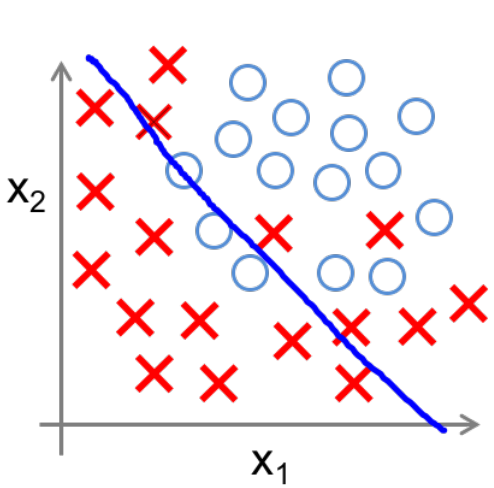
Deep Learning Technology and Application

Ge Li

Peking University

关于正则化方法

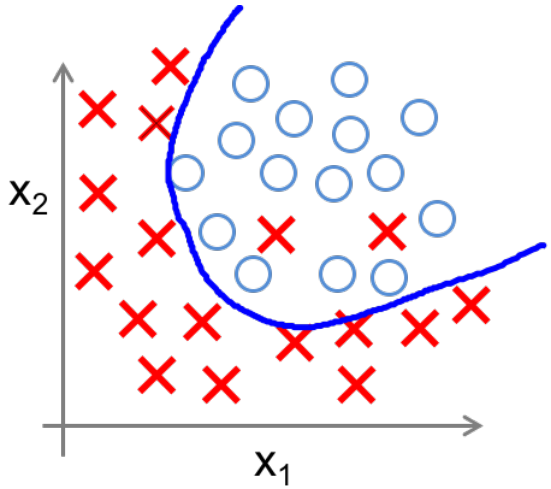
Regularization



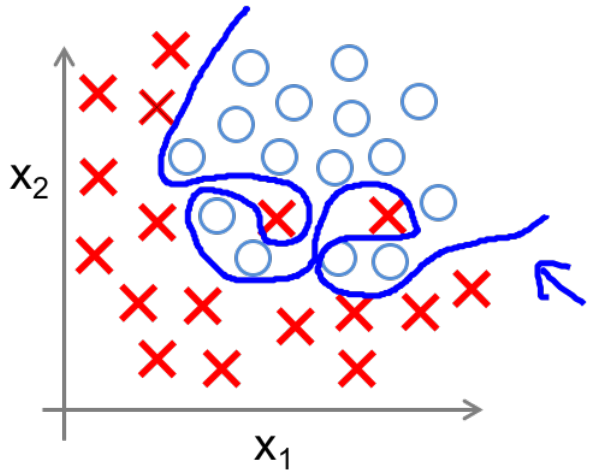
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

"Underfit"



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

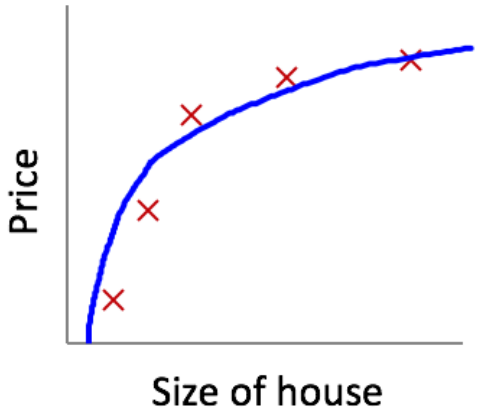


$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

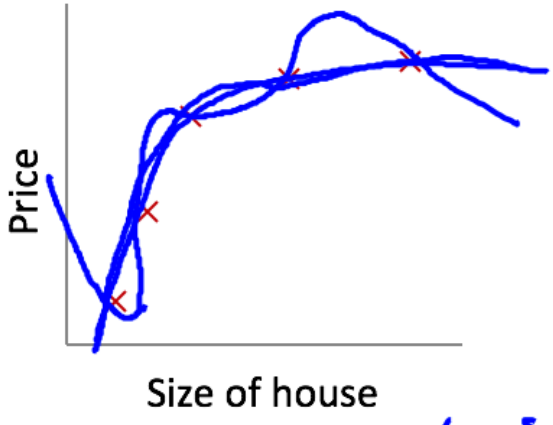
"Overfit"

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.

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

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

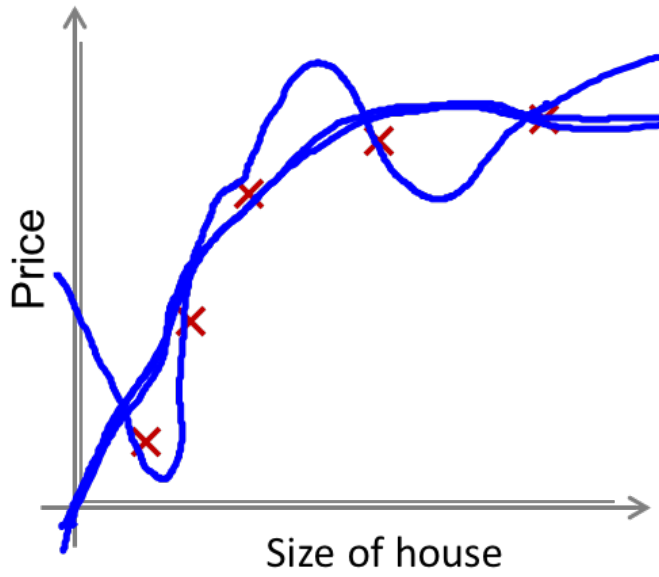
From: Andrew Ng, Machine Learning Course.

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.

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.

