Deep Learning Technology and Application

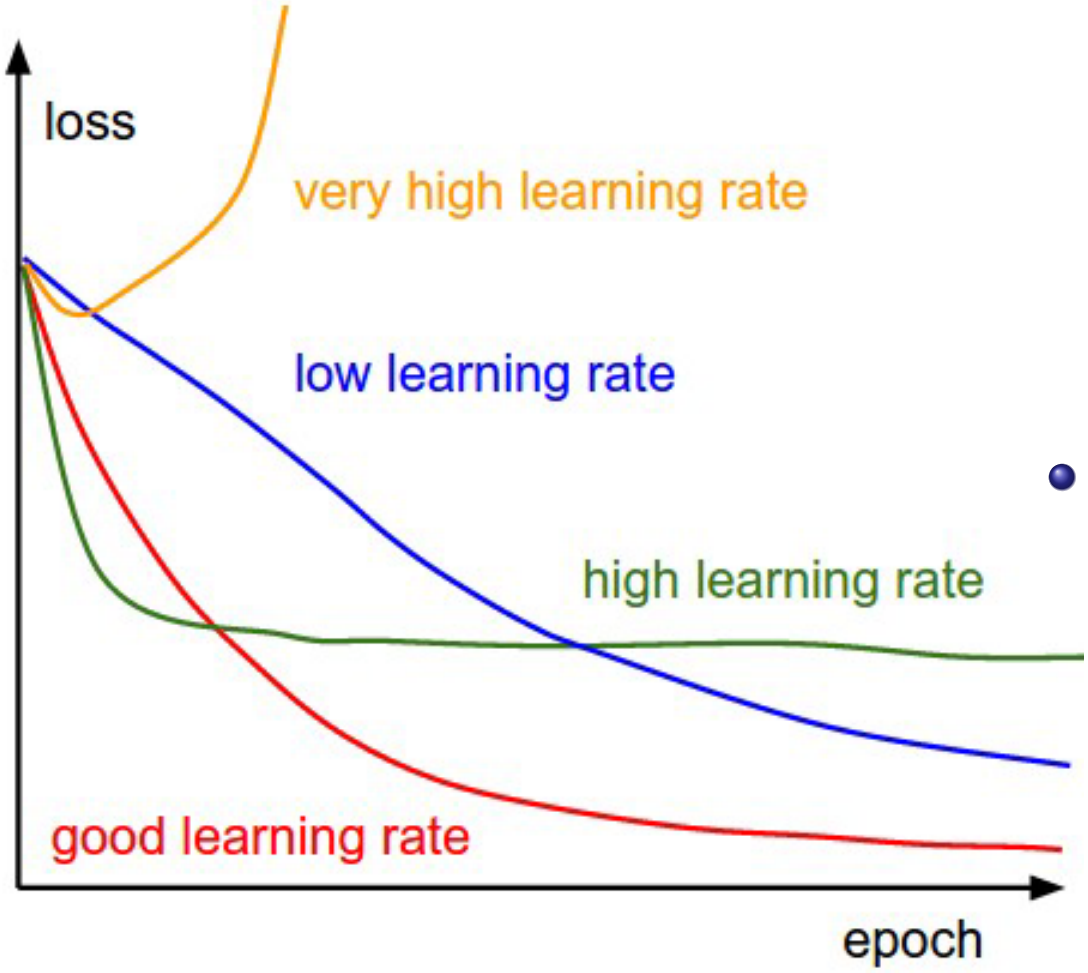
Ge Li

Peking University

Table of contents

关于学习过程

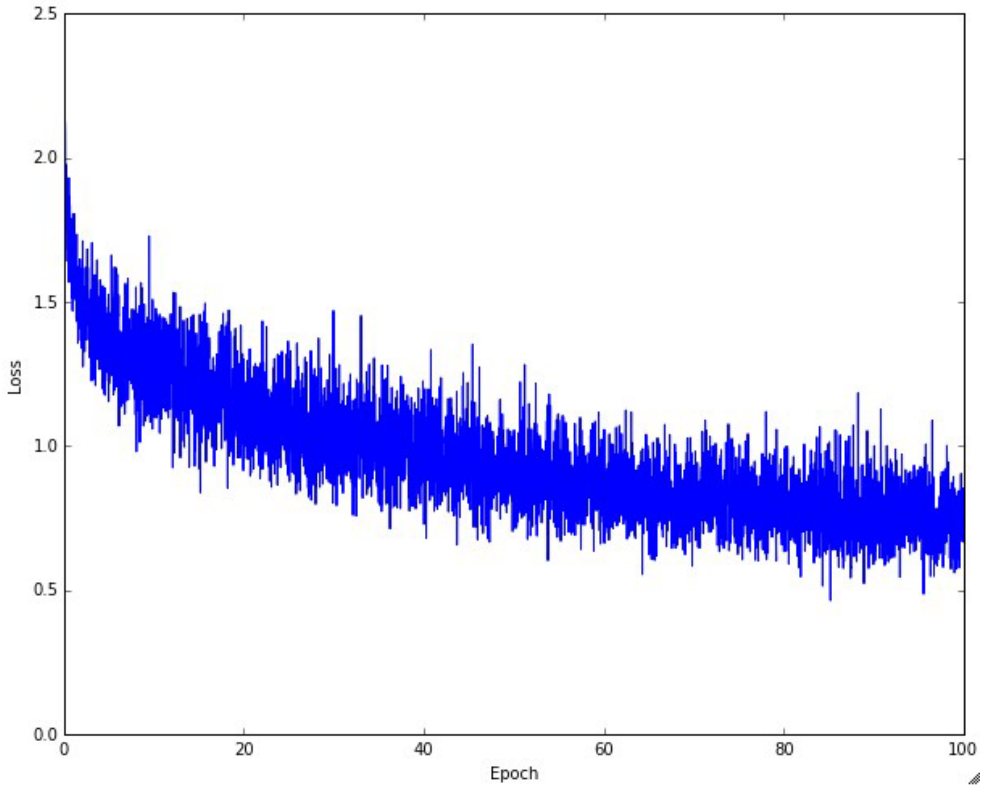
Learning Process



- Learning Rate 对学习过程有较大的影响。

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

