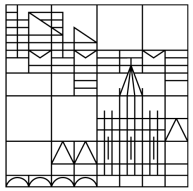
Universität Konstanz



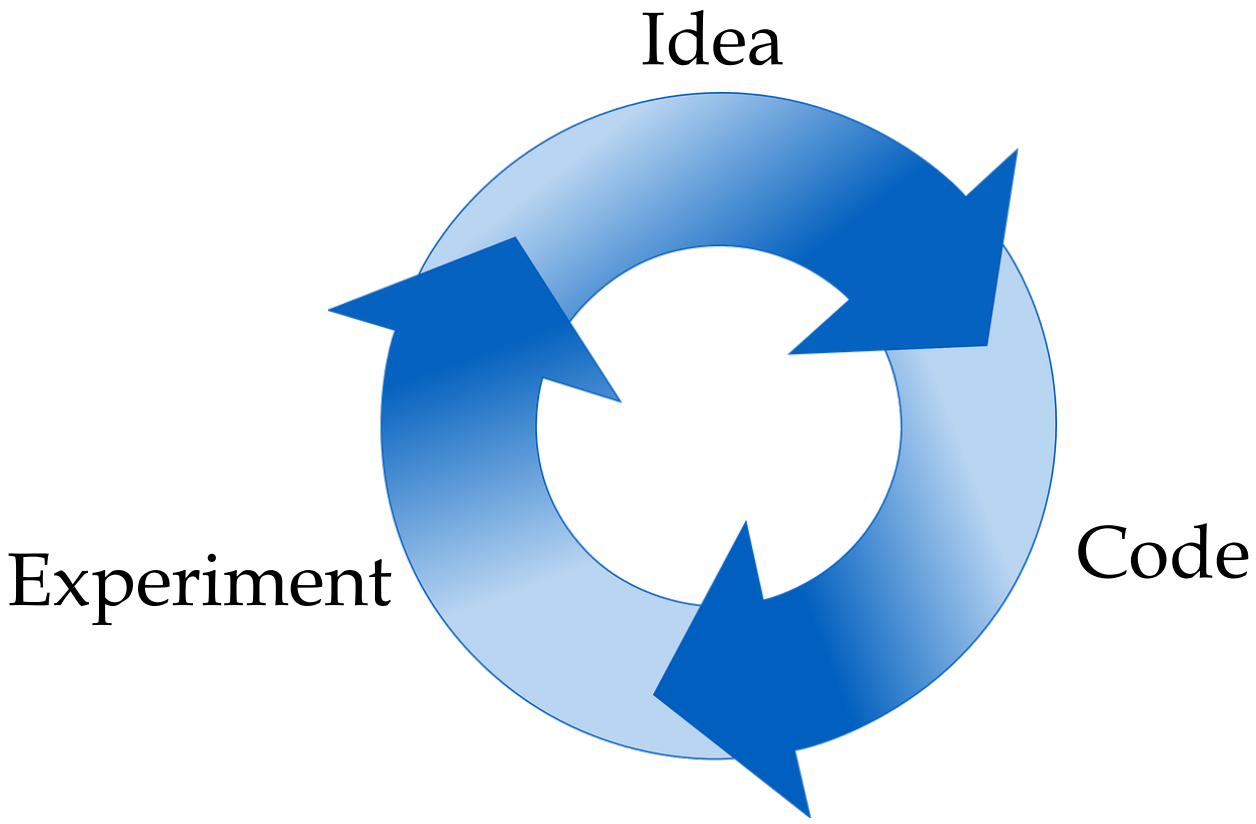
Lecture 11 Practical Methodology

Dr. Hanhe Lin
Dept. of Computer and Information Science
University of Konstanz

Building a ML project

- Step 1: Data collection and annotation
- Step 2: Train and test ML models
- Step 3: Data analysis on your results
- Step 4: Writing technical report

Applying ML is a very empirical process



Getting started on a ML problem

Approach #1: Careful design

- Spend a long term designing exactly the right features, collecting the right dataset, and designing the right algorithmic architecture
- Implement it and hope it works
- Benefit: Nicer, perhaps more scalable algorithms. May come up with new, elegant, learning algorithms
- Contribute to basic research in machine learning

Approach #2: Build-and-fix

- Implement something quick-and-dirty
- Run error analyses and diagnostics to see what's wrong with it, and fix its errors
- Benefit: Will often get your application problem working more quickly
- Faster time to market

Practical machine learning system design process

- Determine your goal - what error metric to use, and your target value for this error metric
- Establish a working end-to-end pipeline as soon as possible, including the estimation of the appropriate performance metrics
- Instrument the system well to determine bottlenecks in performance
- Repeatedly make incremental changes such as gathering new data, adjusting hyper-parameters, or changing algorithms, based on specific findings from your instrumentation

What is your ML project?

- Is your project
  - Supervised or unsupervised?

- If it is supervised, what kind is it?
  - Regression or classification?

Data collection and annotation

- Try to use some available datasets, e.g., Kaggle, ImageNet, YouTube-8M, . . .
- If not, collect data for your specific project, annotate data by yourself if the dataset is small
- If the dataset you collected is very large, it is very time-consuming to annotate it by yourself. Instead, you can use crowdsourcing platforms, e.g., Amazon Mechanical Turk, Appen (Figure eight)

Before training…

- Feature normalization
  - Mean subtraction
  - Normalization
  - PCA
  - …
- Data augmentation

Data augmentation

- Create fake data and add it to the training set if your dataset is small and difficult to get more data
- It is easiest for classification
- It has been a particularly effective technique for object recognition problem
- Not to apply transformation that would change the correct class, e.g., b and d
- Taking the effect of dataset augmentation when comparing machine learning algorithms

Hyper-parameter tuning

- To train a machine learning model, one must tune hyper-parameters. Take neural network as an example, number of layers, learning rate of SGD, number of units in each hidden layers, regularization parameter, ...

