Introduction to Deep Learning

2. Machine Learning Basics

Haibin Lin and Leonard Lausen

gluon-nlp.mxnet.io

8:30-9:00	Continental Breakfast
9:00-9:45	Introduction and Setup
9:45-10:30	Neural Networks 101
10:30-10:45	Break
10:45-11:15	Machine Learning Basics
11:15-11:45	Context-free Representations for Language
11:45-12:15	Convolutional Neural Networks
12:15-13:15	Lunch Break
13:15-14:00	Recurrent Neural Networks
14:00-14:45	Attention Mechanism and Transformer
14:45-15:00	Coffee Break
15:00-16:15	Contextual Representations for Language
16:15-17:00	Language Generation

Model Evaluation

Training Error and Generalization Error

- Training error: model error on the training data
- Generalization error: model error on new data
- Example: practice a future exam with past exams
    - Doing well on past exams (training error) doesn't guarantee a good score on the future exam (generalization error)
    - Student A gets 0 error on past exams by rote learning
    - Student B understands the reasons for given answers

Validation Dataset and Test Dataset

- Validation dataset: a dataset used to evaluate the model
  - E.g. Take out 50% of the training data
  - Should not be mixed with the training data (#1 mistake)
- Test dataset: a dataset can be used once, e.g.
  - A future exam
  - The house sale price I bided
  - Dataset used in private leaderboard in Kaggle

K-fold Cross Validation

- Useful when not sufficient data
- Algorithm:
  - Partition the training data into K parts
  - For $i = 1, \dots, K$
    - Use the i-th part as the validation set, the rest for training
  - Report the averaged the K validation errors
- Popular choices: $K = 5$ or 10

Underfitting
Overfitting

Image credit: hackernoon.com

Underfitting and Overfitting

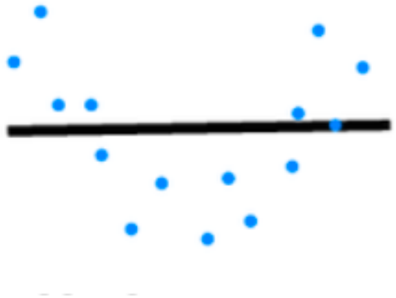
Data complexity

Model capacity

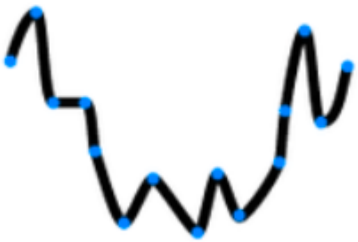
	Simple	Complex
Low	Normal	Underfitting
High	Overfitting	Normal

Model Capacity

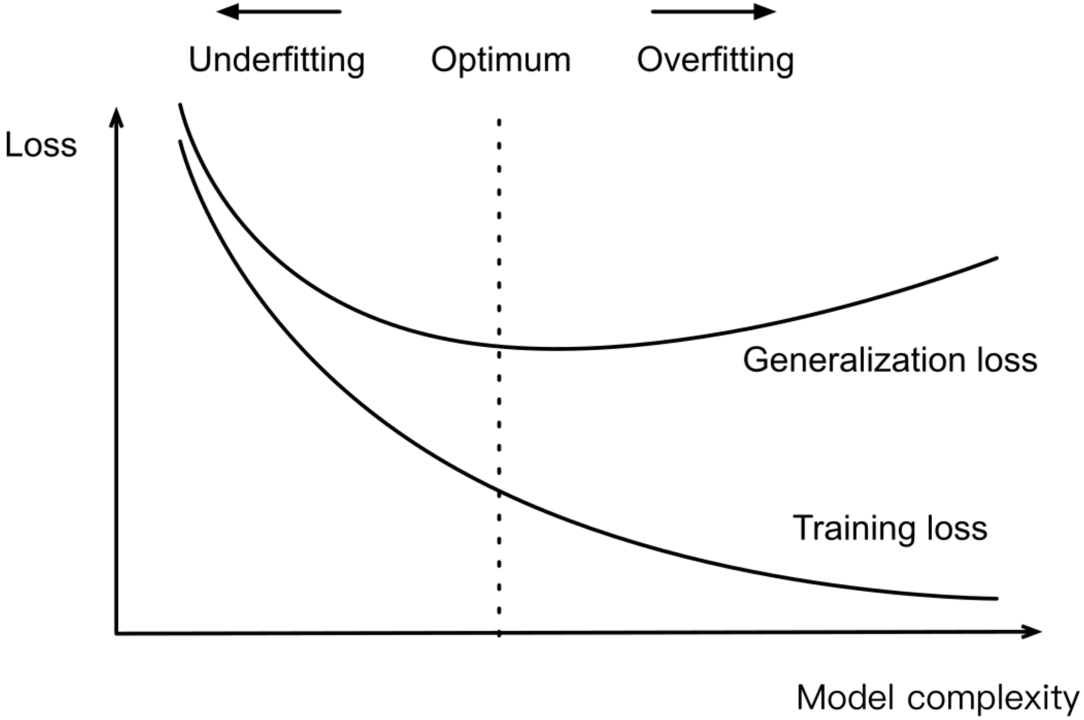
- The ability to fit variety of functions
- Low capacity models struggles to fit training set
    - Underfitting