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：

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

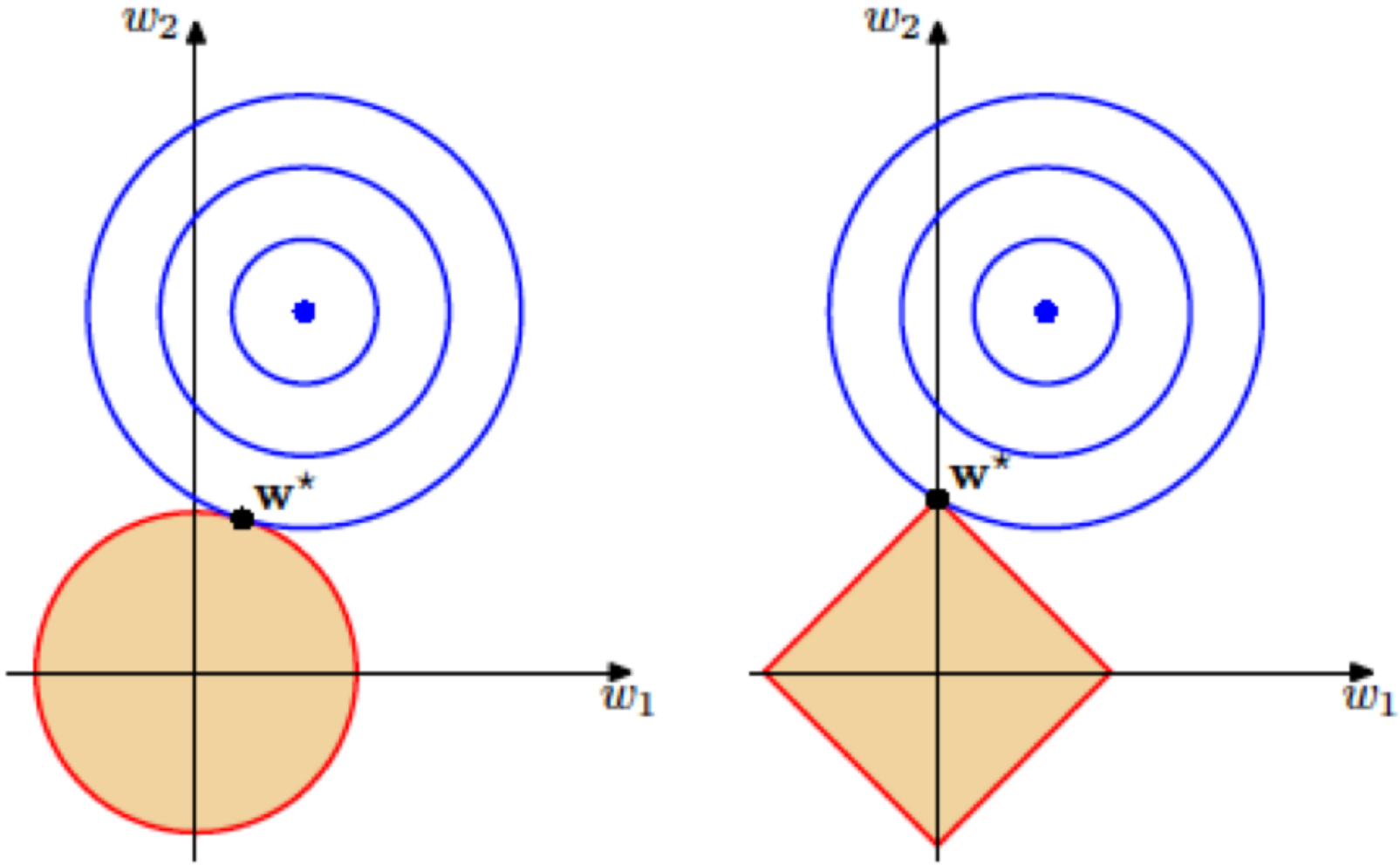
得：

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

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

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



From: Christopher M. Bishop, Pattern Recognition and Machine Learning

卷积神经网络的正则化处理

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

l_p 正则化处理：

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 方法是一种防止 overfitting 的有效方法，它相当于使用多个网络进行训练；

"Dropout can prevent the network from becoming too dependent on any one (or any small combination) of neurons, and can force the network to be accurate even in the absence of certain information."

Dropout 处理

几种 Dropout 方法的改进：

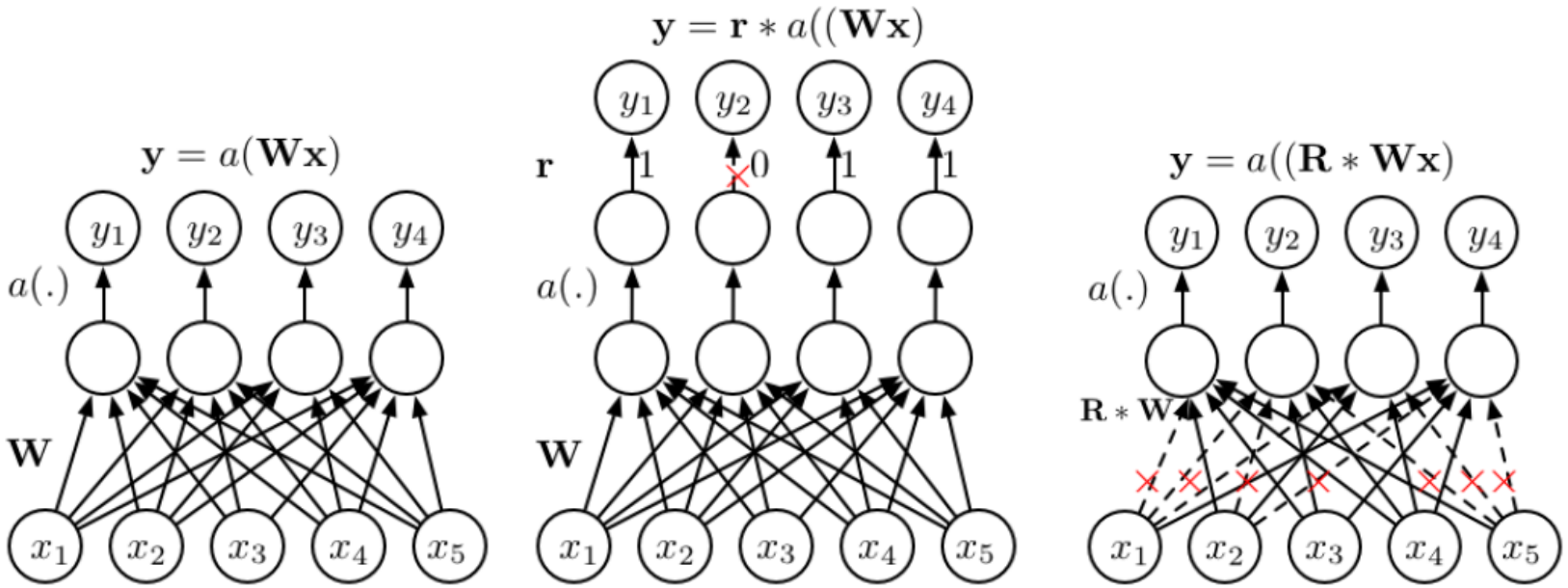
- Fast Dropout[1]: perform fast Dropout training by sampling from or integrating a Gaussian approximation.
- Adaptive Dropout[2]: the Dropout probability for each hidden variable is computed using a binary belief network that shares parameters with the deep network.
- SpatialDropout[3]: extends the Dropout 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.

