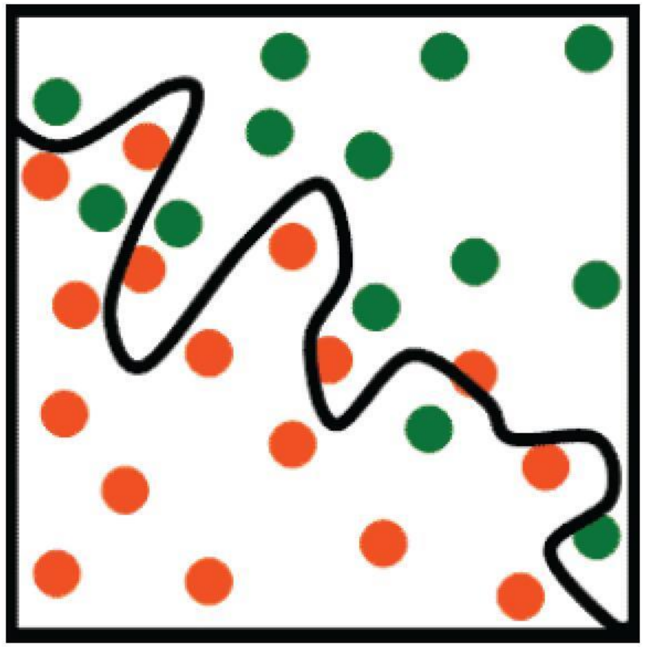
Supervised Learning (I)



Outline

I. What is Machine Learning

II. Supervised Learning

III. Computational Learning Theory

I What is Machine Learning

Machine Learning

A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E.*

---Tom M. Mitchell

Learn from experience/examples/samples

- There are many forms of machine learning. We focus on learning from examples here.
- Learning from experience/examples/samples:
  - Supervised learning (监督学习)
    - Classification, regression
  - Unsupervised learning (无监督学习)
    - Clustering
  - Semi-supervised learning (半监督学习)
  - Reinforcement learning (强化学习/增强学习)
  - …

Types of Machine Learning

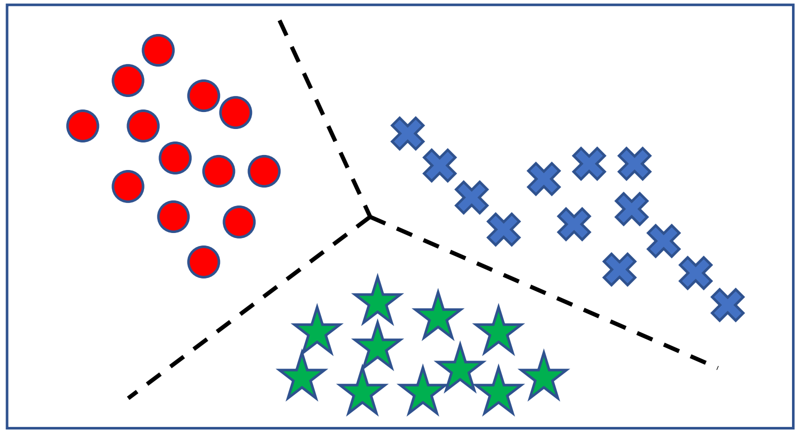
Based on the available teacher/supervision information

- Supervised learning: Training data include both inputs and outputs.
  - Given: training data of inputs $\mathcal{X}_l$ and corresponding outputs $\mathcal{Y}_l$.
  - Goal: predict a ‘correct’ output for a new input.
- Unsupervised learning: Training data do not include outputs.
  - Given: only unlabeled data of inputs $\mathcal{X}_u$.
  - Goal: learn some structure of $\mathcal{X}_u$ or relationship among $\mathcal{X}_u$’s.
- Semi-supervised learning: Some training data are with output labels and some without.
  - Given: A small portion of $(\mathcal{X}_l, \mathcal{Y}_l)$ and large portion of $\mathcal{X}_u$.
  - Goal: prediction (classification).

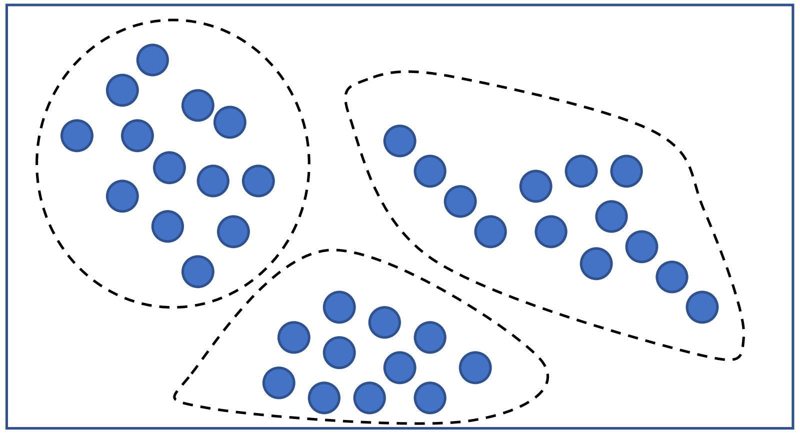
Types of Machine Learning

- Reinforcement learning:
  - Given: Training data do not include output labels, but do have a scalar feedback.
  - Goal: learn a sequence of actions that maximize some cumulative rewards.
- And so on ...