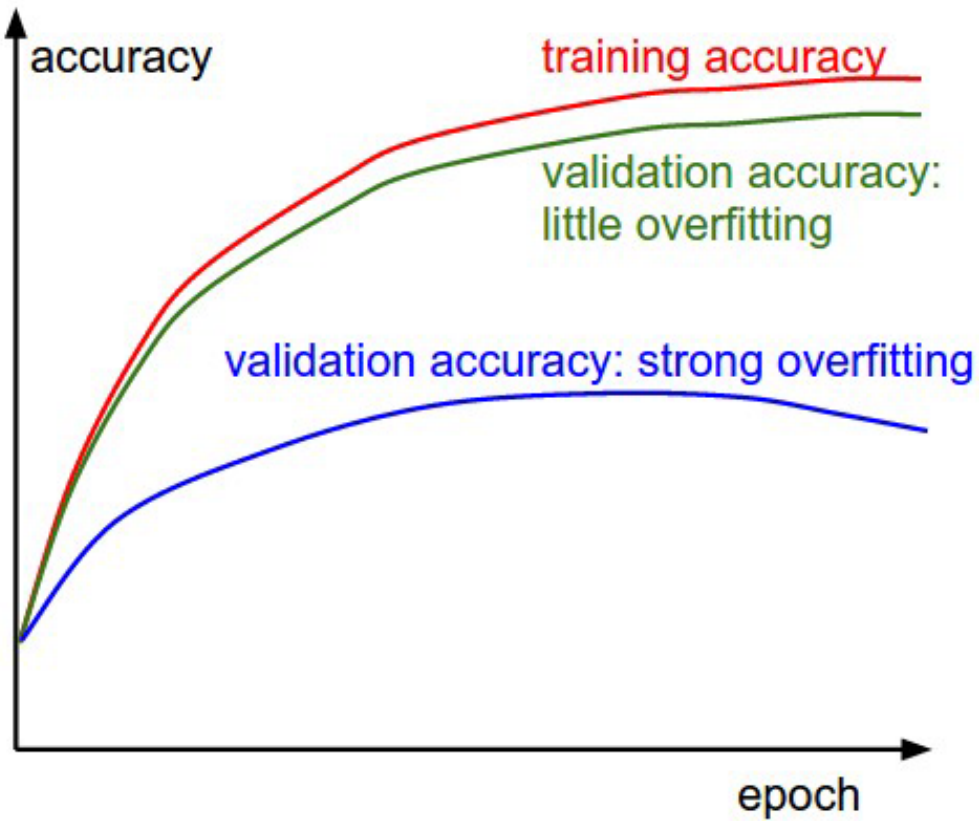
Learning Process



- 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。

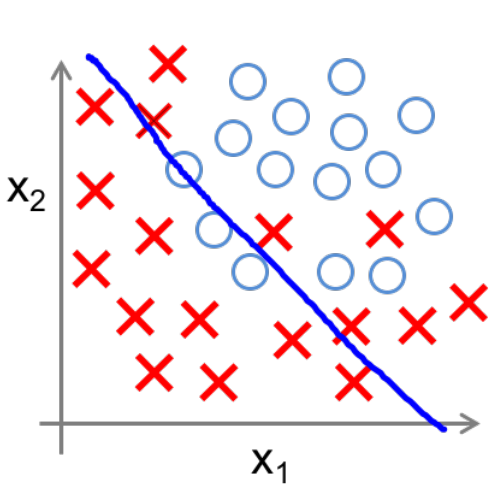
Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In Artificial Intelligence and Statistics, pages 240–248, 2016.

