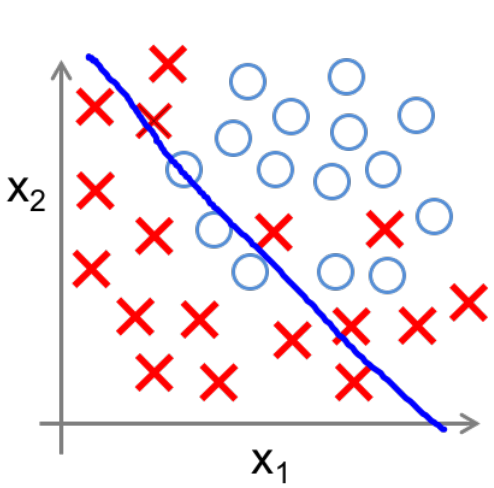
Deep Learning Technology and Application

Ge Li

Peking University

关于正则化方法

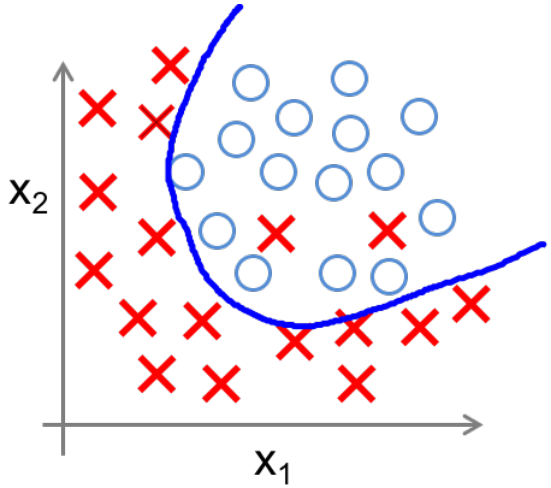
Regularization



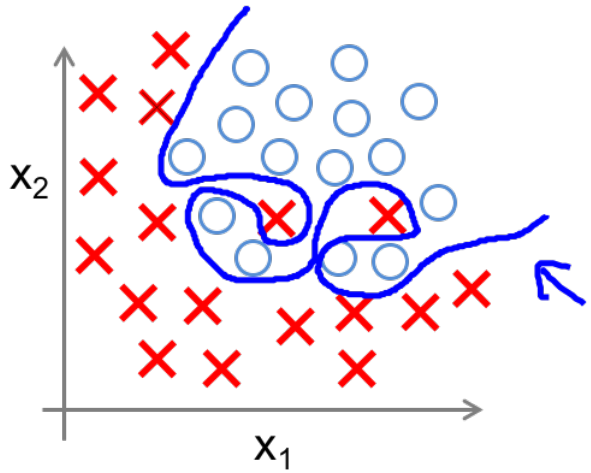
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

"Underfit"



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



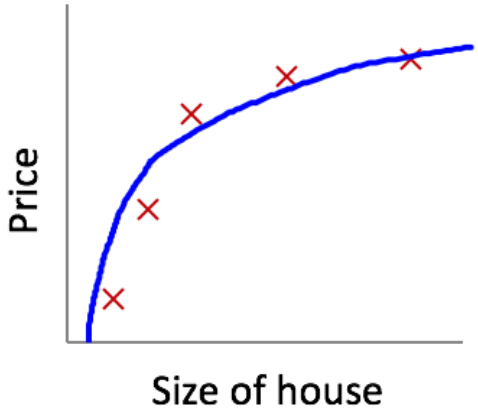
$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

"Overfit"

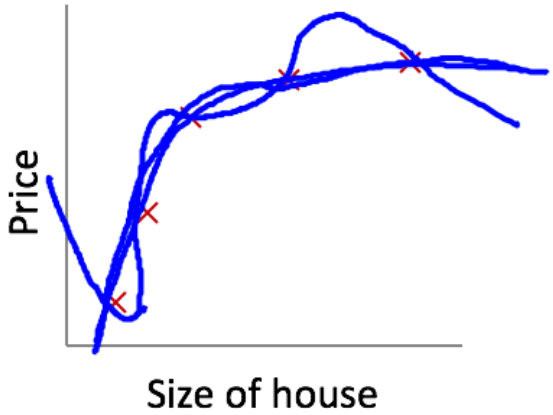
按照"奥卡姆剃刀"原则，在模型精度无明确不同的情况下，我们应该尽量选择尽可能简单的模型。

