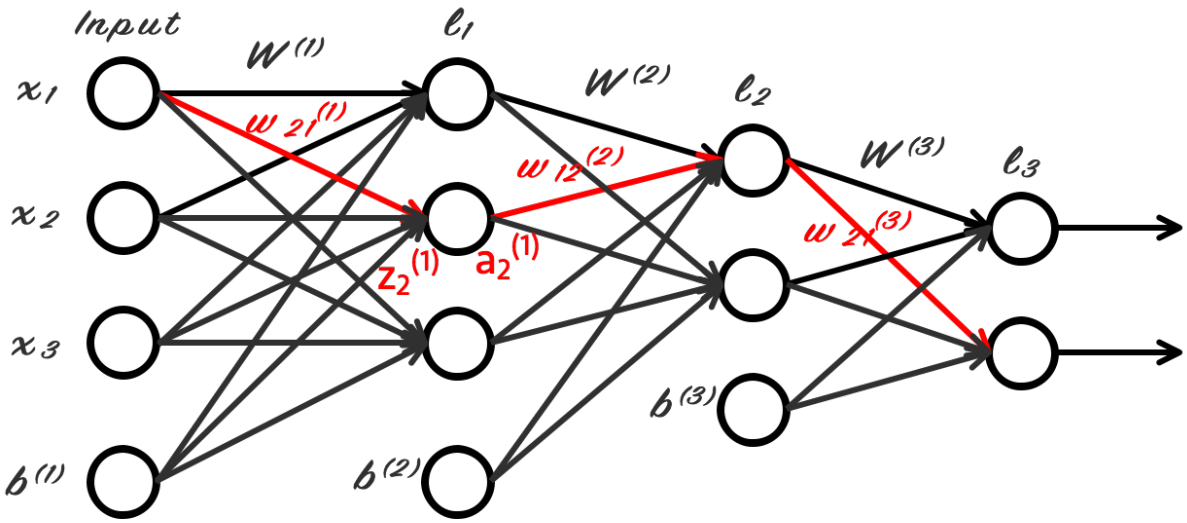
Deep Learning Technology and Application

Ge Li

Peking University

关于训练方法

前向传播计算



$$z^{(1)} = \begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \\ z_3^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{21}^{(1)} & w_{23}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{3} w_{1i}^{(1)} x_i + b_1^{(1)} \\ \sum_{i=1}^{3} w_{2i}^{(1)} x_i + b_2^{(1)} \\ \sum_{i=1}^{3} w_{3i}^{(1)} x_i + b_3^{(1)} \end{bmatrix}$$

批量前向计算

$$z^{(1)} = \begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \\ z_3^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{21}^{(1)} & w_{23}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{bmatrix} \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ x_{13} & x_{23} \end{bmatrix} + \begin{bmatrix} b_1^{(1)} & b_1^{(1)} \\ b_2^{(1)} & b_2^{(1)} \\ b_3^{(1)} & b_3^{(1)} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{3} w_{1i}^{(1)} x_{1i} + b_1^{(1)} & \sum_{i=1}^{3} w_{1i}^{(1)} x_{2i} + b_1^{(1)} \\ \sum_{i=1}^{3} w_{2i}^{(1)} x_{1i} + b_2^{(1)} & \sum_{i=1}^{3} w_{2i}^{(1)} x_{2i} + b_2^{(1)} \\ \sum_{i=1}^{3} w_{3i}^{(1)} x_{1i} + b_3^{(1)} & \sum_{i=1}^{3} w_{3i}^{(1)} x_{2i} + b_3^{(1)} \end{bmatrix}$$

用 A^{l+1} 表示与一个 Batch 对应的 $l+1$ 层所有神经元的输出值（其他变量含义相应可知），则：

$$Z^l = W^{(l)} A^{(l-1)} + B^{(l)} \qquad A^{(l)} = f(Z^{(l)})$$

批量权重更新

设一个 Batch 所对应的输入为：$X =$

$$\begin{bmatrix} x_{11} & x_{21} & \cdots & x_{m1} \\ x_{12} & x_{22} & \cdots & x_{m1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1n} & x_{2n} & \cdots & x_{mn} \end{bmatrix}$$