- 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.

关于正则化方法

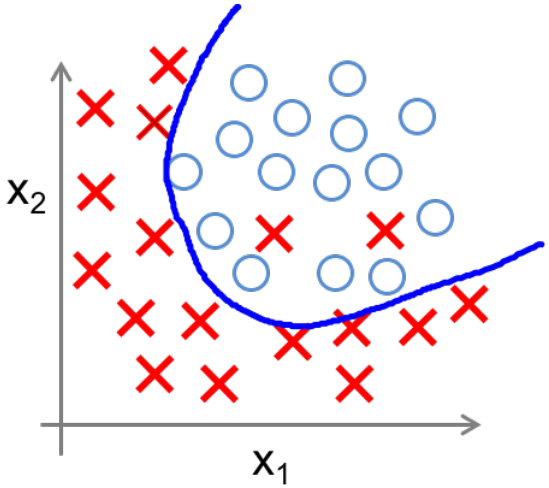
Regularization



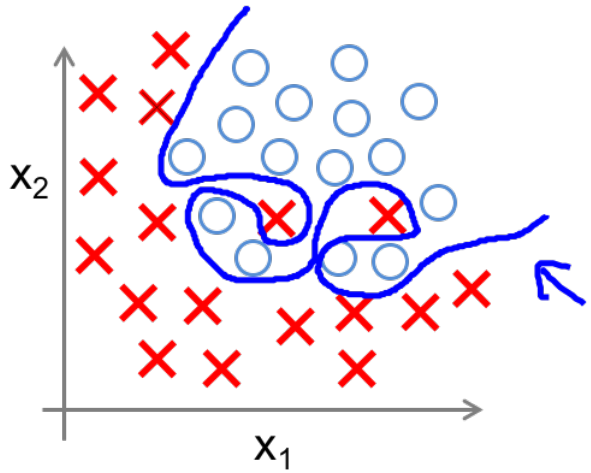
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

"Underfit"



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$



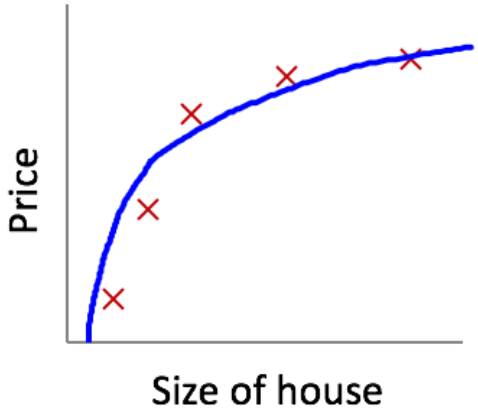
$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

"Overfit"

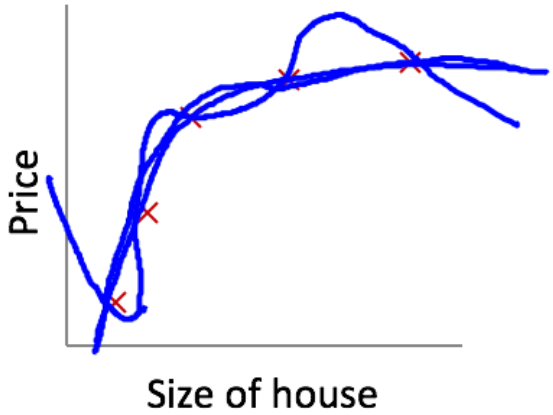
按照"奥卡姆剃刀"原则，在模型精度无明确不同的情况下，我们应该尽量选择尽可能简单的模型。

From: Andrew Ng, Machine Learning Course.

Regularization



$$\theta_0 + \theta_1 x + \theta_2 x^2$$



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Suppose we penalize and make θ_3, θ_4 really small.