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

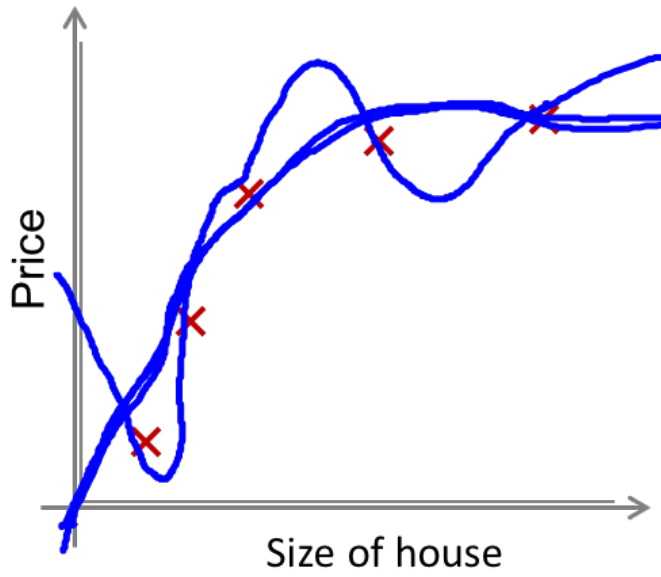
From: Andrew Ng, Machine Learning Course.