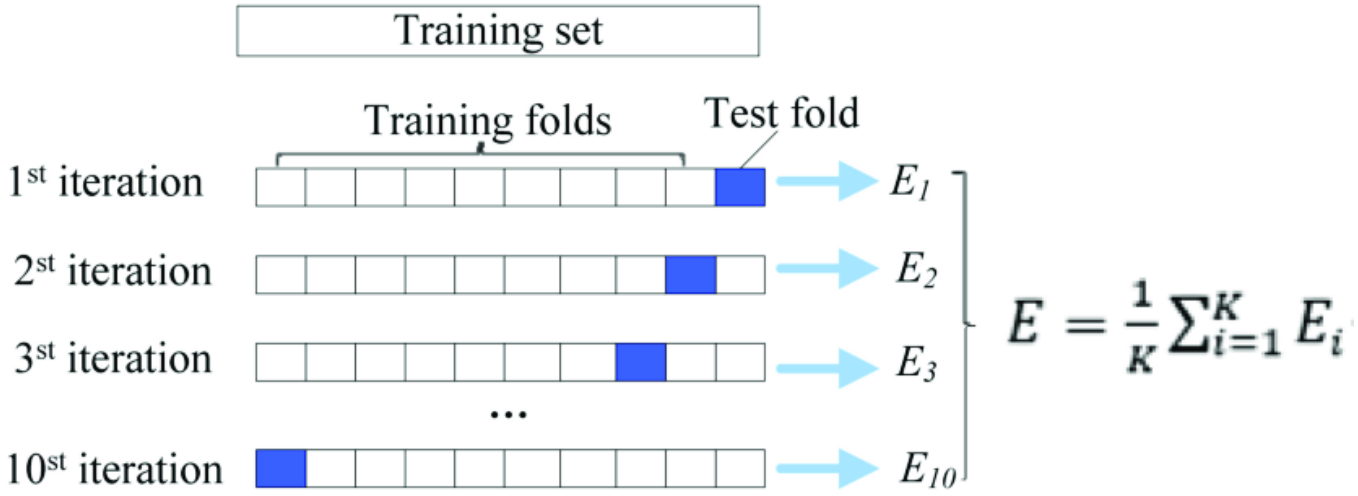
- Two options to find optimal parameters:
  - A separate validation set when you have a large dataset
  - K-fold cross validation when you have a small dataset

A separate validation set

- Split dataset into three non-overlapping sets, i.e., training set, validation set, and test set.
- For example, in MNIST dataset, we can have 50,000 training data, 10,000 validation data, and 10,000 test data
- Train your model with training set, and measure the performance on validation set with different hyper-parameters, choose the optimal ones. Namely, the ones that have the best performance on validation set
- Measure the performance of your model on test set with the optimal parameter, only ONCE
- More bias but less computational time

K-fold Cross Validation (CV)

- Split training set into k equal sized subsets
- A single subset is retained as the validation data for testing the model, and the remaining k - 1 subsets are used as training data
- Repeat training process k times (the folds), with each of the k subsets used exactly once as the validation data.
- The k results from the folds can then be averaged to produce a single estimation
- Less bias but more computational time



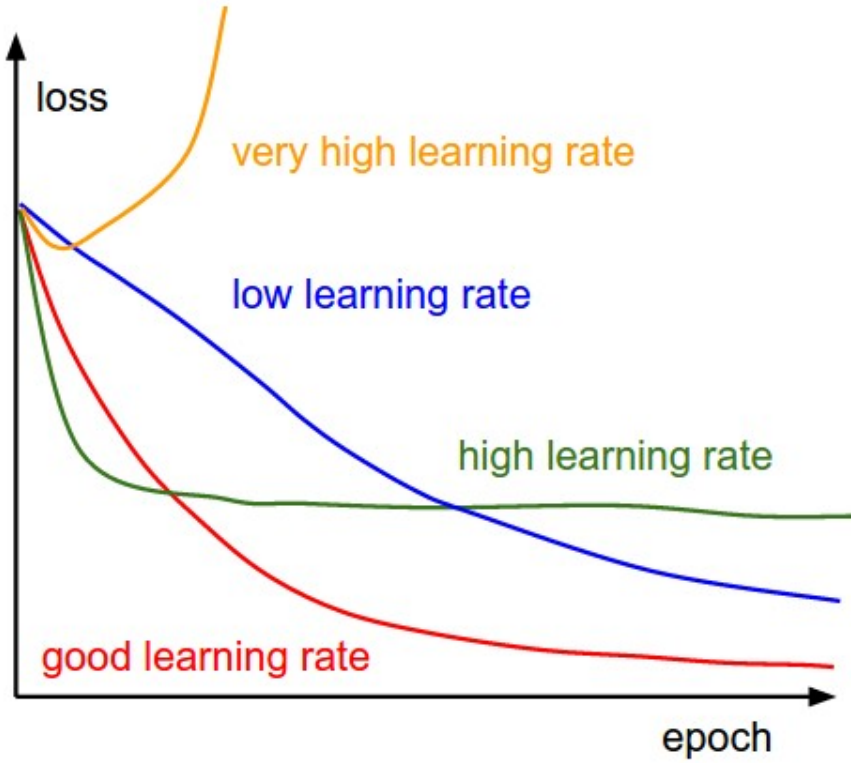
Manual hyper-parameter tuning

- To set hyper-parameters manually, one must understand the relationship between hyper-parameters, training error, validation error and computational resources
- If time to tune only one hyper-parameter, tune the learning rate
- Tuning the hyper-parameters other than the learning rate requires monitoring both training and validation error to diagnose

Learning rate

- Learning rate is a crucial parameter for SGD
  - Too large: overshoots local minimum, loss increases
  - Too small: makes very slow progress, can get stuck
  - Just right: makes steady progress towards local minimum

- Batch gradient descent can uses a fixed learning rate as the true gradient becomes small and then 0 when it reach a minimum, however, this is not suitable for mini-batch SGD