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

$\theta_3 \approx 0$ $\theta_4 \approx 0$

对参数进行正则化处理的直观方法

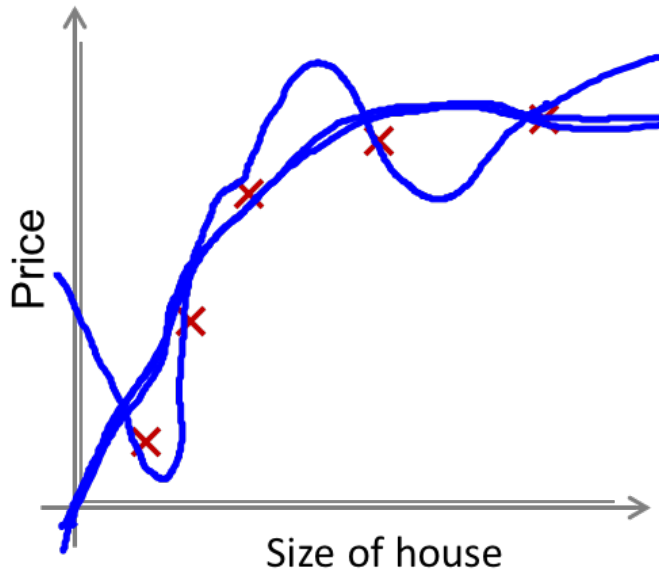
From: Andrew Ng, Machine Learning Course.

Regularization

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$$

regularization parameter

$$\min_\theta J(\theta)$$



From: Andrew Ng, Machine Learning Course.

范数

- 向量的范数
  - 1-范数：向量元素绝对值之和

$$\|w\|_1 = \sum_{i=1}^{N} |w_i|$$

  - 2-范数：欧几里得范数，即向量元素绝对值的平方和再开方

$$\|w\|_2 = \sqrt{\sum_{i=1}^{N} w_i^2}$$

  - p-范数：向量元素绝对值的 p 次方和的 1/p 次幂

$$\|w\|_p = (\sum_{i=1}^{N} |w_i|^p)^{\frac{1}{p}}$$

范数

- 矩阵的范数
  - 1-范数：将矩阵沿列方向取绝对值求和，然后选出最大值。

$$\|W\|_1 = \max_j \sum_{i=1}^{m} |w_{ij}|$$

  - 2-范数：对矩阵 W^TW 的最大特征值 λ_1 开平方

$$\|W\|_2 = \sqrt{\lambda_1}\lambda_1 \text{ 为 } W^TW \text{ 的最大特征值}$$

  - F-范数：即矩阵元素绝对值的平方和再开平方

$$\|W\|_F = (\sum_{i=1}^{m}\sum_{j=1}^{n} |w_{ij}|^2)^{\frac{1}{2}}$$

L2 Regularization

设未经正则化的 Loss Function 为：$J(w,b)$；
正则化项为：$\Omega(\theta)=\frac{1}{2}\|w\|_2^2$；
正则化参数为 λ；
则，正则化后的 Loss Function 为：

$$J(w,b)+\frac{\lambda}{2}\|w\|_2^2=J(w,b)+\frac{\lambda}{2}w^Tw$$

在反向传播的过程中：

$$w=w-\alpha\frac{\partial J(w,b)}{\partial w}\qquad b=b-\alpha\frac{\partial J(w,b)}{\partial b}$$

得：

$$w=w-\alpha\frac{\partial(J(w,b)+\frac{\lambda}{2}w^Tw)}{\partial w}\qquad b=b-\alpha\frac{\partial(J(w,b)+\frac{\lambda}{2}w^Tw)}{\partial b}$$

L2 Regularization

可见：

$$w = w - \alpha\left(\frac{\partial(J(w,b)}{\partial w} + \lambda w\right)$$

$$\text{即：} w = (1-\alpha\lambda)w - \alpha\frac{\partial(J(w,b)}{\partial w}$$

$$\text{而：} b = b - \alpha\frac{\partial J(w,b)}{\partial b}$$

- 可见，正则化方法对于 b 的更新没有影响；
- 而对于 w 则起到了以 $1-\alpha\lambda$ 幅度减小 w 的作用，这被称为 Weight Decay（权重衰减）.
- 即力图通过减小每一个变化因素的影响，以避免过拟合。

L1 Regularization

对于 Loss Function：$J(w,b)$；
正则化项为：$\Omega(\theta) = |w|_1 = \sum_i |w_i|$；
正则化参数为 λ；
则，正则化后的 Loss Function 为：

