ML Refresher / Softmax Regression

机器学习是数据驱动的。监督学习中，训练集导入机器学习算法中进行模型训练，得到的模型可以进行推理预测。

机器学习算法的三个组成部分：

1. The hypothesis class: 模型框架，用于描述如何将输入转化为输出，包含很多参数。
2. The loss function: 描述给定参数下的模型在任务中的表现的函数。
3. An optimization method: 优化参数的方法 / 过程，使得在训练集上的 loss 最小化。

Multi-class Classification Setting

考虑 k-class 分类问题：

对每个训练数据 $x^{(i)} \in R^n$, 有 $y^{(i)} \in \{1, \ldots, k\}$, 其中 $i = 1, \ldots, m$

n 代表输入数据的维度，m 代表训练集大小，k 代表类别数量。

Linear Hypothesis Function

Hypothesis function 将输入 $x \in R^n$ 映射到 k 维向量

$$h : R^n \rightarrow R^k \tag{2}$$

其中 $h_i(x)$ 表示属于类 i 的可能性大小。

Linear hypothesis function 用线性算子（矩阵乘法）来做转换

$$h_\theta(x) = \theta^T x \tag{3}$$

其中 $\theta \in R^{n \times k}$

Matrix Batch Notation

将输入的向量组合成矩阵，把上文所述的矩阵乘向量推广成矩阵乘矩阵，更为高效。

$$X \in R^{m \times n} = \begin{bmatrix} x^{(1)^T} \\ \ldots \\ x^{(m)^T} \end{bmatrix}, y \in \{1, \ldots, k\}^m = \begin{bmatrix} y^{(1)} \\ \ldots \\ y^{(m)} \end{bmatrix} \tag{4}$$

$$h_\theta(X) = \begin{bmatrix} h_\theta(x^{(1)})^T \\ \ldots \\ h_\theta(x^{(m)})^T \end{bmatrix} = X\theta \tag{5}$$

Loss Function

Classification Error

最简单的 loss function, 只分对错。

$$\ell_{err}(h(x), y) = \begin{cases} 0 & \text{if } \operatorname{argmax}_i h_i(x) = y \\ 1 & \text{otherwise} \end{cases}$$

通常用于衡量分类器的质量，但不适合用于优化，因为不可导

Softmax / Cross-Entropy Loss

将 hypothesis function 的每个分量通过 exponentiating 和 normalizing 转化为概率值

$$z_i = p(\text{label} = i) = \frac{\exp\left(h_i(x)\right)}{\sum_{j=1}^{k} \exp\left(h_j(x)\right)} \Longleftrightarrow z \equiv \text{normalize}(\exp(h(x)))$$

也可以直接称为 $softmax(h(x))$

Softmax / cross-entropy loss:

$$\begin{aligned} l_{ce}(h(x), y) &= -log(p(label = y)) \\ &= -h_y(x) + log\sum_{j=1}^{k} exp(h_j(x)) \end{aligned} \tag{6}$$

Softmax Regression Optimization Problem

优化问题在于最小化平均 loss:

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} l(h_\theta(x^{(i)}), y^{(i)}) \tag{7}$$

Softmax regression: linear hypothesis class + softmax loss

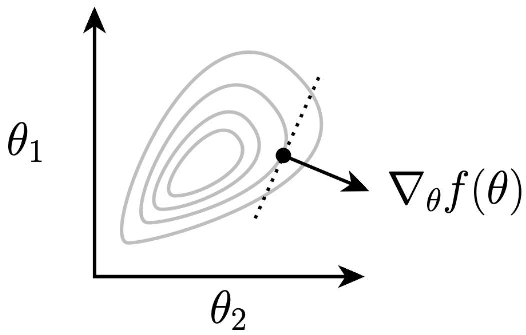
For softmax regression:

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} l_{ce}(\theta^T x^{(i)}, y^{(i)}) \tag{8}$$

