Subject: Stanford CS229 Machine Learning, Lecture 11, Feature / Model selection, ML Advice

Date: from February 9, 2025 to February 12, 2025

Contents

A Weight Decay **4**

CS229 Machine Learning, Feature / Model selection, ML Advice, 2022, Lecture 11

YouTube:Stanford CS229 Machine Learning, Feature / Model selection, ML Advice, 2022, Lecture 11

Complexity Measure

Complexity

在上一节中 overfitting 与 underfitting 所围绕的核心是 model complexity，部分可供选择的 complexity measure 如下：

1. **# parameters.** 即参数数量. 但缺陷在于训练完的模型可能很多参数都很小甚至为 0，导致实际参数量 $<$ # parameters.
2. **Norm of parameters.** 常用 norm 有 $\|\cdot\|_1, \|\cdot\|_2$. 可以解决某些参数为 0 的问题，但缺陷在于当 norm 过小时容易受到噪声影响.
3. **Lipschitzness / Smoothness.** 用于衡量模型表示函数的光滑性（这里暂略）.

How to Decrease Model Complexity? – Regularization

解决 overfitting 的方法之一是降低 model complexity. 针对不同的 complexity measure，其中一种方式是直接减小模型参数量，另一种方式是降低模型参数的 norm，即正则化(**regularization**).

简单来说，regularization 是在 training loss 中加上附加项以促使得到较低复杂度的模型：

$$\text{new loss} = J(\theta) + \lambda \cdot \mathcal{R}(\theta), \tag{1}$$

其中 $J(\theta)$ 为 loss function，$\mathcal{R}(\theta)$ 被称为 regularizer，λ 为 regularization strength，用以平衡 loss 与 regularizer 之间的 trade-off.

常用的 regularizer 有：

1. $\mathcal{R}(\theta) = \frac{1}{2}\|\theta\|_2^2$，被称为 L_2-regularization 或 weight decay[a].
2. $\mathcal{R}(\theta) = \|\theta\|_0 = \#\text{ non-zero in } \theta$，被称为 sparsity[b].
3. $\mathcal{R}(\theta) = \|\theta\|_1 = \sum_{i=1}^{d} |\theta_i|$，是 $\|\theta\|_0$ 的替代，解决了其不可导的问题.

Note 1. *Regularization* 实际上反映了模型的结构*(structure)*，这是建立在先验知识*(prior belief)*上的，例如若希望只使用一部分特征，那么就可以选择 $\|\theta\|_0$ 和 $\|\theta\|_1$；若相信每一个特征都会有作用，需要将他们组合起来使用，就需要选择 $\|\theta\|_2$. 对于线性模型，$J(\theta) + \|\theta\|_1$ 被称为 ***LASSO***；对于深度学习任务，使用的主要是 *L_2-regularization*.

[a] 事实上二者对于 gradient decent 等价，详见附录. 但是对于 Adam 等并不等价，详见(1)

[b] 在线性模型中 $\theta^T x = \sum_{i=1}^{d} \theta_i x_i$，分量 θ_j 为 0 说明模型并没有选择特征 x_j，故称 sparsity.

Implicit Regularization

Implicit Regularization

Motivation. 在现代深度学习模型往往是过参数化的(over-parameterized)，此时并没有很强的正则化(例如 regularization strength λ 很小或为 0)但是仍然有很好泛化能力. 一种解释是在优化过程中发生了隐式正则化(implicit regularization).

对于 training loss 来说，其 loss landscape 可能有很多局部极小点(local mminima)甚至有数个极小点，但是 implicit regularization 会让模型最终选择更好的，即使得 test error 更小的模式. 如下图1 所示，对于 training error 来说 A, B 均为极小点，但是对于 test error 来说 B 点是极小点，implicit regularization 的作用就是使得在优化过程中选择 B 模型使其泛化能力更好(test error 更小)[a].

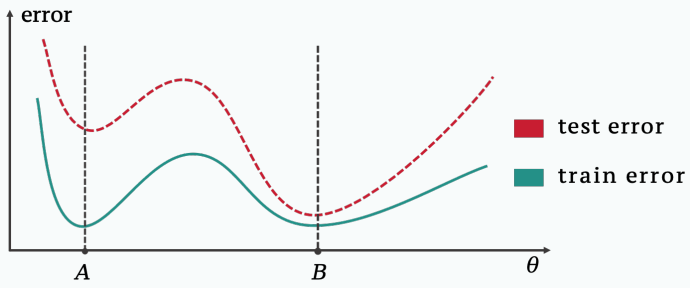

Figure 1: Implicit regularization example

Example – Linear Model. 考虑过参数化(over-parameterized)的线性模型：给定训练集 $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(n)}, y^{(n)})\}$，其中 $x^{(i)} \in \mathbb{R}^d$，$n \ll d$，损失函数 $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \theta^T x)^2$. 此时 training error 存在很多 global minima[b]，但结论表明：使用初始化于 0 点的梯度下降(GD)优化该模型会收敛至 norm 最小的解，即

$$\begin{aligned} &\arg\min \|\theta\|_2^2 \\ &s.t. \quad J(\theta) = 0 \end{aligned} \tag{2}$$

在下面图2 的例子中，考虑 $n = 1, d = 3$ 的可视化情形，此时 $J(\theta) = (\theta_1 x_1 - y_1)^2 + (\theta_3 x_3 - y_3)^2 + (\theta_3 x_3 - y_3)^2$，最终通过梯度下降收敛的点即为点 A，此时向量 $\vec{A}$ 与平面 $J(\theta) = 0$ 垂直.