- High capacity models can memorize the training set
    - Overfitting



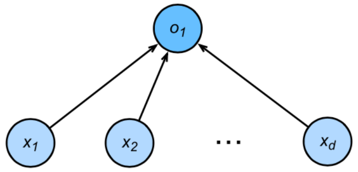
Influence of Model Complexity



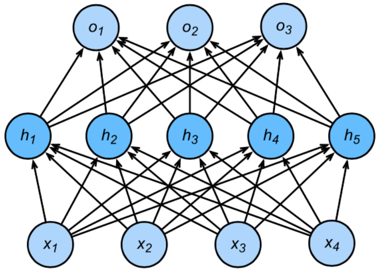
Estimate Model Capacity

- It's hard to compare complexity between different algorithms
  - e.g. tree vs neural network
- Given an algorithm family, two main factors matter:
  - The number of parameters
  - The values taken by each parameter

$$d + 1$$



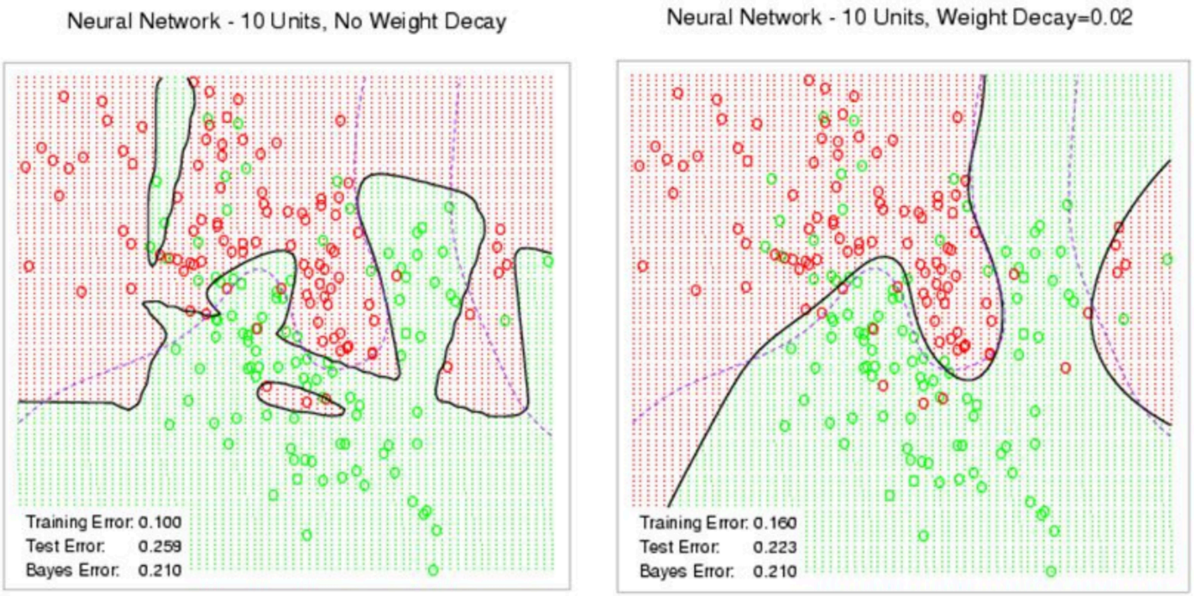
$$(d+1)m + (m+1)k$$



Data Complexity

- Multiple factors matters
    - # of examples
    - # of elements in each example
    - time/space structure
    - diversity