Dropout 处理



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

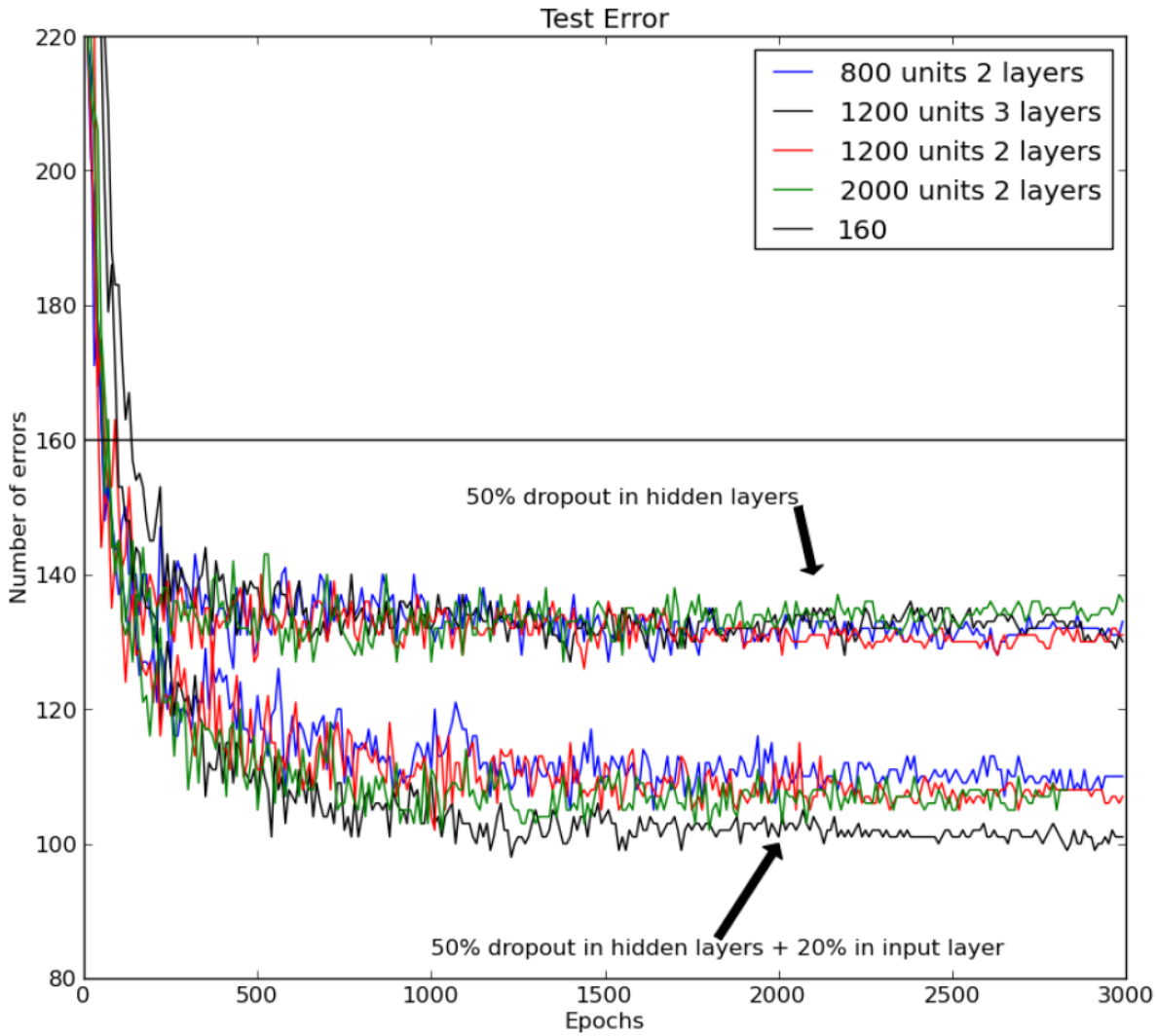
DropConnect 方法：

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：

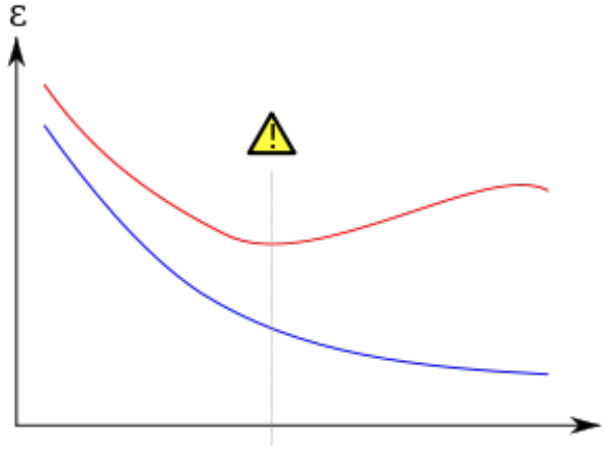


Overfitting vs. Regularization

Overfitting vs. Regularization

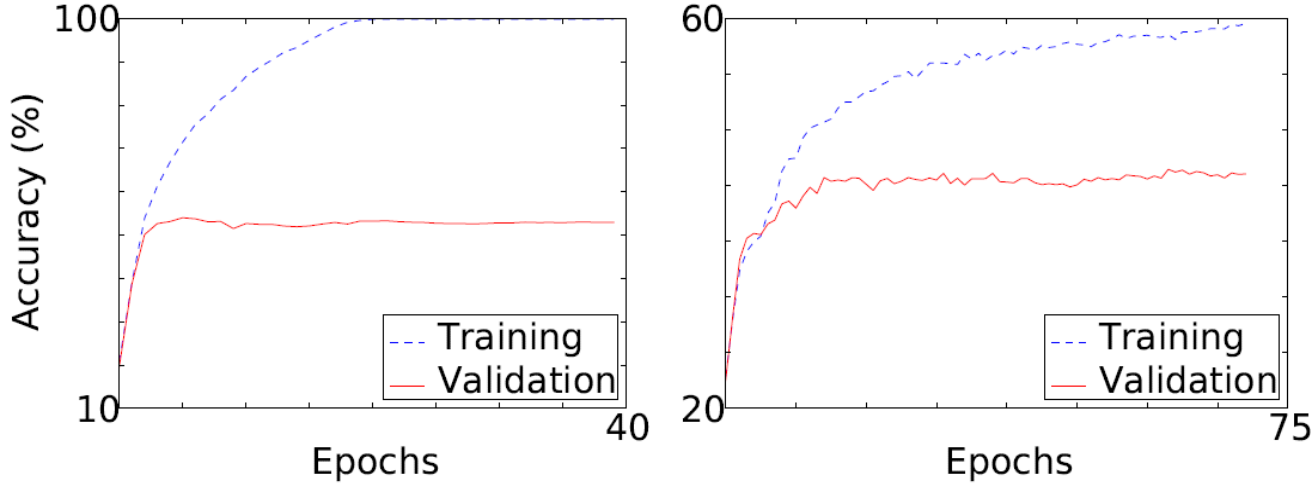
- Neural networks are very prone to overfitting
  - even with low dimensional embeddings (e.g., 50-dimensional), and shallow architectures (e.g., one convolutional layer).
  - Large numbers of parameters contribute much to the VC-dimension, and thus may lead to poor generalization.

If the validation error increases(positive slope) while the training error steadily decreases(negative slope) then a situation of overfitting may have occurred.



Overfitting vs. Regularization

- Training and validation accuracy for relation extraction using the convolutional neural network (Collobert and Weston, 2008).



- Similar phenomenon also appears in sentiment analysis with the recursive neural network (Socher et al., 2011), where training accuracy surpasses validation accuracy by more than 20%.

Overfitting vs. Regularization

- To address the problem of overfitting,
  - A prevailing approach in the machine learning literature is to add regularization.
    - The idea is to reduce model capacity, so that the model is less prone to overfitting.
- Typical regularization approaches