Grid search

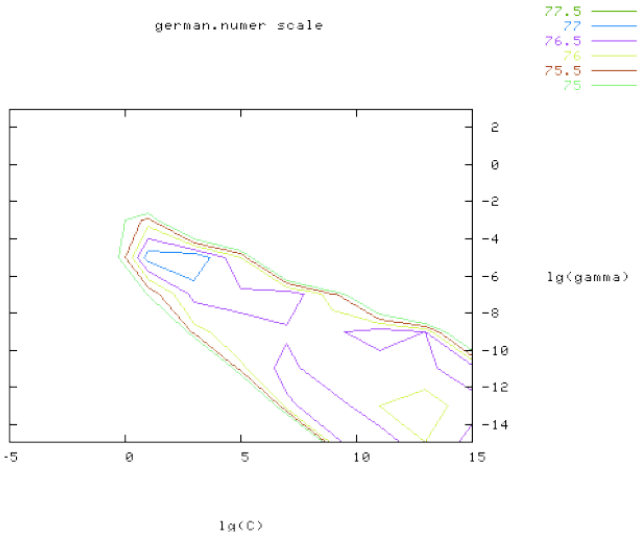
- When there are three or fewer hyper-parameters, the common practice is to perform grid search
- For each hyper-parameter, select a small finite set of values to explore
- Typically, a grid search involves picking values approximately on a logarithmic scale
- Coarse-to-fine strategy
- Cons: computational cost grows exponentially with the number of hyper-parameters

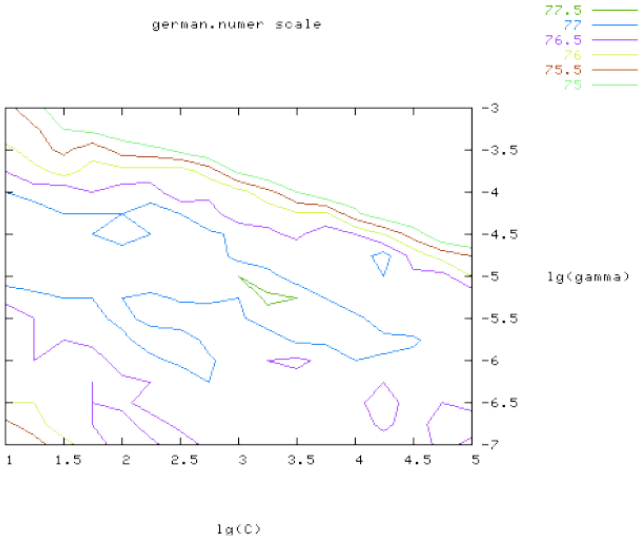
Grid search example: SVM

- Two hyper-parameters to tune in SVM with RBF kernel, C and γ

- Coarse grid: exponentially growing sequences of C and γ
- to identify good parameters: $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}, \gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$

- If we find the best (C, γ) is $(2^{3}, 2^{-5})$, then we conduct a fine grid search on its neighborhood





Random search

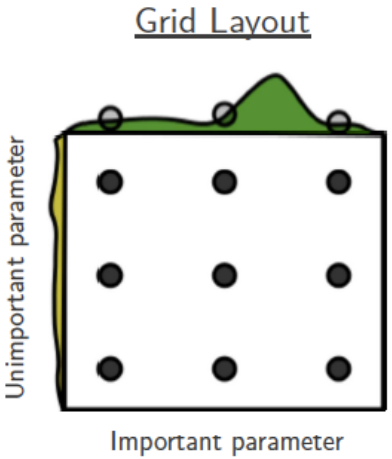
- Define a marginal distribution for each hyper-parameter

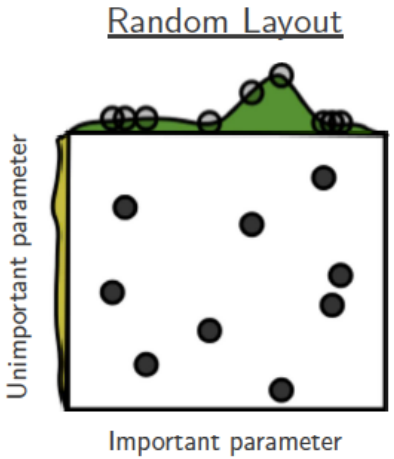
For example:

log_learning_rate ~ $u(-1, -5)$

learning_rate = $10^{\text{log\_learning\_rate}}$

where $u(-1, -5)$ indicates a sample of uniform distribution in the interval $(-1, -5)$





- We should not discretize or bin the values of the hyper-parameters, so that we can explore a larger set of values and avoid additional computational cost

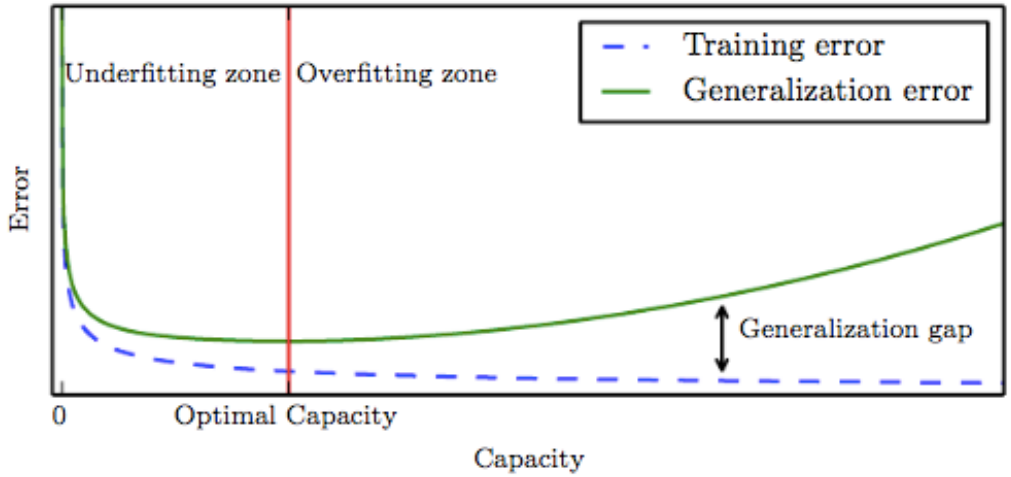
ML model selection

m = # training examples n = # features

- If n is small, m is small and intermediate (up to 10,000), traditional ML methods are more suitable
- If n is small, m is large, neural network are more suitable
- If n is large (e.g., image, audio), m is intermediate and large, convolutional neural networks are more suitable
- If the data is time-dependency, models such as hidden Markov chain, long short-term memory should be considered
- Use MATLAB toolbox!