$$J(w,b) + \lambda|w|_1 = J(w,b) + \lambda\sum_i |w_i|$$

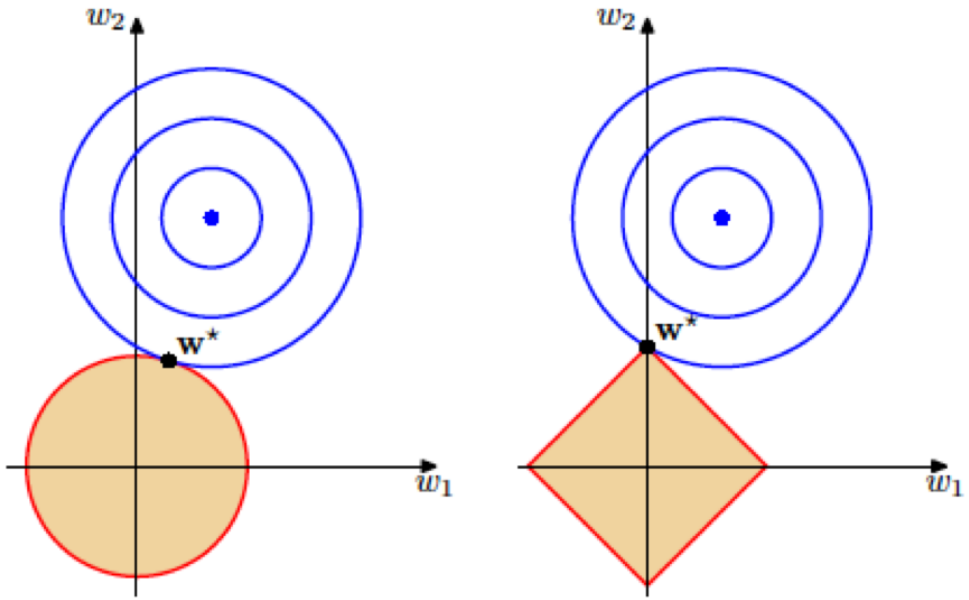
得：

$$w = w - \alpha\frac{\partial(J(w,b) + \lambda\sum_i |w_i|)}{\partial w}$$

$$w = w - \alpha\lambda sign(w) - \alpha\frac{\partial(J(w,b)}{\partial w}$$

其中，sign(w) 表示 w 的符号，当 w 为正时，更新后 w 变小；当 w 为负时，更新后 w 变大，可见其结果仍然是让 w 靠近 0；

Regularization



- 蓝色圆点，表示依据原 Loss 函数（未包含正则化项的函数的最优化点），红色线表示单纯正则化项的最优化结果集；
- 若同时对两者进行最优化，则最终结果必为两者的交集中的某个点；
- 原函数与右侧 L1 正则化的交集，更容易产生在“矩形的角”上；
- 原函数与右侧 L2 正则化的交集，则会产生在圆形的边上；

Figure referred from: Christopher M. Bishop, Pattern Recognition and Machine Learning