Optimization: Gradient Descent

对于输入为矩阵的函数 $f : R^{n \times k} \rightarrow R$, 梯度定义如下:

$$\nabla_\theta f(\theta) \in \mathbb{R}^{n\times k} = \begin{bmatrix} \frac{\partial f(\theta)}{\partial \theta_{11}} & \cdots & \frac{\partial f(\theta)}{\partial \theta_{1k}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(\theta)}{\partial \theta_{n1}} & \cdots & \frac{\partial f(\theta)}{\partial \theta_{nk}} \end{bmatrix}$$



梯度指向让 f 局部增加最大的方向

梯度下降：为了最小化函数值，让 θ 向梯度的反方向移动

$$\theta := \theta - \alpha \boldsymbol{\nabla}_\theta f(\theta) \tag{9}$$

其中 $\alpha > 0$ 称为 step size / learning rate.

α 的选择对优化影响很大。

很多新的优化方法用于加速梯度下降过程，规避 α 选择的不便。

Stochastic Gradient Descent (SGD)

对于一般的 loss function, 需要求 loss 的均值，上文所述的梯度算法会用到所有训练数据，很容易把内存用完。

解决方法：将训练集划分为多个 minibatch, 采用多次梯度下降，每次只用到一个 batch 的数据。在单次遍历所有数据的过程中进行多次参数更新。

Repeat:

Sample a minibatch of data $X \in \mathbb{R}^{B\times n}, y \in \{1, \ldots, k\}^B$

Update parameters $\theta := \theta - \dfrac{\alpha}{B} \displaystyle\sum_{i=1}^{B} \nabla_\theta \ell(h_\theta(x^{(i)}), y^{(i)})$

其中 B 为 batch size.

这种方法迭代速度更快

Gradient of Softmax Objective

目标：

$$\boldsymbol{\nabla}_\theta l_{ce}(\theta^T x, y) \tag{10}$$

令 $h = \theta^T x \in R^k$, 先求 $\frac{\partial l_{ce}(h,y)}{\partial h_i}$

$$\frac{\partial \ell_{ce}(h,y)}{\partial h_i} = \frac{\partial}{\partial h_i}\left(-h_y + \log\sum_{j=1}^{k}\exp h_j\right)$$
$$= -1\{i=y\} + \frac{\exp h_i}{\sum_{j=1}^{k}\exp h_j}$$

于是

$$\nabla_h l_{ce}(h,y) = z - e_y \tag{11}$$

其中 $z = normalzie(exp(h))$

一般方法：链式法则嗯算，但容易出错

诡异但常用的方法：假设一切都是标量，用最基本的链式法则，然后重新排列 / 转置矩阵 / 向量，使维度符合要求，并在最后检查结果数值。

$$\frac{\partial}{\partial\theta}\ell_{ce}(\theta^T x, y) = \frac{\partial \ell_{ce}(\theta^T x, y)}{\partial \theta^T x}\frac{\partial \theta^T x}{\partial \theta}$$
$$= \underbrace{(z - e_y)}_{(k\text{-dimensional})}\underbrace{(x)}_{(n\text{-dimensional})}, \qquad \left(\text{where } z = \text{normalize}(\exp(\theta^T x))\right)$$

为了让维度符合要求，对结果进行重排：

$$\nabla_\theta \ell_{ce}(\theta^T x, y) \in \mathbb{R}^{n\times k} = x(z - e_y)^T$$

这对 matrix batch 也有效：

$$\nabla_\theta \ell_{ce}(X\theta, y) \in \mathbb{R}^{n\times k} = X^T(Z - I_y), \qquad Z = \text{normalize}(\exp(X\theta))$$

> 这种方法中维度很重要，如果 n = k, 则很容易在重排时出错

最终的 softmax regression 算法如下：

- Repeat until parameters / loss converges
  1. Iterate over minibatches $X \in \mathbb{R}^{B\times n}$, $y \in \{1, \dots, k\}^B$ of training set
  2. Update the parameters $\theta := \theta - \frac{\alpha}{B} X^T(Z - I_y)$