Capacity, overfitting and underfitting

- Generalization: the ability to perform well on previously unobserved inputs
- We want a machine learning algorithm:
  - the training error small
  - the gap between training and test error small
- Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set
- Overfitting occurs when the gap between the training error and test error is too large
- A model's capacity is its ability to fit a wide variety of functions



Debugging a learning model

Suppose you have implemented regularized linear regression to predict housing prices:

$$L(w,b) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n} w_j^2\right]$$

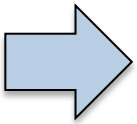
However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

- Get more training examples
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features
- Try decreasing lambda
- Try increasing lambda

Example: linear regression

Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

Randomly Shuffled



Size	Price
2104	400
2400	369
1416	232
3000	540
1534	315
1427	199
1380	212

Training set (70%)

Size	Price
1600	330
2400	300
1494	243

Test set (30%)

Example – linear regression

Size	Price
2104	400
2400	369
1416	232
3000	540
1534	315
1427	199
1380	212

Training set (70%)

Minimize the following loss function using The training set

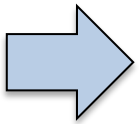
$$L(w,b) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} w_j^2\right]$$

Optimal w, b

Example – linear regression

Size	Price
1600	330
2400	300
1494	243

Test set (30%)



$$\frac{1}{2m}\sum_{i=1}^{m}\left(h_{w,b}\left(x^{(i)}\right)-y^{(i)}\right)^2=\frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)}+b-y^{(i)}\right)^2$$

Mean squared error

Select optimal degree of polynomial

- Take linear regression as an example, you may need to choose the degree of polynomial (d), i.e.,

$$h_{w,b}(x) = \sum_{i=1}^{d} w_i x^i + b$$

- You tried d from 1 to 10, and you find $d = 3$ have the lowest mean square error in test data. So you claim $d = 3$ is the optimal parameter of your model. Anything wrong?

Select optimal degree of polynomial

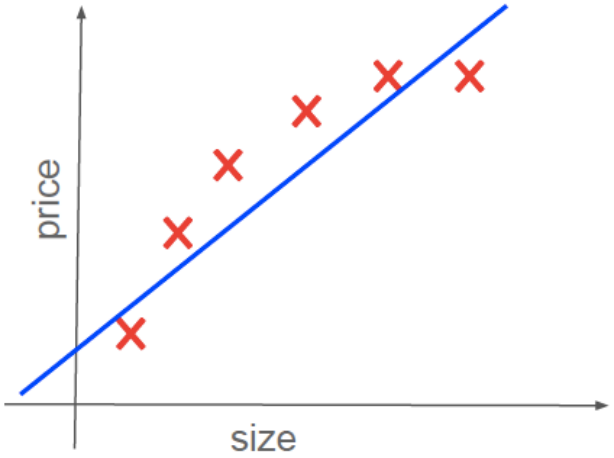
- Take linear regression as an example, you may need to choose the degree of polynomial (d), i.e.,

$$h_{w,b}(x) = \sum_{i=1}^{d} w_i x^i + b$$

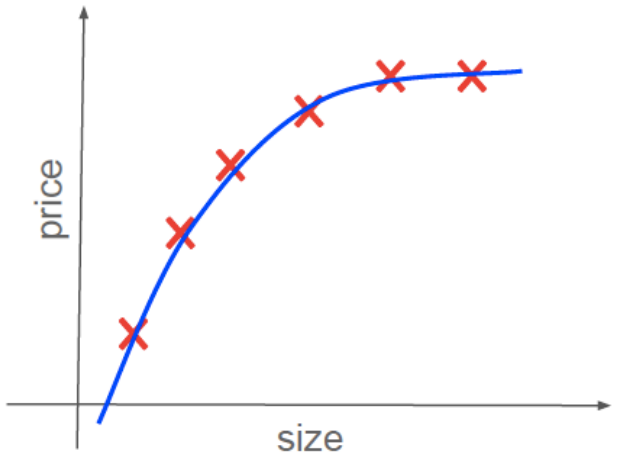
- You tried d from 1 to 10, and you find $d = 3$ have the lowest mean square error in test data. So you claim $d = 3$ is the optimal parameter of your model. Anything wrong?

If you apply your model to other data, the performance may decrease as the parameter is fit to the test data. Namely, you don't know how well your model is generalized to unseen examples.

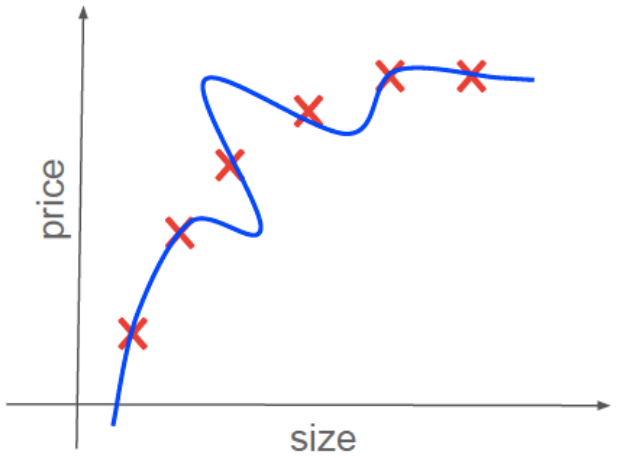
Bias vs. variance on regression



Underfitting, high bias



Just right



Overfitting, high variance

Bias vs. variance on degree of polynomial

- Using the mean squared error which is defined before, we have training error and validation error:

$$err_{train} = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(h_{w,b}\left(x_{train}^{(i)}\right) - y^{(i)} \right)^2$$

$$err_{valid} = \frac{1}{2m_{valid}} \sum_{i=1}^{m_{valid}} \left(h_{w,b}\left(x_{valid}^{(i)}\right) - y^{(i)} \right)^2$$

- If we change the degree of polynomial d, what will the training error and validation error look like?