  - penalizing norm of parameters (Bishop, 2006)
  - dropout (Srivastava et al., 2014)

[BOOK] **Pattern recognition** and **machine learning**
CM **Bishop** - **2006** - sh.st
Machine learning is a key technology in bioinformatics, especially in the analysis of" big data" in bioinformatics. This course gives an overview of basic concepts, algorithms, and applications in **machine learning**, including topics such as classification, linear regression, **...**
Cited by 16543 Related articles All 30 versions Cite Save More

Regularization Strategies

- Penalizing weights
  - E: the cross-entropy error for classification;
  - R: a regularization term.
  - The overall cost function:

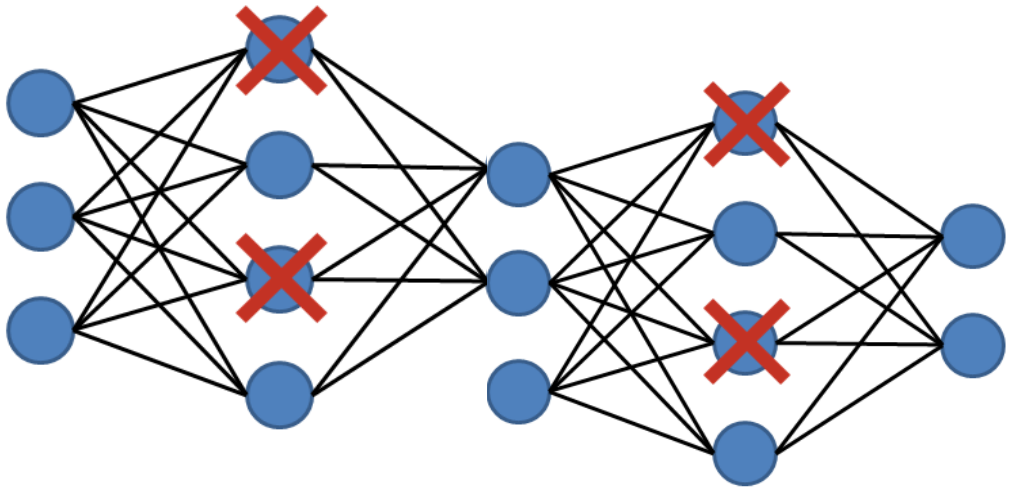
$$J = E + \lambda R$$

$$R = \|W\|^2$$

λ is the coefficient.

Regularization Strategies

- Dropout
  - each neural node is set to *0* with a predefined dropout probability *p* during training;
  - when testing, all nodes are used, with activation multiplied by *1-p*.



Regularization Strategies Comparison

- Regularization Strategies

$$J = E + \lambda R$$

  - Penalizing ℓ_2-norm of weights

$$R = \|W\|^2$$

  - Penalizing ℓ_2-norm of embedding.

$$R = \|\Phi\|^2$$

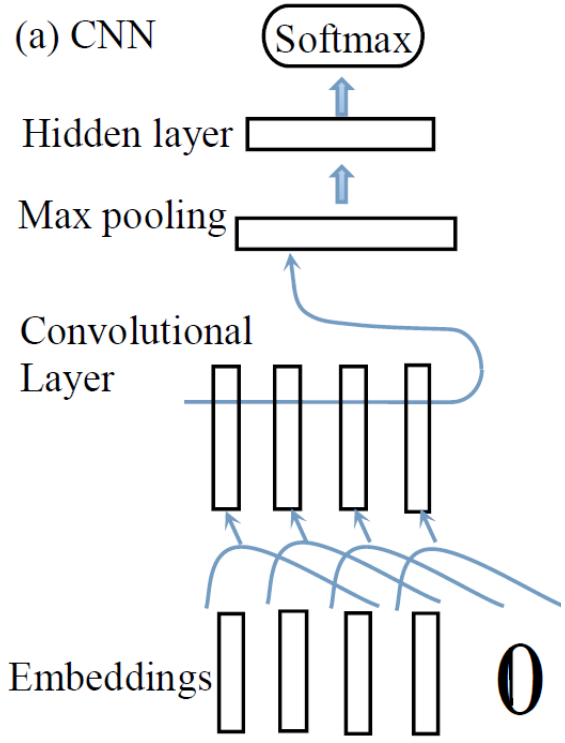
  - Re-embedding words.
    - pretrained embeddings $\mathbf{\Phi}_0$ and fine-tuned embeddings $\mathbf{\Phi}$.

$$R = \|\Phi_0 - \Phi\|_2^2$$

  - Dropout (Srivastava et al., 2014).