1. 设 $\Delta W_r^{(l)}$ 和 $\Delta b_r^{(l)}$ 分别为“当输入第 r 列数据时，第 l 层上的权重和偏置所对应的更新量”；则：
2. $\sum_{r=1}^{m} \Delta W_r^{(l)}$ 和 $\sum_{r=1}^{m} \Delta b_r^{(l)}$ 分别为“当输入完一个 Batch 的数据（X）时，第 l 层上的权重和偏置所对应的更新量的和”；
3. 这时，我们取上述更新量和的平均 $\frac{1}{m}\sum_{r=1}^{m} \Delta W_r^{(l)}$ 和 $\frac{1}{m}\sum_{r=1}^{m} \Delta W_r^{(l)}$，分别作为对第 l 层上的权重和偏置所应进行的更新。

批量梯度下降训练算法

设 X 为 M 列输入向量（一个 Batch），设 $\Delta W^{(l)}$ 和 $\Delta b^{(l)} = 0$ 分别为输入一个 Batch 后，第 l 层进行调整的权重和偏置的更新量；

1. 初始化：$\Delta W^{(l)} = 0$，$\Delta b^{(l)} = 0$
2. 计算权重和偏置的更新量矩阵（共 M 列），其中第 r 列分别为：$\Delta W_r^{(l)}$ 和 $\Delta b_r^{(l)}$；
3. 对上述两个矩阵，分别计算：$\frac{1}{m}\sum_{r=1}^{m}\Delta W_r^{(l)}$ 和 $\frac{1}{m}\sum_{r=1}^{m}\Delta W_r^{(l)}$
4. 利用上述结果进行权重更新：

$$W^{(l)} = W^{(l)} - \alpha(\frac{1}{m}\sum_{r=1}^{m}\Delta W_r^{(l)})$$
$$b^{(l)} = b^{(l)} - \alpha(\frac{1}{m}\sum_{r=1}^{m}\Delta b_r^{(l)})$$

循环执行上述过程，直到 Loss Function 输出值达到要求。

批量梯度下降的训练流程

设 X 为 M 列输入向量（一个 Batch），设 $\Delta W^{(l)}$ 和 $\Delta b^{(l)} = 0$ 分别为输入一个 Batch 后，第 l 层进行调整的权重和偏置的更新量；

1. 初始化：$\Delta W^{(l)} = 0$，$\Delta b^{(l)} = 0$
2. 计算权重和偏置的更新量矩阵（共 M 列），其中第 r 列分别为：$\Delta W_r^{(l)}$ 和 $\Delta b_r^{(l)}$；
3. 对上述两个矩阵，分别计算：$\frac{1}{m}\sum_{r=1}^{m}\Delta W_r^{(l)}$ 和 $\frac{1}{m}\sum_{r=1}^{m}\Delta W_r^{(l)}$
4. 利用上述结果进行权重更新：

$$W^{(l)} = W^{(l)} - \alpha(\frac{1}{m}\sum_{r=1}^{m}\Delta W_r^{(l)})$$
$$b^{(l)} = b^{(l)} - \alpha(\frac{1}{m}\sum_{r=1}^{m}\Delta b_r^{(l)})$$