Regularization

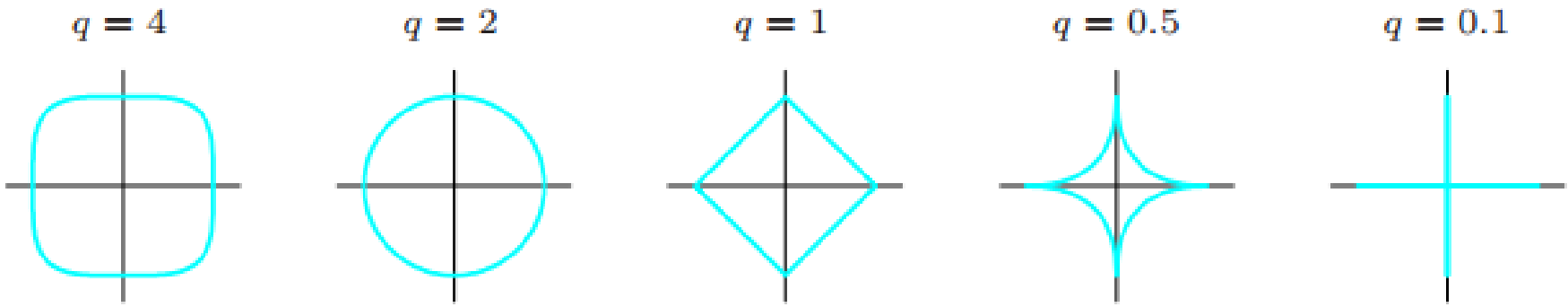
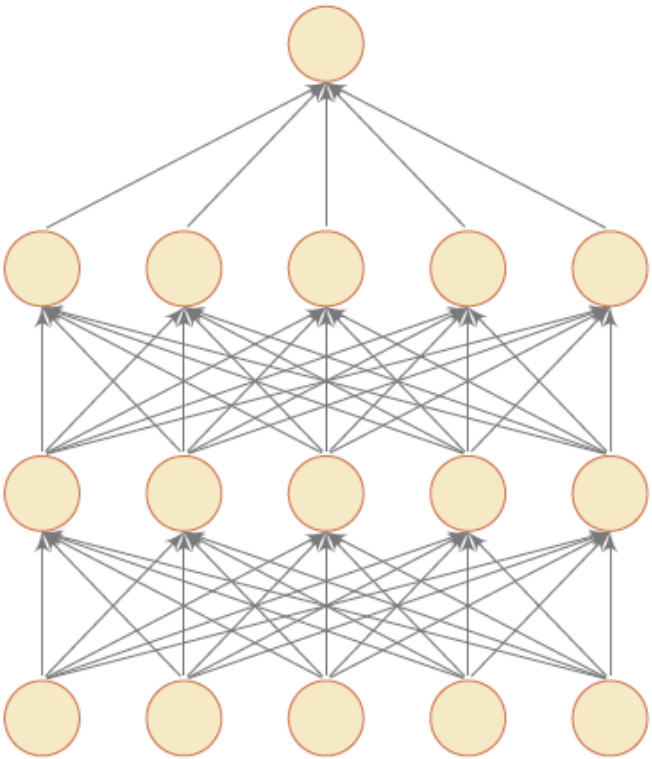
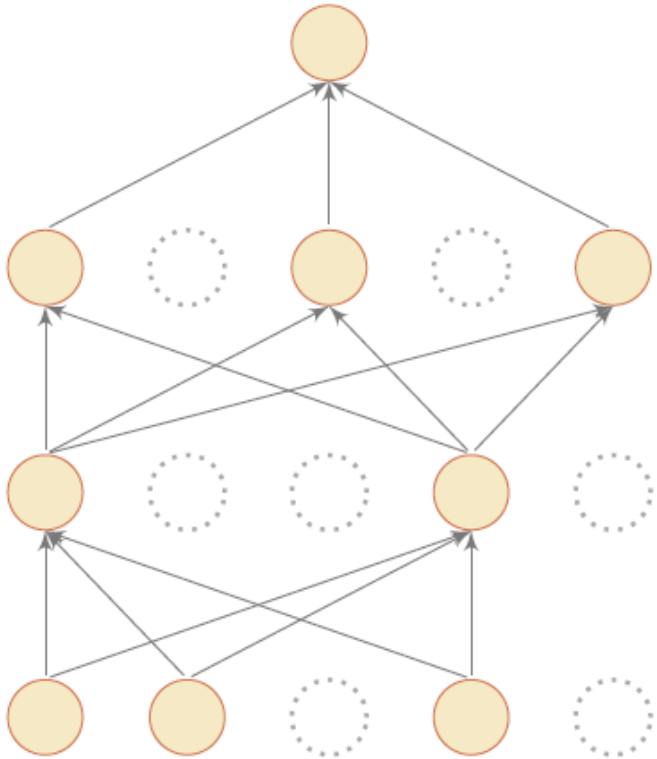


FIGURE 3.12. *Contours of constant value of* $\sum_j |\beta_j|^q$ *for given values of* q.

Dropout 处理



(a) 标准网络



(b) Dropout 神经网络

在训练中，随机丢弃一部分神经元（也丢弃对应的边）来避免过拟合

Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.