Weight Decay

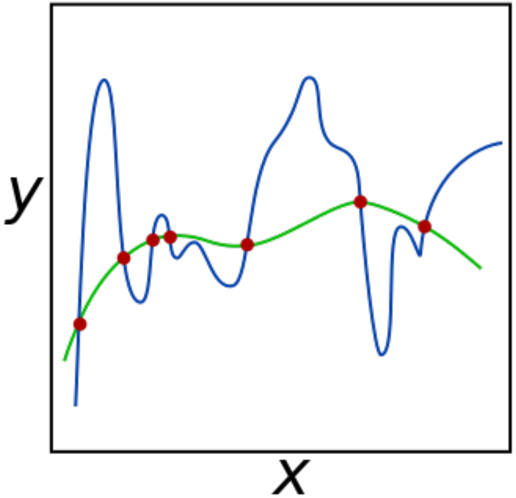


Squared Norm Regularization as Hard Constraint

- Reduce model complexity by limiting value range

$$\min \ \ell(\mathbf{w}, b) \quad \text{subject to} \ \ \|\mathbf{w}\|^2 \leq \theta$$

  - Often do not regularize bias *b*
    - Doing or not doing has little difference in practice
  - A small θ means more regularization



Squared Norm Regularization as Soft Constraint

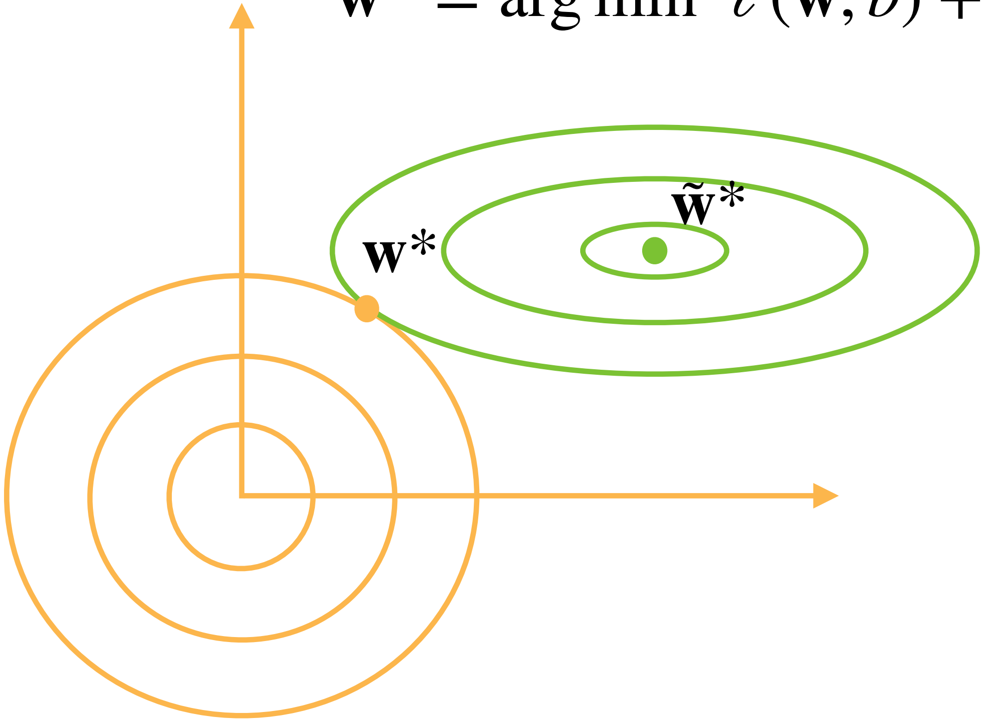
- For each θ , we can find λ to rewrite the hard constraint version as

$$\min \ \ell(\mathbf{w}, b) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

  - Hyper-parameter λ controls regularization importance
  - $\lambda = 0$: no effect
  - $\lambda \to \infty, \mathbf{w}^* \to \mathbf{0}$

Illustrate the Effect on Optimal Solutions

$$\mathbf{w}^* = \arg\min \ \ell(\mathbf{w}, b) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$



$$\tilde{\mathbf{w}}^* = \arg\min \ \ell(\mathbf{w}, b)$$

Update Rule

- Compute the gradient

$$\frac{\partial}{\partial \mathbf{w}}\left(\ell(\mathbf{w}, b) + \frac{\lambda}{2}\|\mathbf{w}\|^2\right) = \frac{\partial \ell(\mathbf{w}, b)}{\partial \mathbf{w}} + \lambda \mathbf{w}$$