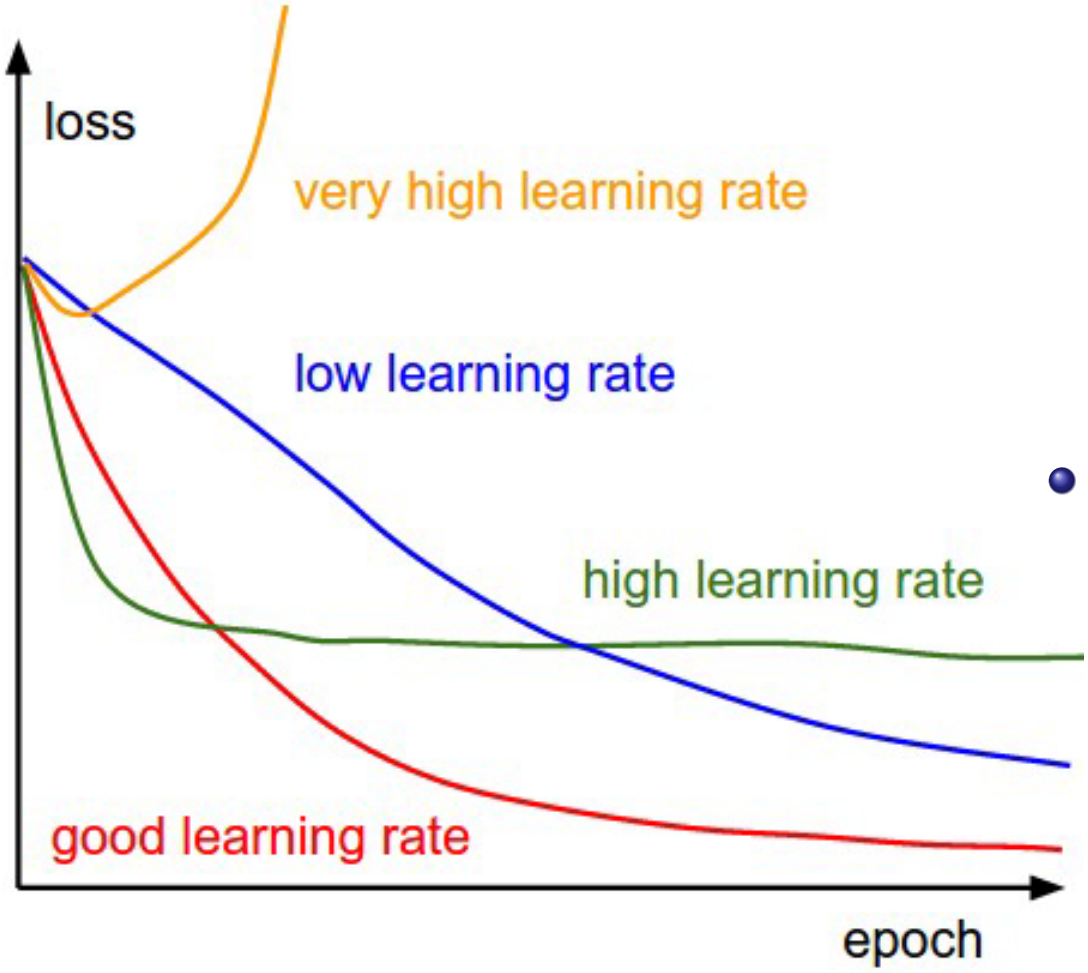
循环执行上述过程，直到达到收敛条件。

各种训练方法

1. 批量梯度下降（Batch GD）
   1. 每轮权重更新所有样本都参与训练；
   2. 迭代多轮，直到达到收敛条件；
2. 随机梯度下降（SGD）
   1. 每轮权重更新只随机选取一个样本参与训练；
   2. 迭代多轮，达到收敛条件便可终止；
3. 小批量梯度下降（Mini-Batch SGD）
   1. 每轮权重更新（随机）取一部分样本参与训练；
   2. 迭代多轮，直到满足收敛条件；