Dropout 处理

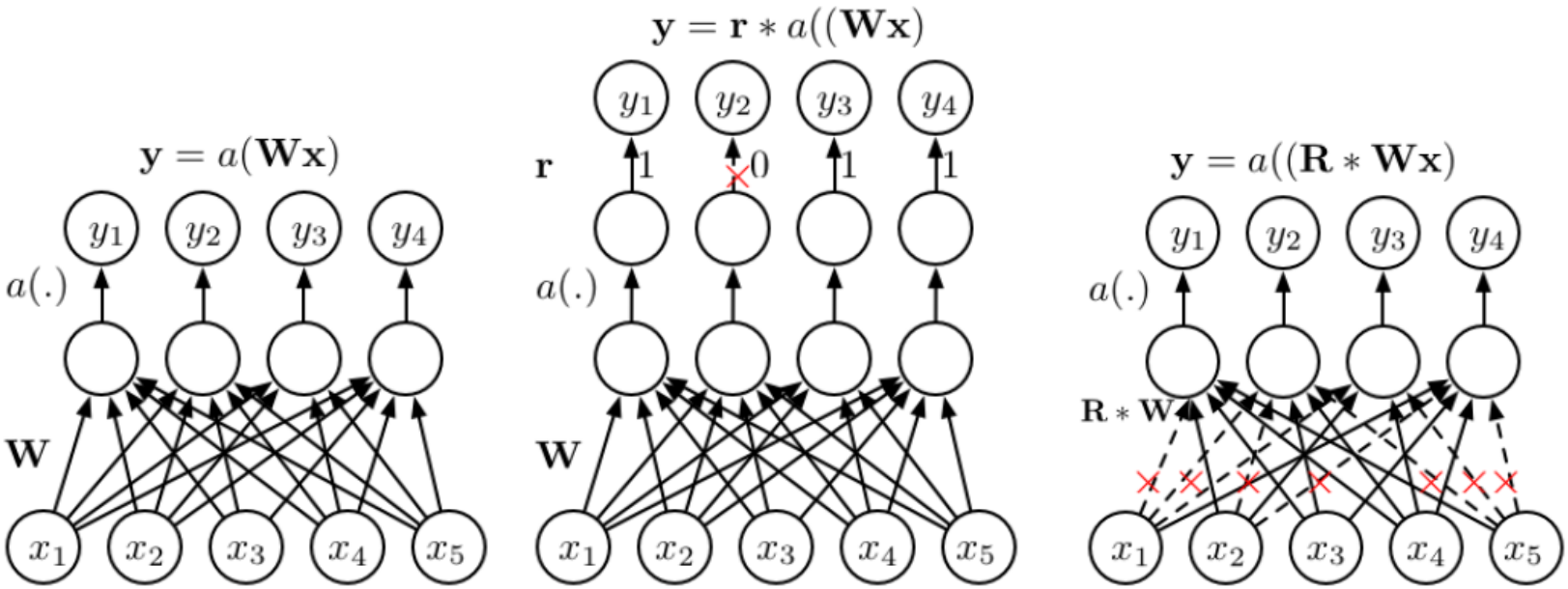
在全连接情况下，对隐藏层神经元的输出进行如下处理：

$$y = r * f(W^T x + b)$$

其中：

- x 为 n 维输入向量，$W \in R^{(d \times n)}$；
- r 为 d 维向量，且 $r_i \sim Bernoulli(p)$，p 为参数；
- Dropout 相当于使用多个网络进行训练，每做一次 dropout，相当于从原始的网络中采样得到一个子网络。
- 每次迭代都相当于训练一个不同的子网络，这些子网络共享原始网络的参数。
- 最终的网络可以看作是集成了指数级个不同网络的组合模型。