Regularization Strategies Comparison

- Experiment I: Relation Extraction with CNN
  - Dataset: SemEval-2010 Task 8
  - Model: Collobert ‘s CNN (2008)



SemEval-2010 Task 8
- Each sentence is tagged with two entities.
- The goal is to classify the relationship between these two entities.
- The dataset contains 8,000 samples with additional 3,000 for testing.

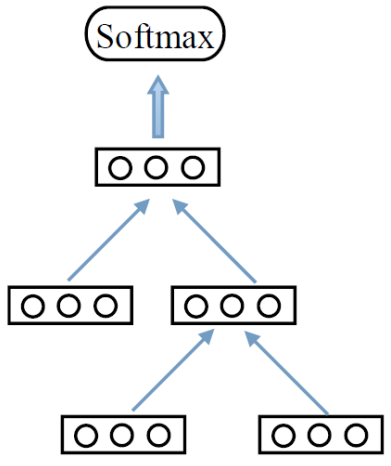
Regularization Strategies Comparison

- Experiment II: Sentiment Analysis with RNNs
  - Dataset: Stanford treebank
  - Model： Recursive neural networks, Socher (2011 and 2012)

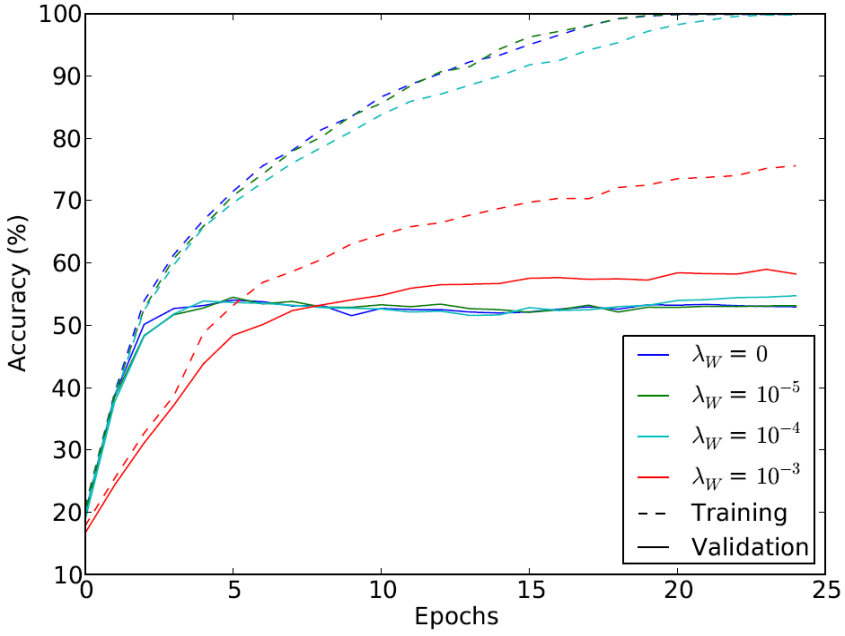
Stanford treebank

- consists of 152,847 human labeled sentences and phrases for training
- Additional 1101/2210 sentences for validation and testing.
- Target labels include 5 classes:
  - strongly positive, positive, neutral, negative, and strongly negative.

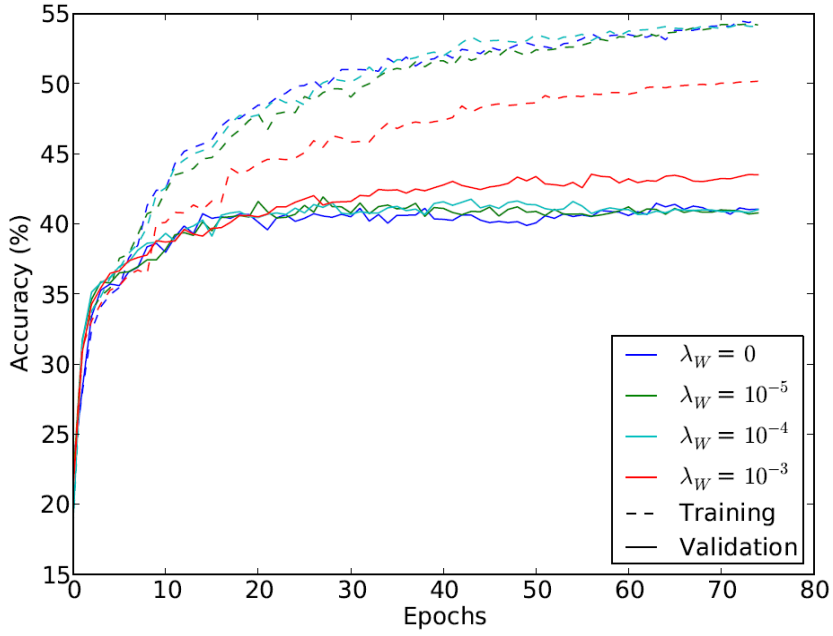
b) RNN



Regularization Strategies Comparison



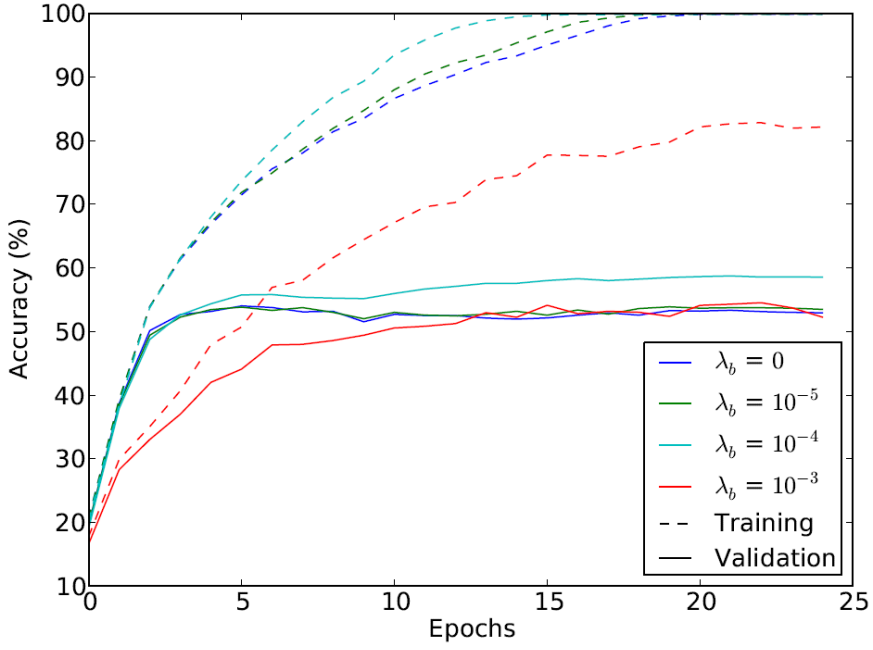
(a) Penalizing weights in Experiment I.