关于学习过程

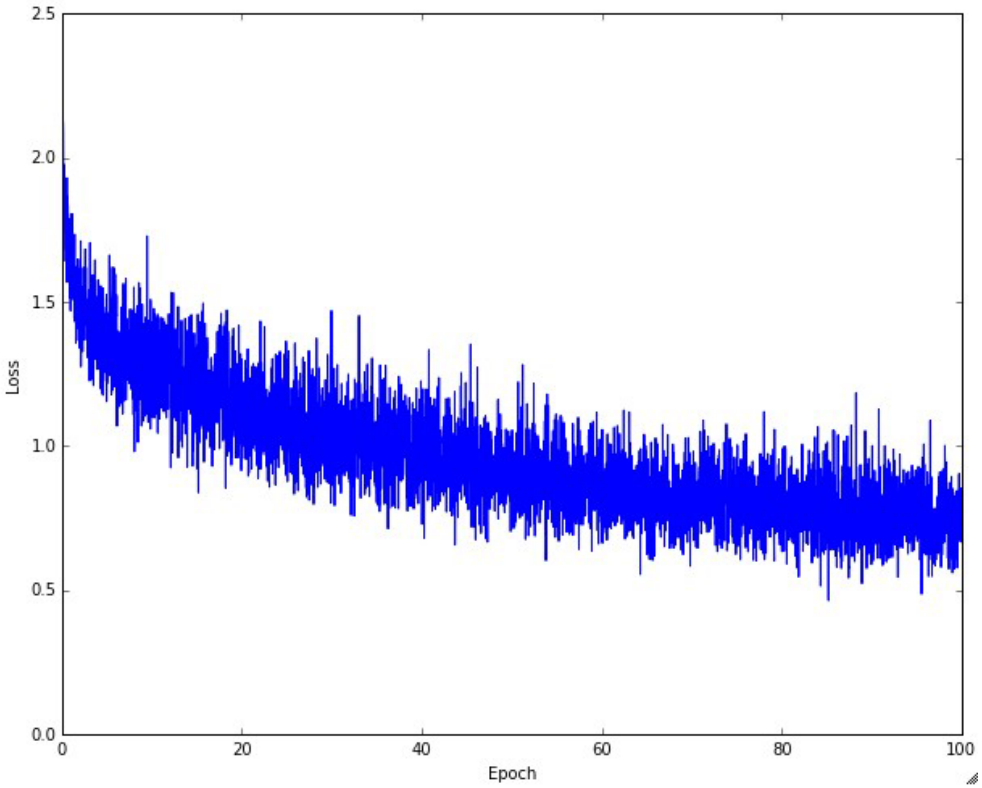
Learning Process



- Learning Rate 对学习过程有较大的影响。

Figure referred from: Stanford CS231n Convolutional Neural Networks for Visual Recognition