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：

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

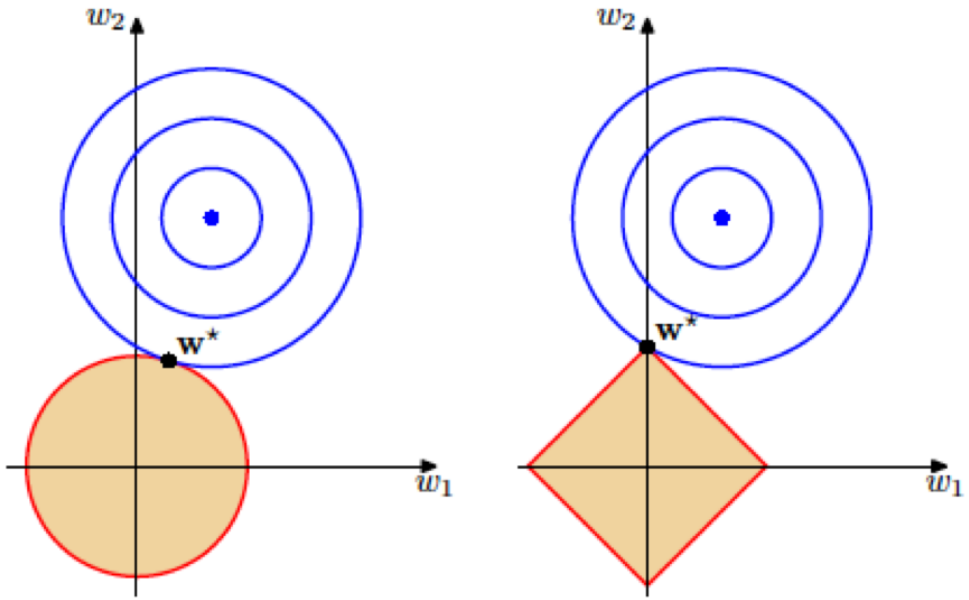
得：

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

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

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

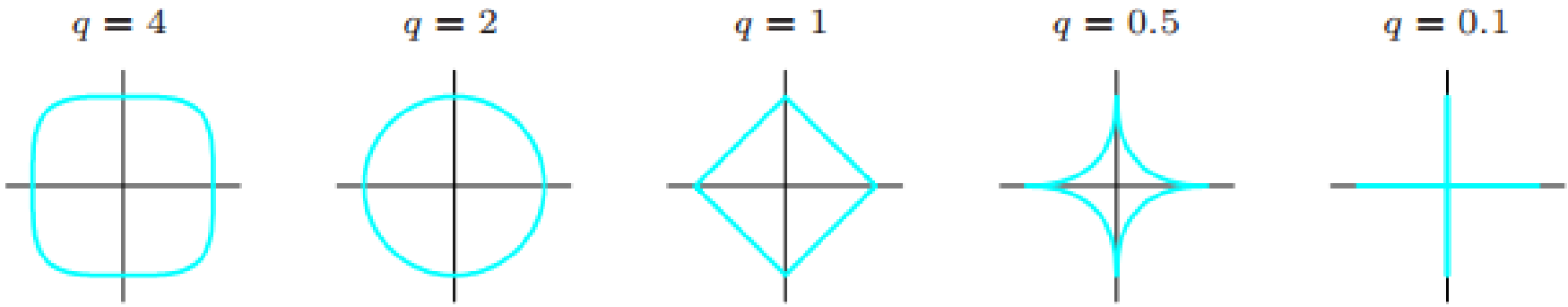


FIGURE 3.12. *Contours of constant value of* $\sum_j |\beta_j|^q$ *for given values of* q.

正则化处理

l_p 正则化处理：

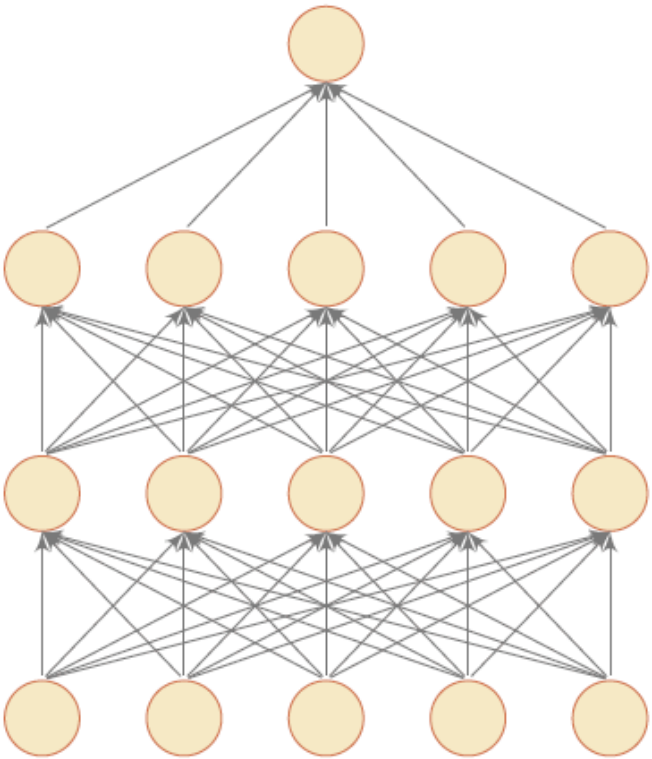
$$\hat{J}(W,b) = J(W,b) + \Omega(\alpha) = J(W,b) + \lambda \sum_j \parallel W_j \parallel |_p^p$$

其中：

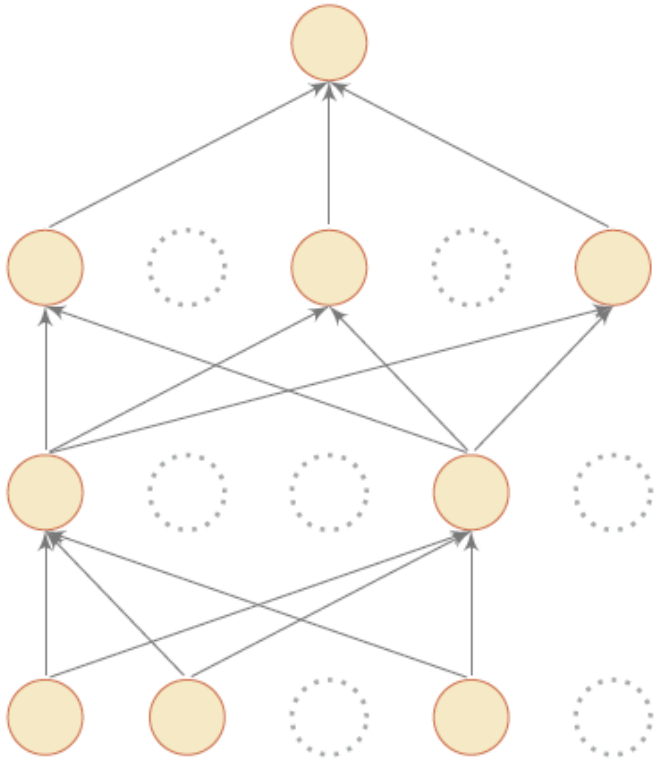
- 若 $p > 1$，则 l_p 为突函数，相当于权值衰减；
- 若 $p < 1$，则 l_p 的作用相当于稀疏化处理，最小化代价函数使其趋向于零；