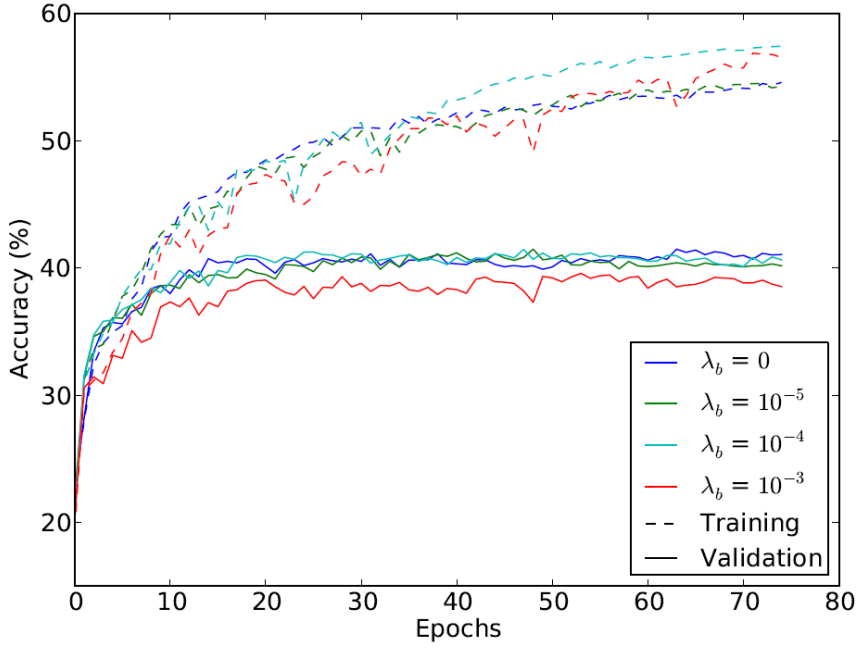
(b) Penalizing weights in Experiment II.

1. Penalizing l_2-norm of weights does help generalization;

Regularization Strategies Comparison



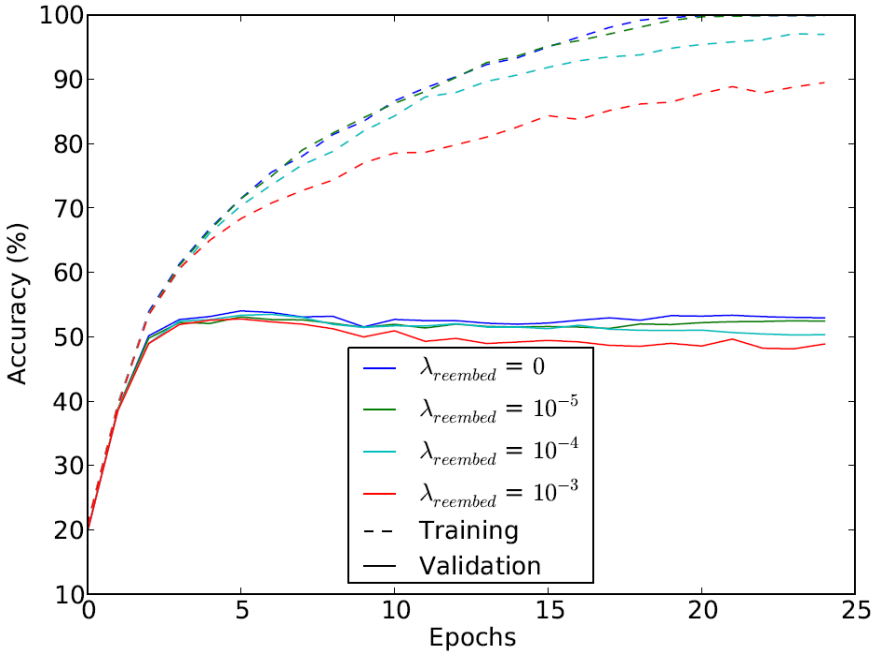
(c) Penalizing embeddings in Experiment I.



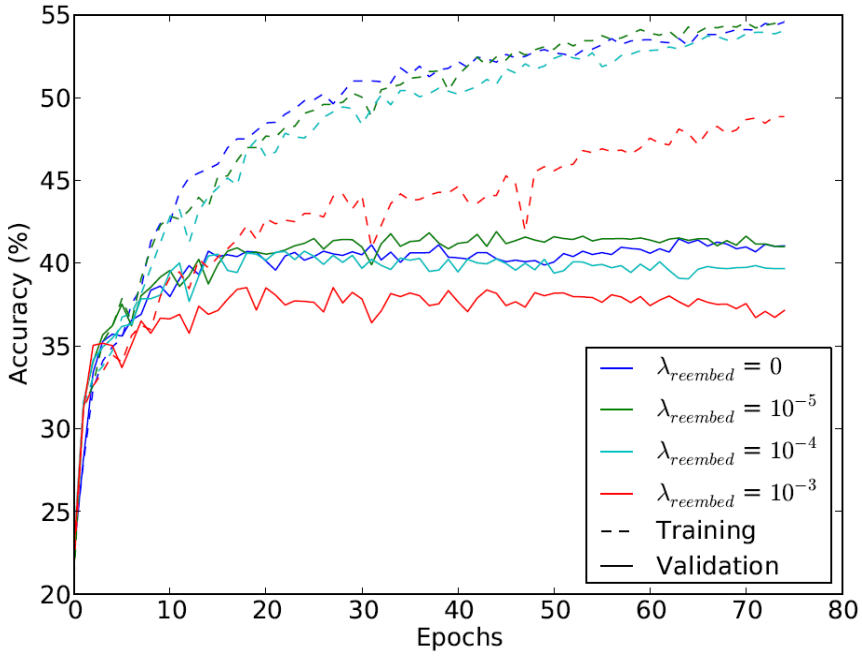
(d) Penalizing embeddings in Experiment II.