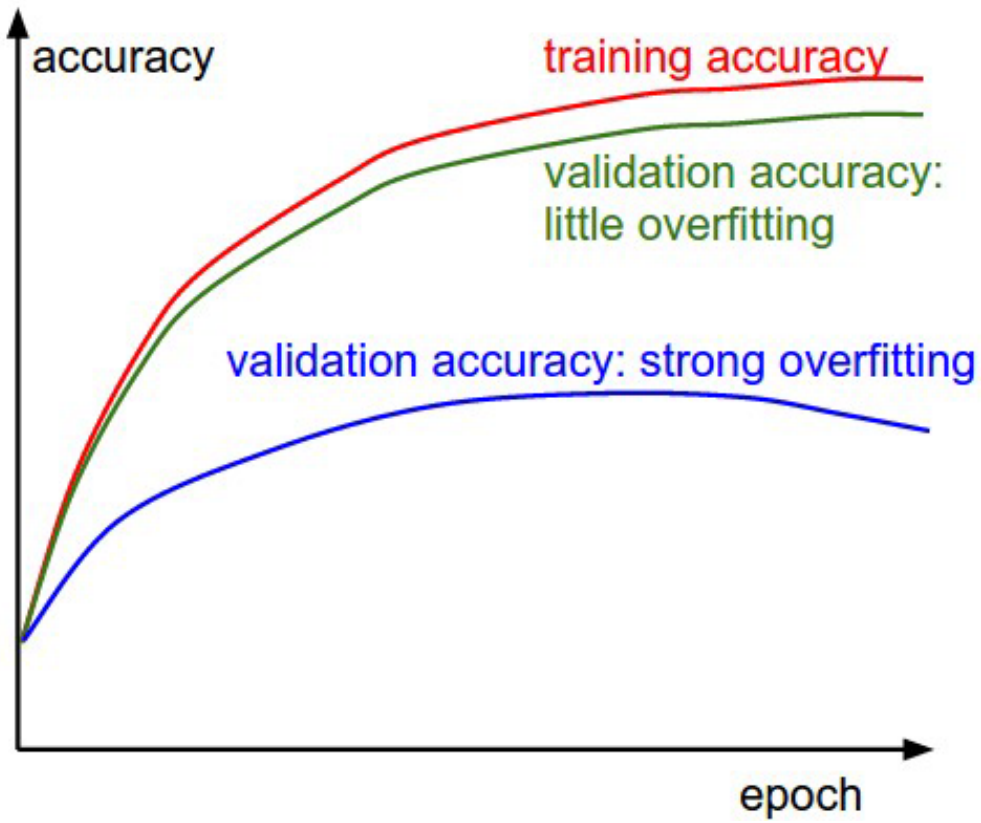
Learning Process



- Loss 曲线的摆动幅度与 Batch Size 相关
- Batch Size 越小，摆动幅度有可能越大
- Batch Size 越大，摆动幅度可能越小，Loss 越平稳

Figure referred from: Stanford CS231n Convolutional Neural Networks for Visual Recognition

Learning Process



- Training 曲线与 Validation 曲线之间的差距体现了 overfitting 的大小
- 若 Validation 曲线与 Training 曲线之间的差距比较大，则说明可能出现了 overfitting，这时应该考虑：
  - 增加或使用正则化方法如 L2 权重惩罚、加大 Dropout 等等；
  - 适当减小模型参数的数量，减小模型的 Capacity；
  - 增加训练数据；

Figure referred from: Stanford CS231n Convolutional Neural Networks for Visual Recognition

Training vs. Validation vs. Testing

Training - 训练 Validation - 验证 Testing - 测试

- Training Set
  - 数据集已知；
  - 用于对模型参数（Weights）进行训练（调整 Weights）
- Validation Set
  - 在训练过程中，数据集未知；
  - 不能用于调整模型参数（Weights）
  - 用于进行模型选择（根据 Validation Set 上的结果选择模型，选择模型的超参数）
- Testing Set
  - 数据机绝对未知；
    【这是区分 Validation Set 与 Testing Set 的重要标准】
  - 不可能根据 Test Set 调整任何参数；
  - 用于测试/评估模型的准确性；

Training Set vs. Validation Set

几种使用训练数据集的方法：

- 目的：
  - 面对现有的数据集，既要充分利用已有数据进行训练，又要对模型的泛化能力进行评估；
- 问题：
  - 如何划分训练数据集与验证数据集?
- 几种对划分训练数据集与验证数据集的方法
  - Exhaustive Cross-validation
    - Leave-one-out Cross-validation
    - Leave-p-out cross-validation
  - Non-exhaustive Cross-validation
    - Holdout method
    - K-fold Cross-validation
    - Monte Carlo Cross-validation

Cross-validation

- Exhaustive Cross-validation
  - Leave-p-out cross-validation
    - 每次从数据集中选取 p 个数据作为验证数据，其他数据作为训练数据；
    - 每次选取不同的 p 个数据作为验证数据，直至所有数据均被选择一次；
    - 因此，共需形成 C_n^p 组训练-验证数据；
    - 确保所有数据均曾进入训练，且能够在所有数据上进行验证；
  - Leave-one-out Cross-validation
    - Leave-p-out cross-validation 的特殊情况：$p = 1$;
    - 每次选取一个样本作为验证数据集，而将剩余样本作为训练数据集；
    - 几乎能够利用全部数据样本进行训练（每次只缺少一个样本）
    - 能够在所有样本上进行验证，结果相对准确；
    - 并且，不受训练数据样本划分情况的影响；
    - 缺点：在数据集较大时，训练规模较大，调参时机难掌握；