Diagnosing bias vs. variance on degree of polynomial

- Suppose your machine learning model is performing less well than you were hoping. Is it a bias problem or a variance problem?

- High bias (underfitting): both training error and validation error are high

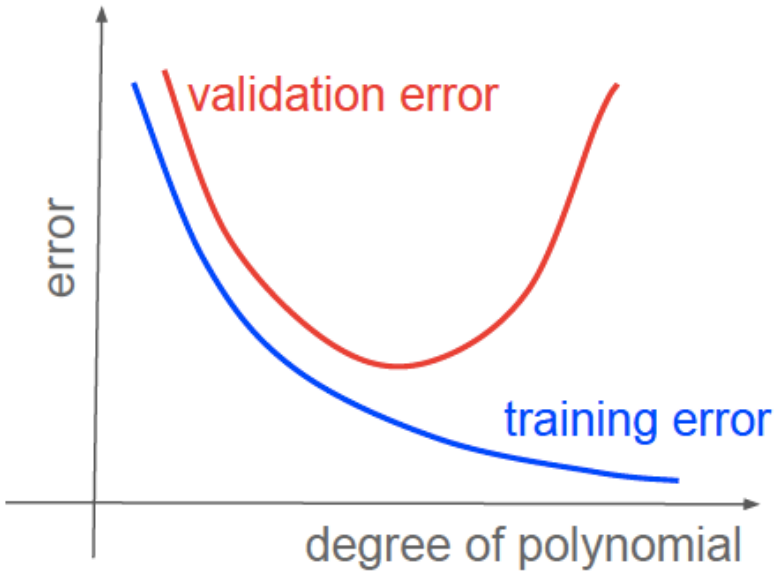
- High variance (overfitting): training error << test error

Diagnosing bias vs. variance on degree of polynomial

- Suppose your machine learning model is performing less well than you were hoping. Is it a bias problem or a variance problem?

- High bias (underfitting): both training error and validation error are high

- High variance (overfitting): training error << test error

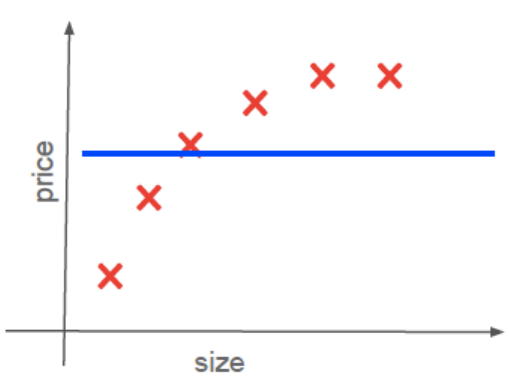


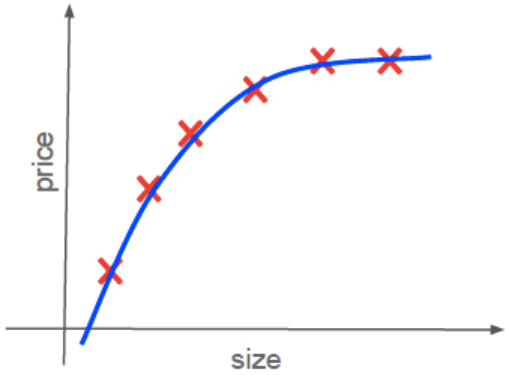
Bias vs. variance on regularization

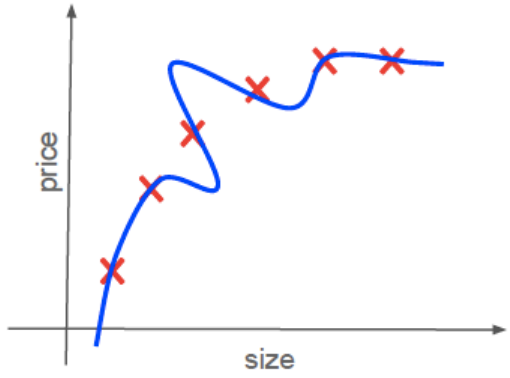
- Let's fix the degree of polynomial $d = 4$, what will the hypothesis look like with different values of lambda λ?

$$h_{w,b}(x) = \sum_{i=1}^{d} w_i x^i + b = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$$

$$L(w,b) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} w_j^2\right]$$





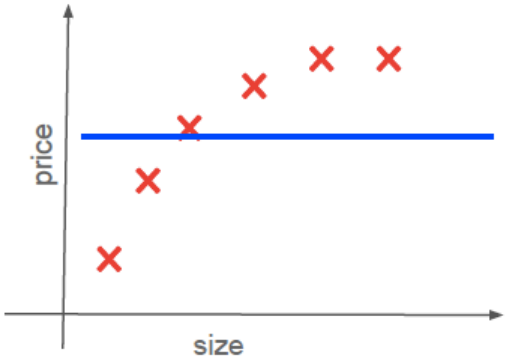


Bias vs. variance on regularization

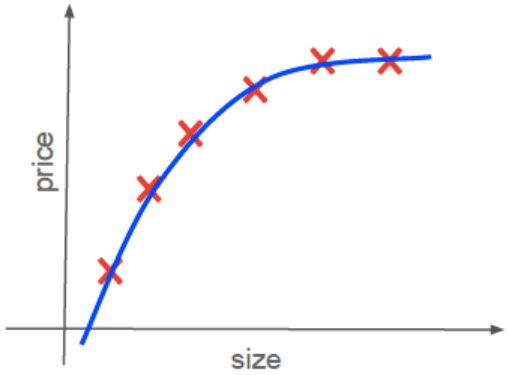
- Let's fix the degree of polynomial $d = 4$, what will the hypothesis look like with different values of lambda λ?

