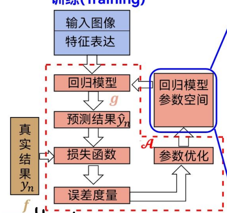
§3.1 线性回归问题

⊗ 增广 $\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$

训练(Training)



$\mathcal{H}(\cdot)$ 像什么?

输入: $\mathbf{x} = (x_1, x_2, \dots x_d)^T$

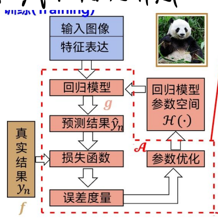
计算输入 x 的加权和: $\hat{y} = \sum_{i=1}^{d} w_i x_i$

回归模型的参数空间: $h(\ ) \in \mathcal{H}(\cdot)$

$h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$

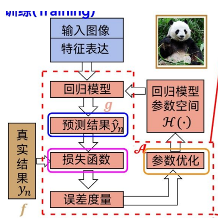
线性回归的目的: 找到误差最小的拟合模型

§3.2 线性回归算法



算法$\mathcal{A}$的目的是在$\mathcal{H}(h(\cdot))$中找到最优结果作为回归模型 g

- 最优结果: $g \approx f$
- 挑战: f 未知
- 学习的资源: 在训练集$\mathcal{D}$上, 如果每一个样本都有: $g(\mathbf{x}_n) = y_n = f(\mathbf{x}_n)$, 则在训练集$\mathcal{D}$上做到了$g \approx f$
- 困难: $\mathcal{H}(h(\cdot))$ 的候选模型无穷多



$$\hat{y}_n = \mathbf{w}^T \mathbf{x}_n$$

$$L_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2$$

$$g = \underset{\mathbf{w}}{argmin}\ \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2$$

有矩阵/向量形式 $L_{in}(w)$

$$L_{in}(w) = \frac{1}{N}\sum_{n=1}^{N}(w^T x_n - y_n)^2 = \frac{1}{N}\left\|\begin{matrix} x_1^T w - y_1 \\ \cdots \\ x_N^T w - y_N \end{matrix}\right\|^2 = \frac{1}{N}\left\|\begin{bmatrix} -x_1^T- \\ \vdots \\ -x_N^T- \end{bmatrix} w - \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}\right\|^2$$

$$= \frac{1}{N}\|Xw - Y\|^2$$

$X: N\times(d+1)$
$w: (d+1)\times 1$
$Y: N\times 1$

求最优解 $\min_{w} \underset{\text{连续、可微、凸函数}}{L_{in}(w)} = \frac{1}{N}\|Xw - Y\|^2 \Rightarrow \nabla L_{in}(w) = 0$ 有最佳解

$$\nabla L_{in}(w) = \begin{bmatrix} \frac{\partial L_{in}(w)}{\partial w_0} \\ \cdots \\ \frac{\partial L_{in}(w)}{\partial w_d} \end{bmatrix} = \begin{bmatrix} 0 \\ \cdots \\ 0 \end{bmatrix}$$

$\nabla L_{in}(\mathbf{w})$求解:

$$L_{in}(\mathbf{w}) = \frac{1}{N}\|X\mathbf{w} - Y\|^2 = \frac{1}{N}(\mathbf{w}^T \underbrace{X^T X}_{A} \mathbf{w} - 2\mathbf{w}^T \underbrace{X^T Y}_{b} + \underbrace{Y^T Y}_{c}) = \frac{1}{N}(\mathbf{w}^T A\mathbf{w} - 2\mathbf{w}^T b + c)$$

当w是单变量时
$L_{in}(w) = \frac{1}{N}(aw^2 - 2bw + c)$
$\nabla L_{in}(w) = \frac{1}{N}(2aw - 2b)$

当$\mathbf{w}$是向量时
$L_{in}(\mathbf{w}) = \frac{1}{N}(\mathbf{w}^T A\mathbf{w} - 2\mathbf{w}^T b + c)$
$\nabla L_{in}(\mathbf{w}) = \frac{1}{N}(2A\mathbf{w} - 2b)$

$$\nabla L_{in}(\mathbf{w}) = \frac{2}{N}(X^T X\mathbf{w} - X^T Y)$$

令 $\nabla L_{in}(w^*) = 0$ 解得

$g = w^*$

$= (X^T X)^{-1} X^T Y = X^{+} Y$

伪逆 X^{+}: $(d+1)\times N$

⊗ 参数量大时 X^{+} 计算繁

§3.3 GD

感知器算法的GD

- 对样本的特征向量x和权向量w 增广化
- 初始化权向量w_0 (例如: $w_0 = 0$)
- $for\ t = 0,1,2,\dots$ (t 代表迭代次数)

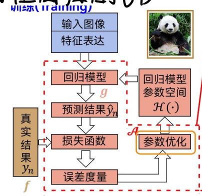
  ① 对某些样本n, 通过下式对权向量w_t进行更新:

$$w_{t+1} = w_t + \underbrace{1}_{\eta} \cdot \underbrace{(\llbracket \mathrm{sign}(w_t^T \mathbf{x}_{n(t)}) \neq y_n \rrbracket \mathbf{x}_{n(t)})}_{\mathbf{v}}$$