2. Penalizing l_2-norm of embeddings unexpectedly helps optimization (improves tranning accuracy).

Regularization Strategies Comparison



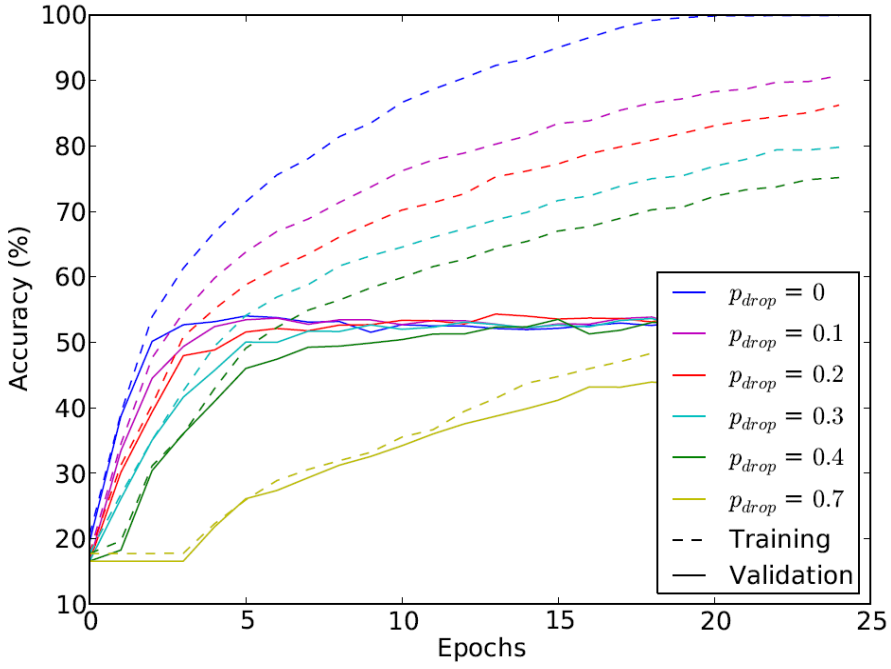
(e) Re-embedding words in Experiment I.



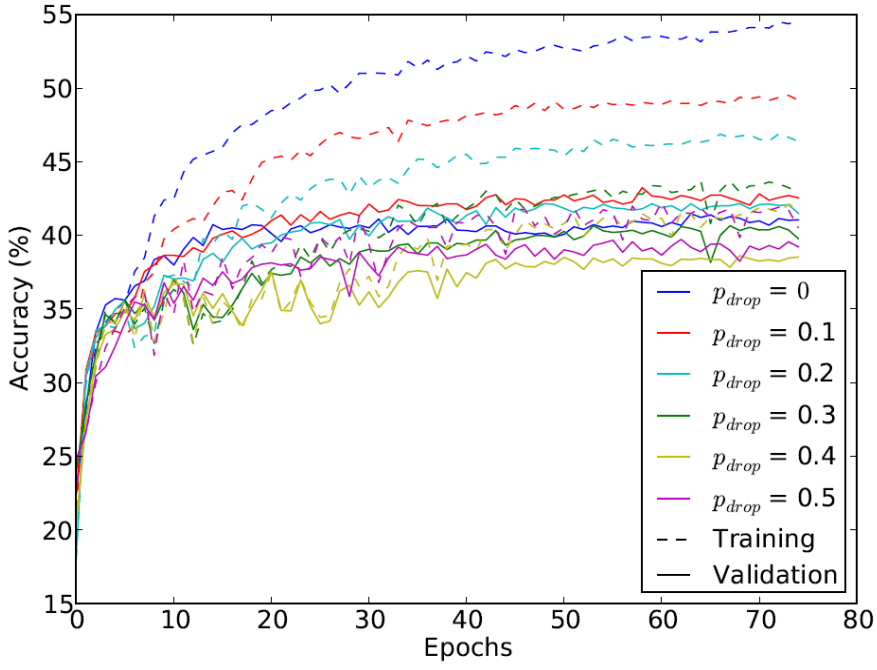
(f) Re-embedding words in Experiment II.

3. Re-embedding words, proposed in Labutov and Lipson (2013), does not improve generalization.

Regularization Strategies Comparison



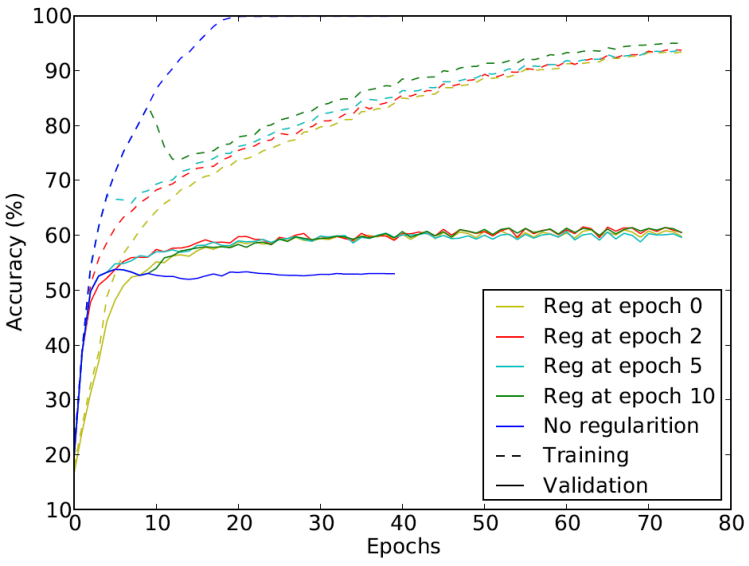
(g) Applying dropout in Experiment I. $p = 0.5, 0.6$ are omitted because they are similar to small values.