$$h_{w,b}(x) = \sum_{i=1}^{d} w_i x^i + b = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$$

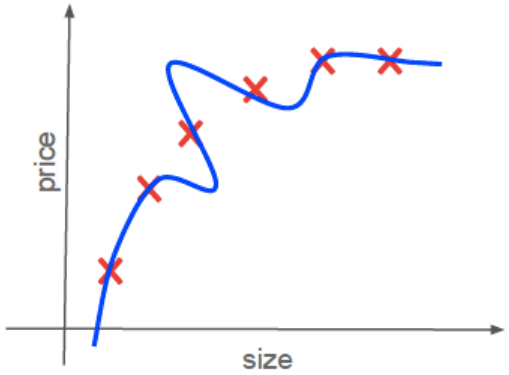
$$L(w,b) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} w_j^2\right]$$



Large λ



Intermediate λ



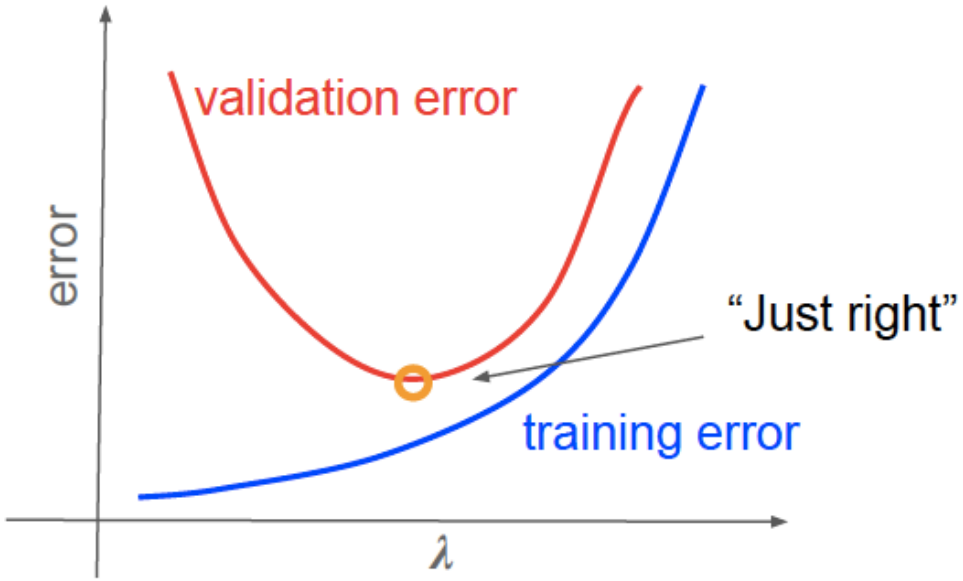
Small λ

Diagnosing bias vs. variance on regularization

- If we change the value of regularization hyper-parameter λ, what will the training error and validation error look like?

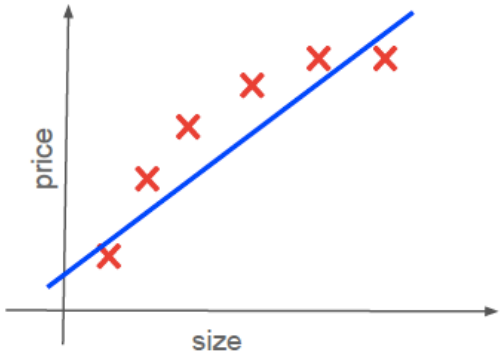
Diagnosing bias vs. variance on regularization

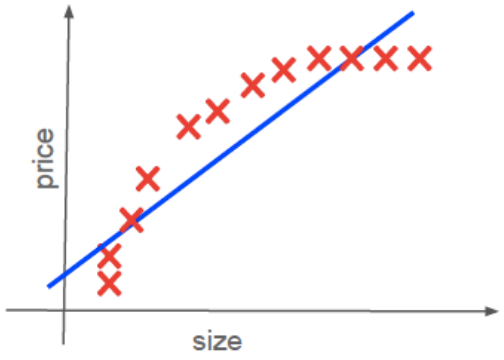
- If we change the value of regularization hyper-parameter λ, what will the training error and validation error look like?

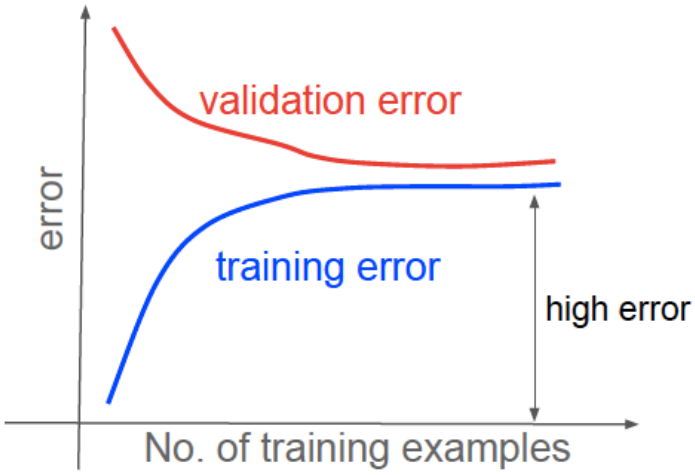


Bias vs. variance on size of training examples

- If a learning algorithm is suffering from high bias, what will the training error and validation error look like when increasing training examples?