…直到满足停止条件, 此时的w_{t+1}作为学到的g

算法可理解成通过选择(η, $\mathbf{v}$), 以及确定"停止条件"的找到最佳解的迭代优化过程

线性回归的GD



梯度下降法:

$$w_{t+1} \leftarrow w_t - \eta\nabla L_{in}(w_t)$$

$$L_{in}(\mathbf{w}) = \frac{1}{N}\|X\mathbf{w} - Y\|^2$$

$$\nabla L_{in}(\mathbf{w}) = \frac{2}{N}(X^T X\mathbf{w} - X^T Y)$$

$$L_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2$$

$$\nabla L_{in}(\mathbf{w}) = \frac{2}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)\mathbf{x}_n$$

- 初始化权向量w_0
- $for\ t = 0,1,2,\dots$ (t 代表迭代次数)

  ① 计算梯度: $\nabla L_{in}(\mathbf{w}) = \sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)\mathbf{x}_n$

  ② 对权向量w_t进行更新: $w_{t+1} \leftarrow w_t - \eta\nabla L_{in}(w_t)$

…直到$\nabla L_{in}(\mathbf{w}) = \mathbf{0}$, 或者迭代足够多次数

返回最终的w_{t+1}作为学到的g

据梯度调整学习率η引出改良优化器

① Adagrad

$$w_{i,t+1} \leftarrow w_{i,t} - \frac{\eta}{\sigma_{i,t}}\frac{\partial L_{in}}{\partial w_{i,t}}$$

$$\sigma_{i,t} = \sqrt{\frac{1}{t+1}\sum_{t=0}^{t}\left(\frac{\partial L_{in}}{\partial w_{i,t}}\right)^2}$$

曲线平缓σ小, 步大

② RMS prop

$$w_{i,t+1} \leftarrow w_{i,t} - \frac{\eta}{\sigma_{i,t}}\frac{\partial L_{in}}{\partial w_{i,t}}$$

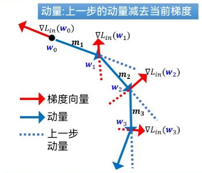
$\frac{\partial L_{in}}{\partial w_{i,0}}, \frac{\partial L_{in}}{\partial w_{i,1}}, \dots, \frac{\partial L_{in}}{\partial w_{i,t-1}}$ $\quad 0 < \alpha < 1$

$$\sigma_{i,t} = \sqrt{\alpha(\sigma_{i,t-1})^2 + (1-\alpha)\left(\frac{\partial L_{in}}{\partial w_{i,t}}\right)^2}$$

通过 α 的取值, 使得当前的梯度影响更大, 而以往的梯度影响较小

逃离鞍点/局部极值引出改良优化器

③ Momentum



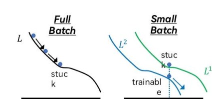
初始值为w_0
动量 $m_0 = 0$
计算梯度: $\nabla L_{in}(w_0)$
动量 $m_1 = \lambda m_0 - \eta\nabla L_{in}(w_0)$
权向量更新$w_1 \leftarrow w_0 + m_1$
计算梯度: $\nabla L_{in}(w_1)$
动量 $m_2 = \lambda m_1 - \eta\nabla L_{in}(w_1)$
权向量更新$w_2 \leftarrow w_1 + m_2$

权向量更新时不仅考虑负梯度方向还同时要考虑上一步的动量

④ 结合两者 Adam

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector) → for momentum
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector) → for RMSprop
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t/(\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)



批量用于梯度下降

⊗ 批次↑ ⇒ Batch速度↓ epoch速度↑ 性能↓, 小批量时部分批次鞍点无梯度, 其余批次可能有梯度仍可继续训练

- 初始化权向量w_0
- $for\ t = 0,1,2,\dots$ (t 代表迭代次数)

  ① 计算梯度: $\nabla L_{in}(\mathbf{w}) = \frac{2}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)\mathbf{x}_n$ $\qquad \sum_{n=1}^{B}(w^T x_n - y_n)x_n$

  每次迭代N个样本均要计算, 时间复杂度与Pocket算法相似

  ② 对权向量w_t进行更新: $w_{t+1} \leftarrow w_t - \eta\nabla L_{in}(w_t)$

…直到$\nabla L_{in}(\mathbf{w}) = \mathbf{0}$, 或者迭代足够多次数

返回最终的w_{t+1}作为学到的g

- 初始化权向量w_0
- $for\ t = 0,1,2,\dots$ (t 代表迭代次数)

  ① 计算梯度: $\nabla L_{in}(\mathbf{w}) = (\mathbf{w}^T\mathbf{x}_n - y_n)\mathbf{x}_n$

  随机梯度下降法 (Stochastic Gradient Descent) (SGD)

  ② 对权向量w_t进行更新: $w_{t+1} \leftarrow w_t - \eta\nabla L_{in}(w_t)$

…直到$\nabla L_{in}(\mathbf{w}) = \mathbf{0}$, 或者迭代足够多次数

返回最终的w_{t+1}作为学到的g

	批量小	批量大
一次更新需要的速度(无并行处理)	Faster	Slower
一次更新需要的速度(有并行处理)	Same	Same (not too large)
一个epoch花费的时间	Slower	Faster
梯度的特点	Noisy	Stable
优化性能	Better	Worse
泛化性能	Better	Worse