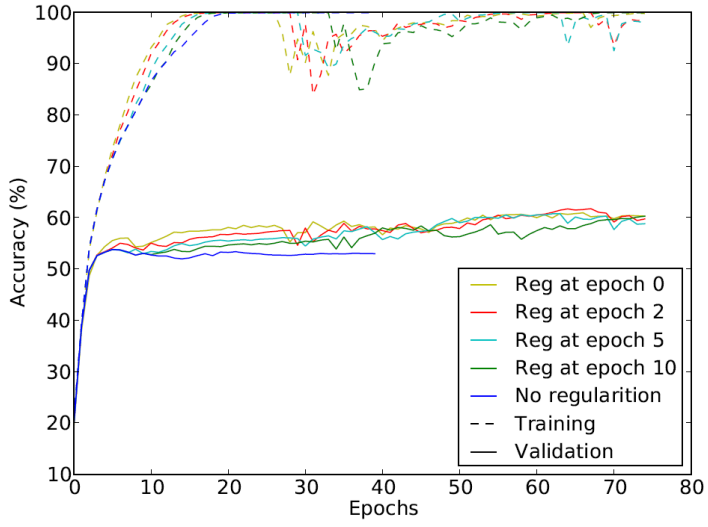
(h) Applying dropout in Experiment II.

- 4. Dropout does help generalization. Even though the training process becomes slower, the validation accuracy is still rising slightly after 200 epochs.

Regularization Strategies Comparison – Incrementally penalizing



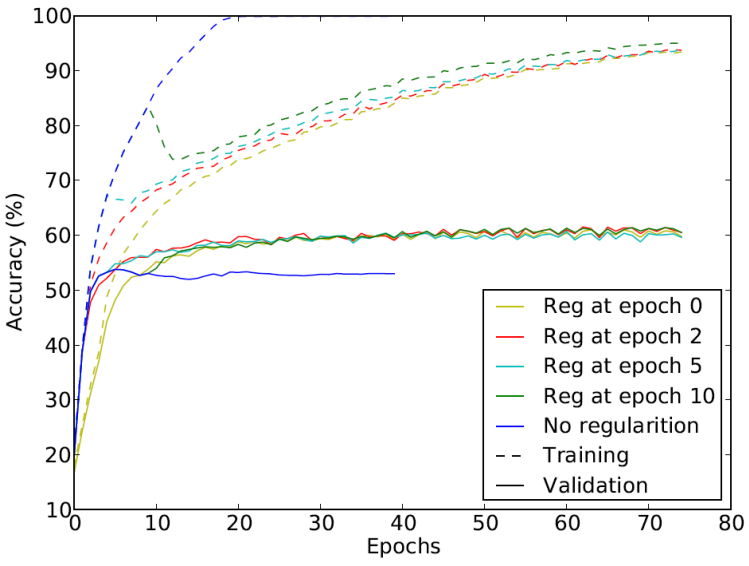
(a) Incrementally penalizing ℓ_2-norm of weights.



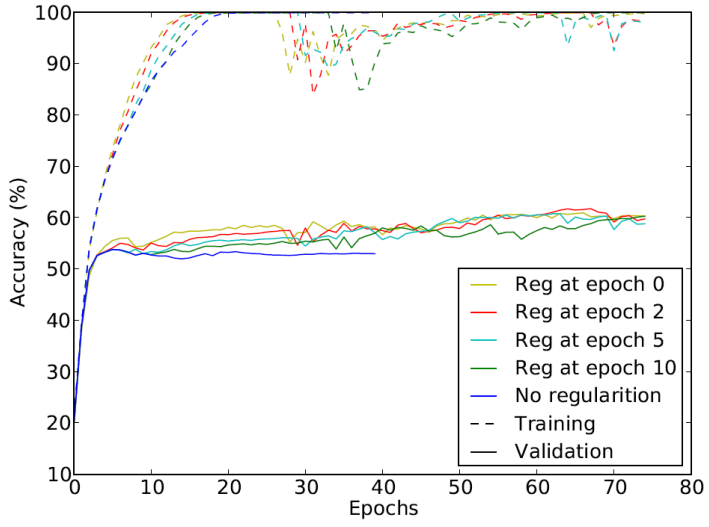
(b) Incrementally penalizing ℓ_2-norm of biases.

- all settings yield similar ultimate validation accuracies no matter when penalty is added.
- This shows l_2 regularization mainly serves as a "local" effect

Regularization Strategies Comparison – Incrementally penalizing



(a) Incrementally penalizing ℓ_2-norm of weights.



(b) Incrementally penalizing ℓ_2-norm of biases.

- (a) regularization helps generalization as soon as it is added: while the training accuracy first drops and then rises, the test accuracy goes up immediately.
- (b) that regularizing embeddings also helps optimization (training accuracy improves) right after the penalty is added.

Regularization Strategies Comparison – Combination of Regularizations

λ_{embed}	λ_W = 0	10^{-4}	$3 \cdot 10^{-4}$	10^{-3}
0	54.02	57.88	59.96	61.00
10^{-5}	54.94	57.82	60.68	62.05
$3 \cdot 10^{-5}$	55.68	61.02	**64.00**	**63.15**
10^{-4}	60.91	**64.00**	**63.07**	60.56
$3 \cdot 10^{-4}$	58.92	61.33	59.85	42.93
10^{-3}	54.77	56.43	54.05	16.50

- Table 1 shows that combining l_2-norm of weights and embeddings further improved accuracy by 3–4 in percentage than applying either single one of them.
- In a certain range of coefficients, weights and embeddings are compensatory: given one hyperparameter, we can tune the other to achieve the results among highest ones.

Regularization Strategies Comparison – Combination of Regularizations

	λ_W			λ_{embed}		
p	10^{-4}	$3\cdot10^{-4}$	10^{-3}	10^{-5}	$3\cdot10^{-5}$	10^{-4}
0	57.88	**59.96**	**61.00**	54.94	55.68	**60.91**
1/6	58.36	59.36	43.42	58.49	59.59	**60.00**
2/6	58.22	**60.00**	16.60	59.34	**60.08**	59.61
3/6	58.63	59.73	16.60	59.59	**59.98**	58.82
4/6	56.43	54.63	16.60	56.76	59.19	56.64
5/6	38.07	16.60	16.60	49.79	53.63	49.75

- Such compensation is also observed in penalizing l_2-norm versus dropout
- the peak performance is obtained by pure l_2-norm regularization
- applying dropout with small l_2 penalty also achieves similar accuracy.
- The dropout rate is not very sensitive, provided it is small (less than 0.5, say).

Thanks.