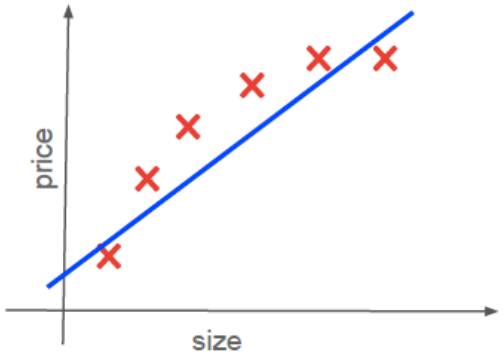


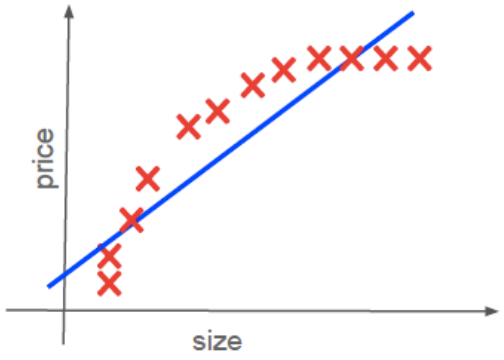


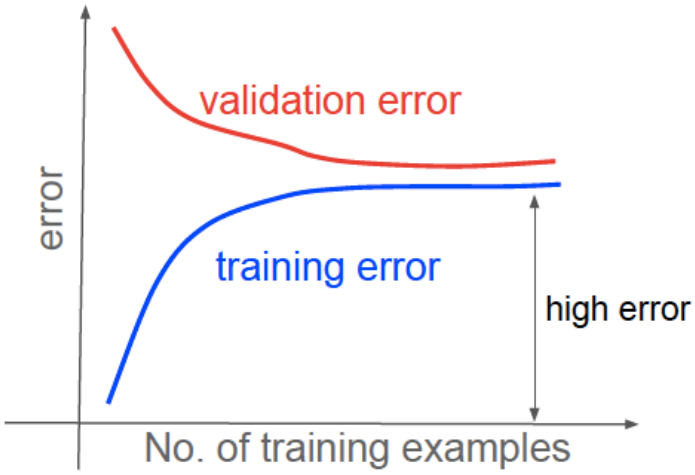


Bias vs. variance on size of training examples

- If a learning algorithm is suffering from high bias, what will the training error and validation error look like when increasing training examples?



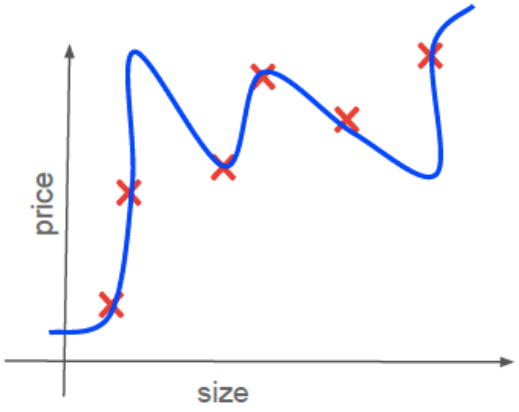


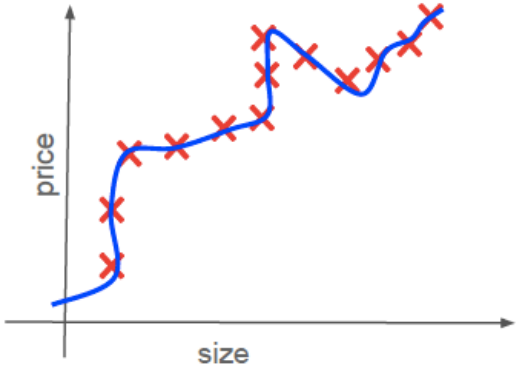


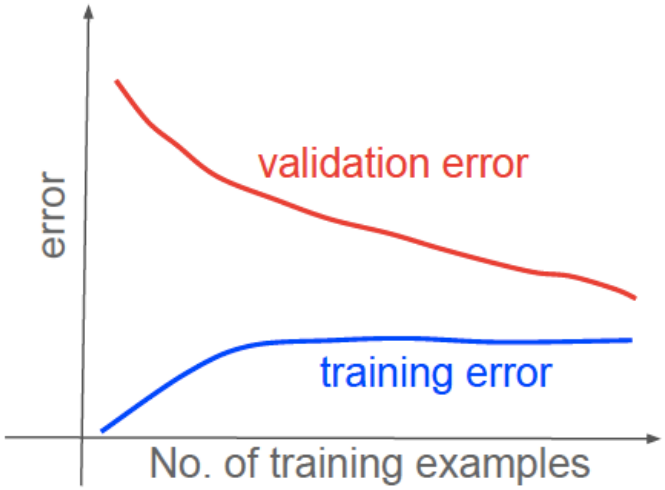
Increasing number of training examples will not help much if the model is high bias!

Bias vs. variance on size of training examples

- If a learning algorithm is suffering from high variance, what will the training error and validation error look like when increasing number of training examples?