Dropout 处理

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

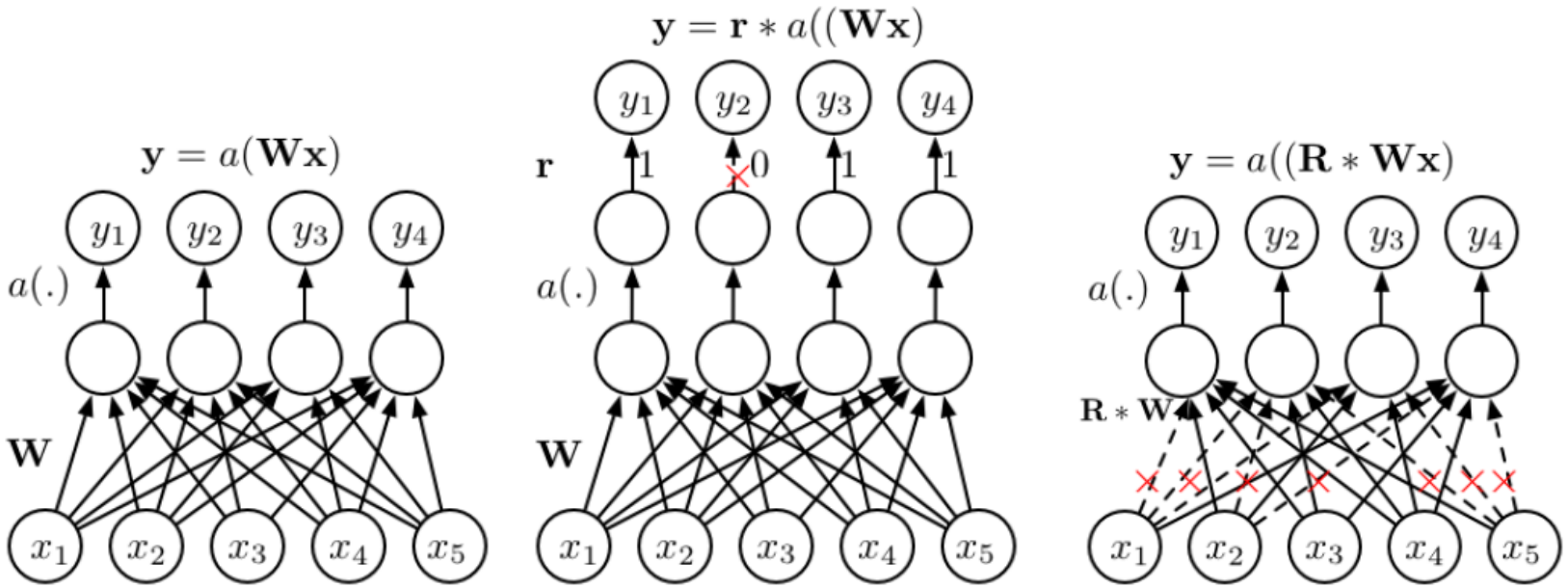
在全连接情况下，对隐藏层神经元的输出进行如下处理：

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

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