- Update weight at time *t*

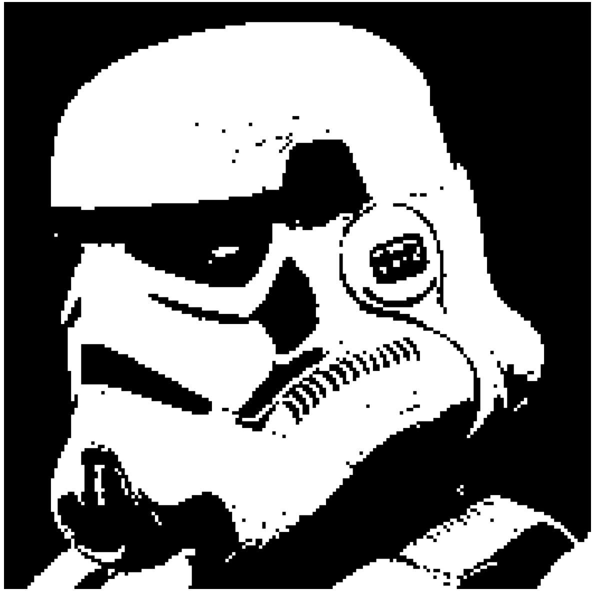
$$\mathbf{w}_{t+1} = (1 - \eta\lambda)\mathbf{w}_t - \eta \frac{\partial \ell(\mathbf{w}_t, b_t)}{\partial \mathbf{w}_t}$$

  - Often $\eta\lambda < 1$, so also called weight decay in deep learning

Dropout

Motivation

- A good model should be robust under modest changes in the input



Add Noise without Bias

- Add noise into ***x*** to get ***x'***, we hope

$$\mathbf{E}[\mathbf{x}'] = \mathbf{x}$$

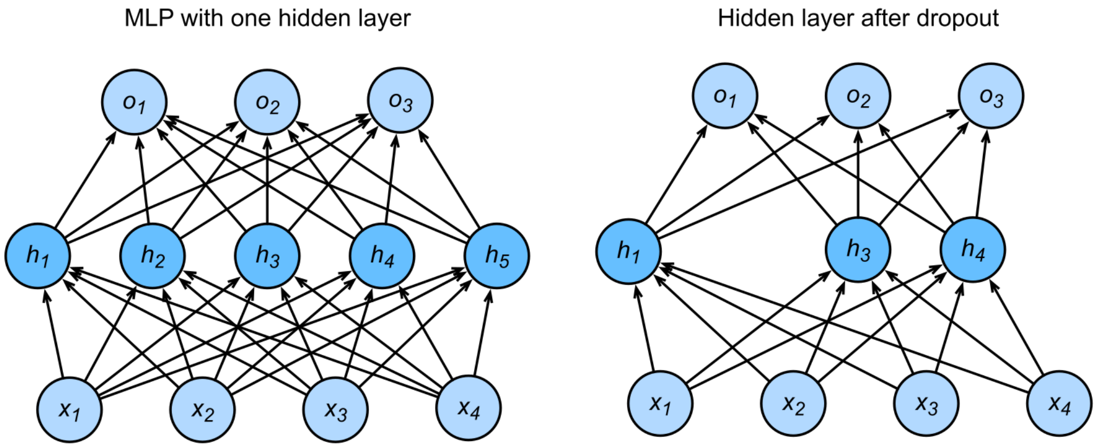
- Dropout perturbs each element by

$$x_i' = \begin{cases} 0 & \text{with probablity } p \\ \frac{x_i}{1-p} & \text{otherise} \end{cases}$$

Apply Dropout

- Often apply dropout on the output of hidden fully-connected layers

$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{h}' = \text{dropout}(\mathbf{h})$$
$$\mathbf{o} = \mathbf{W}_2\mathbf{h}' + \mathbf{b}_2$$
$$\mathbf{y} = \text{softmax}(o)$$



Dropout in Inference

- Regularization is only used in training
- The dropout layer for inference is

$$\mathbf{h}' = \text{dropout}(\mathbf{h})$$

  - Guarantee deterministic results