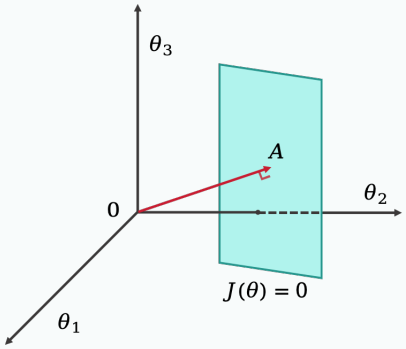

Figure 2: Caption

[a] 此处不做更多详细数学化解释，此领域仍然是一个开放性探索领域.
[b] 向量方程 $\theta^T x = y$ 共 n 个方程、d 个未知量，由于 $n \ll d$ 因此有很多组解，导致存在很多 global minima.

Validation

How to choose hyperparameters?

在介绍完 regularization 之后可以看到 model size, regulizer, regularization strength, optimization, learning rate 等参数的选择也会对模型效果产生影响，我们将上述并非模型参数的参数称为超参数(hyperparameter). 参数与超参数的调整是一个循环的过程：

$$\text{调整参数} \to \text{调整超参数} \to \text{在更优超参数上重新调整参数} \to \cdots$$

由于调整超参数的目的是提高模型的泛化能力，需要在模型"没见过"的数据上进行，因此不能使用训练集；而如果在测试集上调整超参数会导致模型拟合测试集的数据，使得模型评价失去意义，故需要新划分出集合用于调整超参数，称之为验证集(validation set / development set). 因此三种集合的作用如下：

1. **Training set.** 用于调整模型参数(parameter)
2. **Validation set / Development set.** 用于调整模型超参数(hyperparameter)
3. **Test set.** 用于测试模型泛化能力，只使用一次

Note 2. 三种数据集的划分一般而言是随机选取数据进行归类，但并不完全是，例如对于时序数据就需要按照时序排列进行划分以模拟时序预测的情形. 模型在验证集上表现好并不一定代表在测试集上表现好，但有结果表明二者具有强相关性*(2)*.

Machine Learning Advice

Machine Learning Advice

参考 ML Advice(Clipart day)，在构建机器学习系统(ML system)时有如下步骤：

1. **Acquire data.** 在获取 data 时要收集尽量好的数据，而非 spam data，例如期待 training data 与 test data 的分布要相近.
2. **Look at your data.** 观察数据，确保数据无异常，实际上在每一步后都需要做这件事.
3. **Create train / validation / test set.** 进行数据集划分，常见比例为 $6:2:2$.
4. **Create / Refine a specification.** 建立一个好的评定标准，如合适的损失函数.
5. **Build model.** 构建模型，实际上这一步是最简单的.
6. **Measurement.** 建立合适的模型效果评定标准.
7. **Repeat.**

A Weight Decay

Weight Decay

Proposition 1. *SGD with weight decay λ is:*

$$\theta_{t+1} = (1 - \lambda)\theta_t - \alpha \nabla f_t(\theta_t), \tag{3}$$

where λ is the weight decay coefficient. It is equivalent to the following update rule:

$$\theta_{t+1} = \theta_t - \alpha \nabla f_t^{reg}(\theta_t), \quad f_t^{reg}(\theta) = f_t(\theta) + \frac{\omega}{2}\|\theta\|_2^2, \quad \omega = \frac{\lambda}{\alpha}. \tag{4}$$

Proof. SGD without weight decay has the following iterates on $f_t^{\text{reg}}(\theta) = f_t(\theta) + \frac{\omega}{2}\|\theta\|_2^2$:

$$\theta_{t+1} \leftarrow \theta_t - \alpha \nabla f_t^{\text{reg}}(\theta_t) = \theta_t - \alpha \nabla f_t(\theta_t) - \alpha\omega\theta_t \tag{5}$$

SGD with weight decay has the following iterates on $f_t(\theta)$:

$$\theta_{t+1} \leftarrow \theta_t - \alpha \nabla f_t(\theta_t) - \lambda\theta_t. \tag{6}$$

These iterates are identical since $\omega = \frac{\lambda}{\alpha}$. □

Last updated: August 15, 2025

References

[1] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[2] Rebecca Roelofs, Vaishaal Shankar, Benjamin Recht, Sara Fridovich-Keil, Moritz Hardt, John Miller, and Ludwig Schmidt. A meta-analysis of overfitting in machine learning. *Advances in neural information processing systems*, 32, 2019.