
  - 告知我们往哪个方向前进

2-D Optimization



Source: offconvex.org

Gradient Ascent 梯度上升

- 应对使目标函数最大化的情况
- 对每个坐标进行上坡方向更新
- 斜率越陡（即导数越大），该坐标的更新步长越大

- 例如，考虑：$g(w_1, w_2)$
  - 更新:

$$w_1 \leftarrow w_1 + \alpha * \frac{\partial g}{\partial w_1}(w_1, w_2)$$

$$w_2 \leftarrow w_2 + \alpha * \frac{\partial g}{\partial w_2}(w_1, w_2)$$

- 用向量标记表示更新:

$$w \leftarrow w + \alpha * \nabla_w g(w)$$

with:

$$\nabla_w g(w) = \begin{bmatrix} \frac{\partial g}{\partial w_1}(w) \\ \frac{\partial g}{\partial w_2}(w) \end{bmatrix}$$

= gradient 梯度

Steepest Ascent 最陡上升

- Idea:
  - Start somewhere
  - Repeat: Take a step in the **steepest ascent direction**

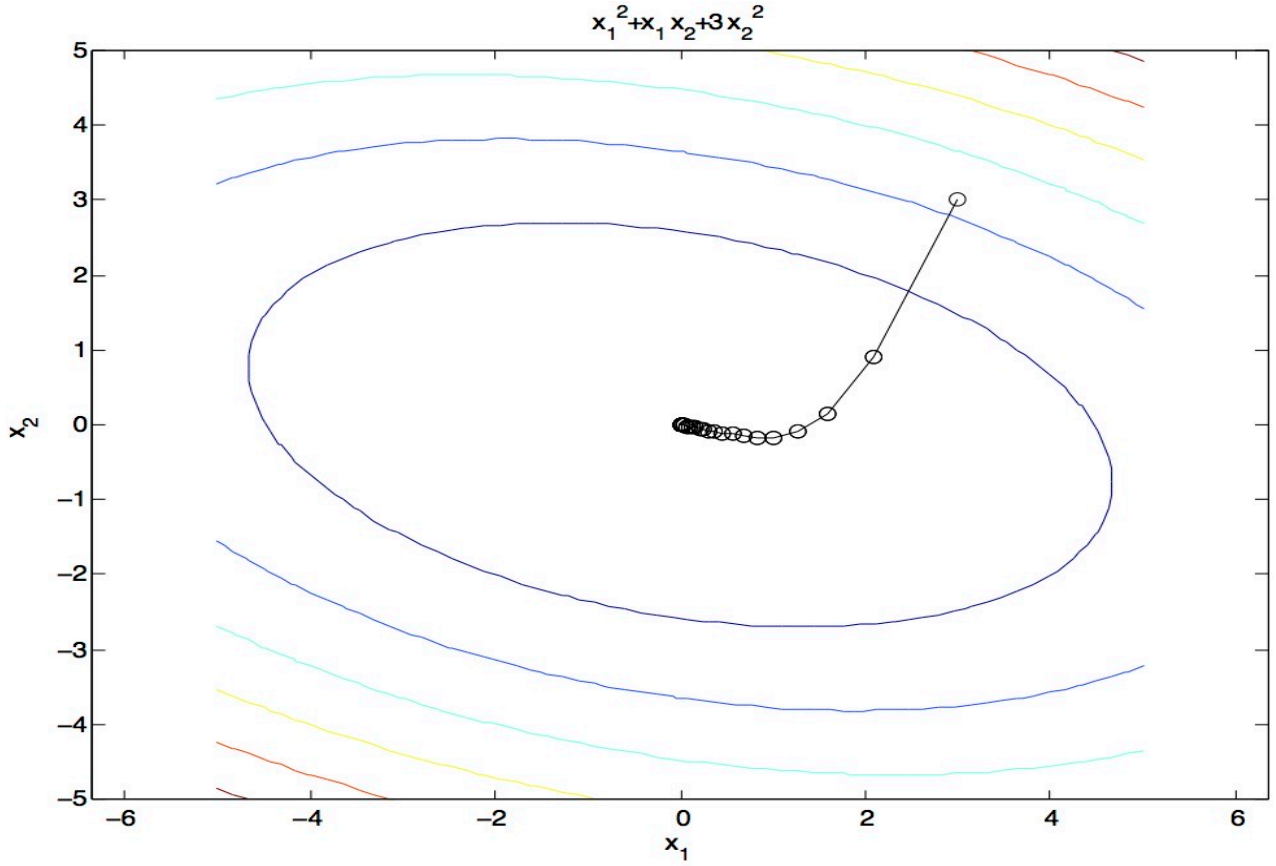


Figure source: Mathworks

Steepest Direction

- Steepest Direction = direction of the gradient

$$\nabla g = \begin{bmatrix} \frac{\partial g}{\partial w_1} \\ \frac{\partial g}{\partial w_2} \\ \cdots \\ \frac{\partial g}{\partial w_n} \end{bmatrix}$$

Optimization Procedure: Gradient Ascent 梯度上升

- init w
- for iter = 1, 2, …

$$w \leftarrow w + \alpha * \nabla g(w)$$

α :学习速率 --- 需要谨慎选择的超参数

Batch Gradient Ascent on the Log Likelihood Objective

$$\max_w \quad ll(w) = \max_w \quad \underbrace{\sum_i \log P(y^{(i)}|x^{(i)};w)}_{g(w)}$$

- init w
- for iter = 1, 2, …

$$w \leftarrow w + \alpha * \sum_i \nabla \log P(y^{(i)}|x^{(i)};w)$$

能够得到保证收敛到全局的唯一极小值（只要α足够小），
但收敛速度可能非常缓慢。为什么?
每一步都需要遍历所有训练数据，收敛往往又需要很多步骤。怎么改进?