Cross-validation

- Non-exhaustive Cross-validation
  - Holdout method
    - 随机选取出一部分数据作为验证数据，将剩余数据作为训练数据;
    - 通常划分比例：1:4，1:3，1:2...
  - K-fold Cross-validation
    - 将数据集随机分为 K 份（通常选择 5 份、10 份...）
    - 每次选择其中一份作为验证数据集，将剩余 K-1 分作为训练数据集;
    - 重复 K 次，每次选择一份不同的数据集作为验证数据集;
    - 将最终 K 次评估结果的平均值作为模型精度的评估值;
    - Leave-one-out Cross-validation 也可以视为其特殊情况;
  - Monte Carlo Cross-validation

Cross-validation

- Non-exhaustive Cross-validation
  - Holdout method
  - K-fold Cross-validation
  - Monte Carlo Cross-validation
    - 每次随机将数据集划分成两部分，一部分作为训练数据集，一部分作为验证数据集；
    - 而后交换训练集与验证数据集，再进行验证；
    - 最终评估结果取所有划分的平均值；
    - 优点：不受 K 的限制；
    - 缺点：可能有些数据从未被选为训练集或验证集；

超参数的优化

- 不同学习任务需要不同的超参数。
- 常见超参数：
  - 优化参数：学习率相关参数，正则化系数，优化方法
  - 训练数据：Mini-batch 大小，初始化方法，数据预处理方法
  - 网络结构：神经元层数、神经元数量、激活函数
- 超参数优化
  - 超参数优化是一个组合优化问题，无法通过梯度下降方法来优化；
  - 评估一组超参数配置的时间代价非常高；

超参数的优化

- 网格搜索
  - 通过尝试所有超参数的组合来寻址合适一组超参数；
  - 如果超参数是连续的，则将超参数离散化，尝试离散化后的各种组合；
  - 一般不按照等间隔的方式进行离散化，需要根据超参数的可能分布进行离散化；
  - 网格搜索根据这些超参数的不同组合分别训练一个模型，然后测试这些模型在验证集上的性能，选取一组性能最好的配置。

超参数的优化

- 随机搜索（Random Search）
  - 不同超参数对模型性能的影响有很大差异, 有些超参数（比如正则化系数）对模型性能的影响有限，而有些超参数（比如学习率）对模型性能影响比较大。
  - 在这种情况下，采用网格搜索会在不重要的超参数上进行不必要的尝试。
  - 一种比较有效的方法是对超参数进行随机组合，然后在验证集上对超参数组合进行选取，这就是随机搜索；
  - 在实践中更容易实现，一般会比网格搜索更加有效。

上述两种搜索方法比较低效，因为不同超参数组合之间具有相关性，如果模型的超参数组合比较类似，没有必要进行重复搜索。于是，研究者提出了其他不同的搜索方法。

超参数的优化

- 贝叶斯优化
  - 根据当前已经试验的超参数组合，来预测下一个可能带来最大收益的组合。
  - 基于序列模型的优化（Sequential Model based Optimization，SMBO）
    - 假设超参数优化过程是一个高斯过程，则根据已有的 N 组试验结果来建模高斯过程；
    - 进而通过计算该高斯过程的后验分布来预测下一个参数组合；
  - 缺点：高斯过程建模需要计算协方差矩阵的逆，时间复杂度是 $O(n^3)$，不能很好地处理高维情况。
  - 需要一些更高效的高斯过程建模。

超参数的优化

- 动态资源分配
  - 逐次减半方法：在超参数优化中，每组超参数的评估代价比较高，如果可以在较早的阶段估计出一组配置的效果会比较差，那么我们就可以中止这组配置的评估，将更多的资源留给其它配置。

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In Artificial Intelligence and Statistics, pages 240–248, 2016.

- 神经架构搜索
  - 通过神经网络来自动实现网络架构的设计。
  - 一个神经网络的架构表示为一个变长的字符串，利用元学习的思想，利用一个控制器来生成另一个子网络的架构描述。
  - 控制器可以由一个循环神经网络来实现。控制器的训练可以通过强化学习来完成，其奖励信号为生成的子网络在开发集上的准确率。

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In Proceedings of 5th International Conference on Learning Representations, 2017.

Thanks.