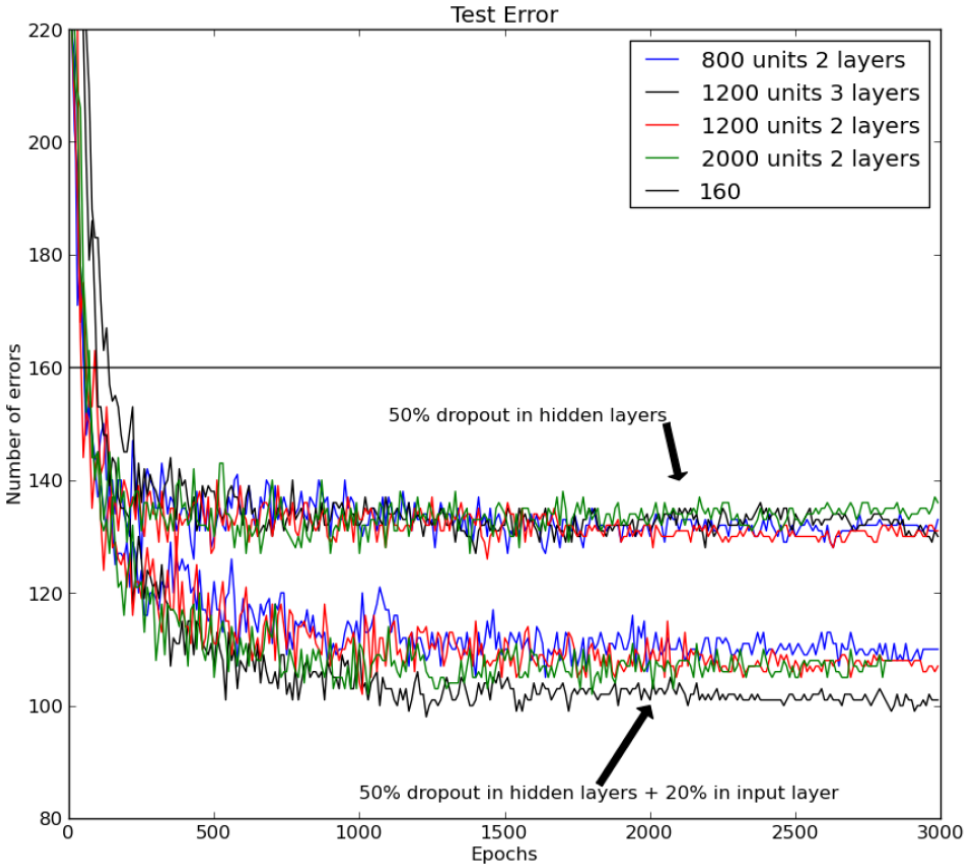
DropConnect 方法：

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

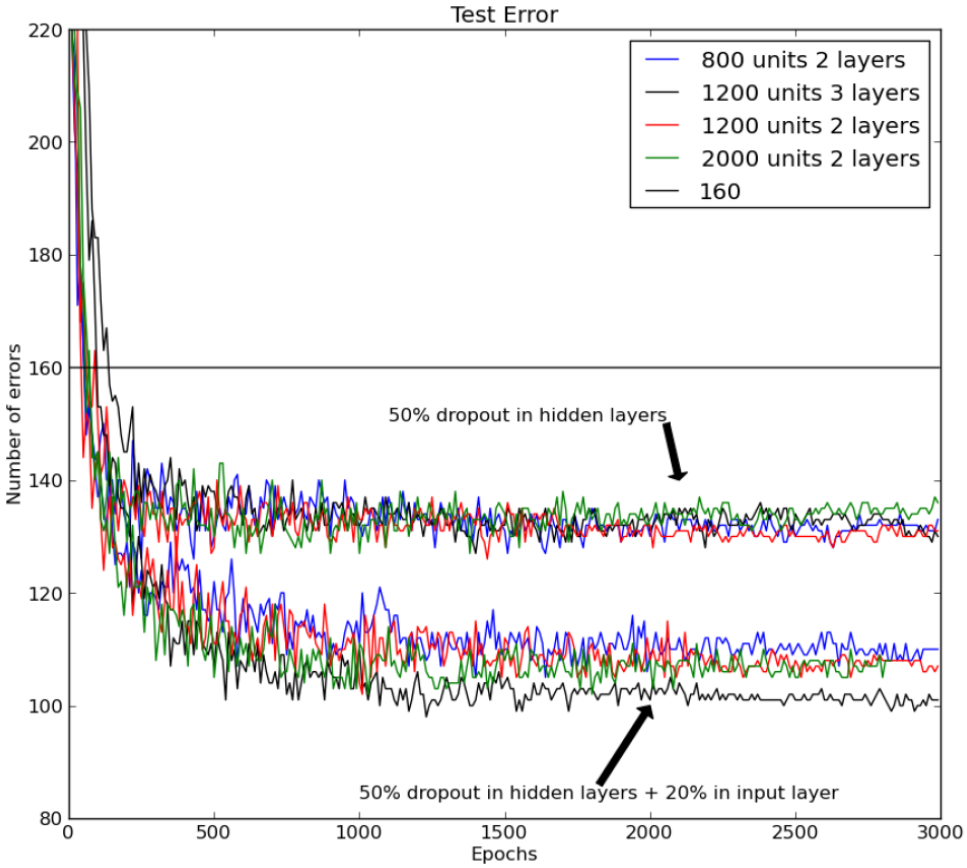
Dropout 的效果：



- 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.