Stochastic Gradient Ascent on the Log Likelihood Objective

$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)};w)$$

Observation：一旦计算了一个训练样本的梯度，不妨在计算下一个样本之前合并 (随机梯度上升):

- init w
- for iter = 1, 2, …
  - pick random j

$$w \leftarrow w + \alpha * \nabla \log P(y^{(j)}|x^{(j)};w)$$

随机梯度下降经常比批梯度下降快。如果学习速率固定，则不能保证收敛；它可能在极小值周围游离不定。在某些情况下，学习速率的递减序列能保证收敛。

Mini-Batch Gradient Ascent on the Log Likelihood Objective

$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

Observation：也可以并行计算少量训练样例（= mini-batch）上的梯度，而不是计算单个样例:

- init w
- for iter = 1, 2, …
  - pick random subset of training examples J

$$w \leftarrow w + \alpha * \sum_{j \in J} \nabla \log P(y^{(j)}|x^{(j)}; w)$$

Mini-Batch Gradient Ascent on the Log Likelihood Objective

$$\frac{\partial s_i}{\partial z_j} = s_i \cdot \frac{\partial}{\partial z_j} log(s_i) = s_i \cdot \left(1\{i = j\} - s_j\right)$$

https://towardsdatascience.com/derivative-of-the-softmax-function-and-the-categorical-cross-entropy-loss-ffceefc081d1

Mini-Batch Gradient Ascent on the Log Likelihood Objective

How about using a neural network with softmax?

Example: four outputs

$$J_{softmax} = \begin{pmatrix} s_1 \cdot (1 - s_1) & -s_1 \cdot s_2 & -s_1 \cdot s_3 & -s_1 \cdot s_4 \\ -s_2 \cdot s_1 & s_2 \cdot (1 - s_2) & -s_2 \cdot s_3 & -s_2 \cdot s_4 \\ -s_3 \cdot s_1 & -s_3 \cdot s_2 & s_3 \cdot (1 - s_3) & -s_3 \cdot s_4 \\ -s_4 \cdot s_1 & -s_4 \cdot s_2 & -s_4 \cdot s_3 & s_4 \cdot (1 - s_4) \end{pmatrix}$$