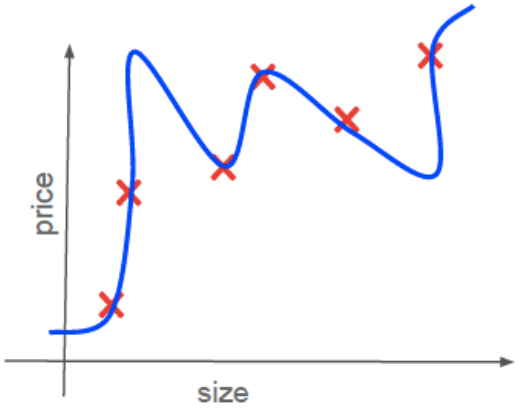


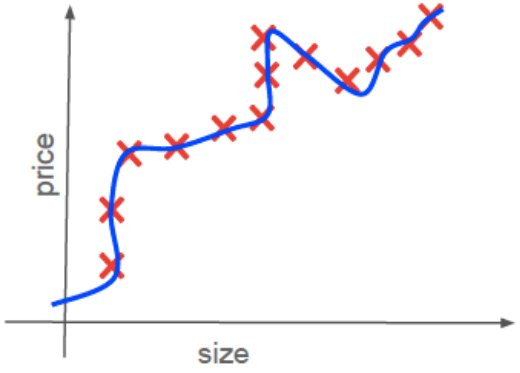


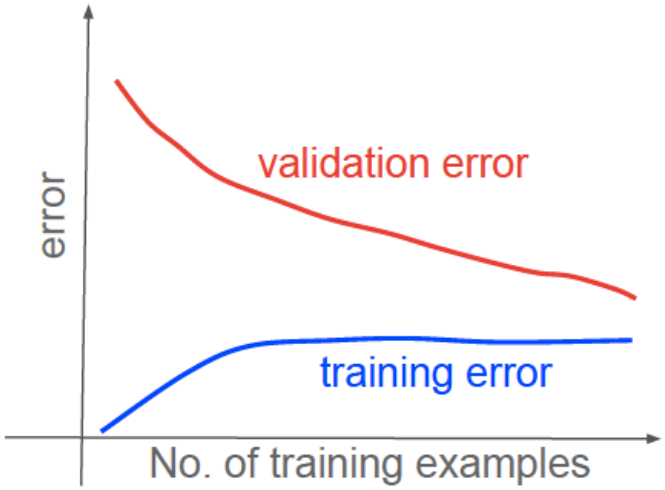


Bias vs. variance on size of training examples

- If a learning algorithm is suffering from high variance, what will the training error and validation error look like when increasing number of training examples?







Increasing number of training examples is likely to help if the model is high variance

Debugging a learning model

Suppose you have implemented regularized linear regression to predict housing prices:

$$L(w,b) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n} w_j^2\right]$$

However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

- Get more training examples
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features
- Try decreasing lambda
- Try increasing lambda

Debugging a learning model

Suppose you have implemented regularized linear regression to predict housing prices:

$$L(w,b) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n} w_j^2\right]$$

However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

- Get more training examples ➔ fixes high variance
- Try smaller sets of features ➔ fixes high variance
- Try getting additional features ➔ fixes high bias
- Try adding polynomial features ➔ fixes high bias
- Try decreasing lambda ➔ fixes high bias
- Try increasing lambda ➔ fixes high variance

Bayes error

- Keep in mind that for most applications, it is impossible to achieve absolute zero error
- The Bayes error defines the minimum error rate that you can hope to achieve
- Why?
  - Your input features may not contain complete information about the output variable
  - The system might be intrinsically stochastic
  - Having a finite amount of training data

Is your error metric fair

- Suppose you have trained a deep model to predict cancer. In your test set, only 0.5% of patients have cancer (skewed classes). You got 1% error on test set. Is your model a good classifier?
  - Positive example (1) - patient have cancer
  - Negative example (0) - patient no cancer

  def predictCancer(x)

  return y = 0

- You will achieve 0.5% error without doing anything!

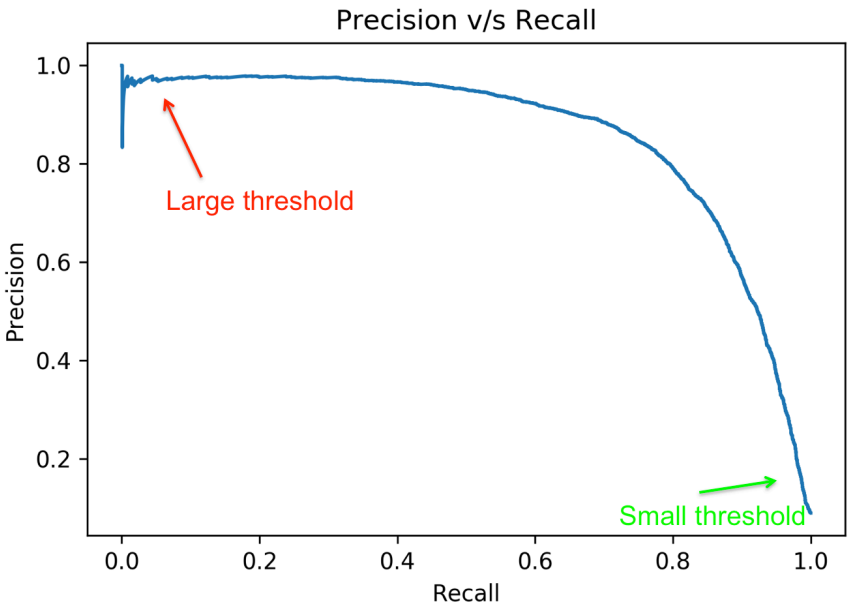
Precision / Recall

		Predicted condition	
	Total population	Positive	Negative
True condition	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

- Precision = TP/(TP+FP)
- Recall = TP/(TP+FN)
- Precision: Of all patients where we predicted have cancer, what fraction of patients actually have cancer
- Recall: Of all patients that actually have cancer, what fraction of patients did we correctly detected as having cancer

Tradeoff between precision and recall

- Suppose we want to predict cancer (y = 1) only if very confident
  - Higher precision, lower recall (large threshold)
- Suppose we want to avoid missing too much cases of cancer (avoid false negatives)
  - Higher recall, lower precision (small threshold)
- Generate the curve by tuning threshold



F1-measure

- Suppose you have the precision and recall of three learning algorithms, which one is better?
- F1-measure:

$$F_1 = 2 \cdot \frac{\text{precision} \times \text{recall}}{\text{precision+recall}}$$

- Algorithm 1 has the highest F1-measure
- Another option is to report the total area lying under the PR curve (AUC)

	Precision	Recall
Algorithm 1	0.6	0.3
Algorithm 2	0.2	0.9
Algorithm 3	0.9	0.1

Summary

The procedure of a machine learning project:

1. Collect data and divide it into training, validation, and test sets.
2. Choose the machine learning model you would like to use
3. Select the optimal parameters by means of training and validation sets
4. With the optimal parameters, predict results on test set
5. Measure and analyze your result, improve your model if possible
6. Write your project report