Dropout 处理



(a) No-Drop (b) DropOut (c) DropConnect

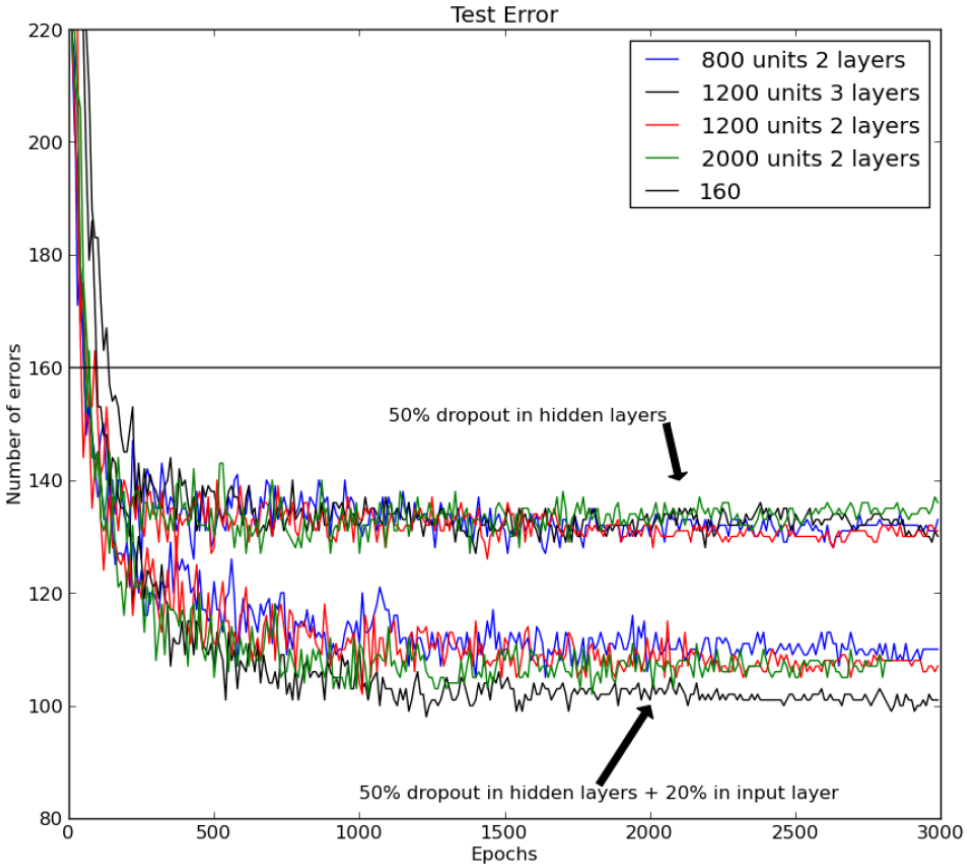
DropConnect 方法：

类似于 Dropout 方法，DropConnect 方法将权重矩阵 W 的某些值设置为 0；在全连接情况下，DropConnect 如下处理：

$$y = r * f(RW^T x + b) \text{其中：} R_{ij} \sim Bernoulli(p)$$

Dropout 处理

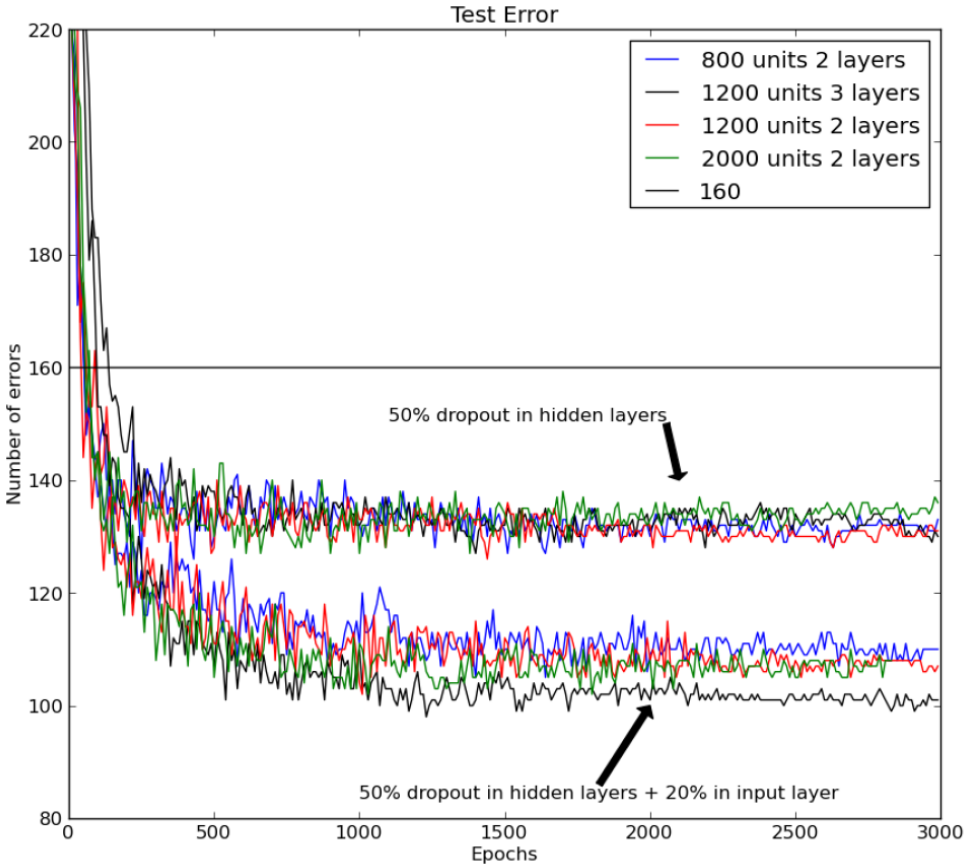
Dropout 的效果：



- 一般而言，对于隐藏层神经元，其 dropout 率取 0.5 时效果好，因为此时随机生成的网络结构更具多样性。
- 对于输入层神经元，其 dropout 率通常设为更接近 1 的数，使得输入变化不会太大。当对输入层神经元进行 dropout，相当于给数据增加噪声或进行数据增强。

Dropout 处理

Dropout 的效果：



- 在训练时，有一部分神经元被丢弃，而在测试时，所有的神经元都可以激活。因此，每个神经元训练时的净输入值会比测试时小。这会造成训练和测试时网络的输出不一致。为了缓解这个问题，在测试时需要将每一个神经元的输出都相应减小，相当于把不同的神经网络做了平均。

Dropout 处理

几种 Dropout 方法的改进：

- Fast Dropout[1]: perform fast Dropout training by sampling from or integrating a Gaussian approximation.
- Adaptive Dropout[2]: the Dropout probability for each hidden variable is computed using a binary belief network that shares parameters with the deep network.
- SpatialDropout[3]: extends the Dropout value across the entire feature map, it works well especially when the training data size is small.

[1] S. Wang, C. Manning, Fast dropout training, in: ICML, 2013.

[2] J. Ba, B. Frey, Adaptive dropout for training deep neural networks, in: NIPS, 2013.

[3] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, C. Bregler, Efficient object localization using convolutional networks, in: CVPR, 2015.

Thanks.