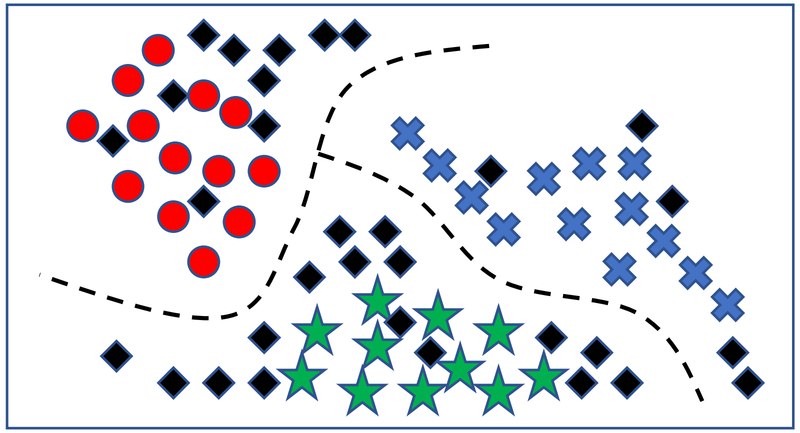
Types of Machine Learning: Illustration



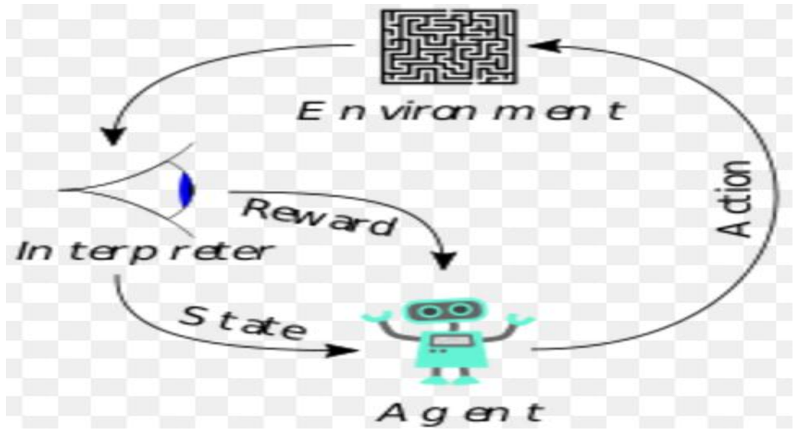
Supervised learning



Unsupervised learning



Semi-supervised learning



Reinforcement learning

II Supervised Learning

1. **What is Supervised Learning (formulation)**
2. Hypothesis Space $\mathcal{H}$ for Curve Fitting

Supervised Learning

- Using past experiences to improve future performance on some task.
- Experience: the training examples or training data.
- What does it mean to improve performance? Learning is guided by an objective, e.g. a loss function to be minimized.

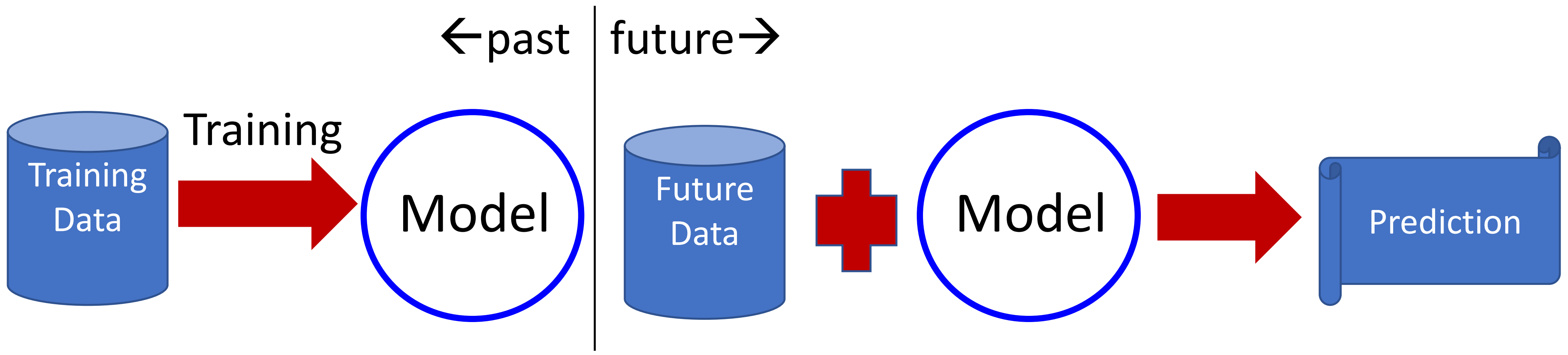


Figure generated by Liyan Song.

Training

- Data collection: Start with training data $\mathcal{D}$ from which experience is learned.
- Data representation: Encode $\mathcal{D}$ to be the input to the learning system.
- Modeling: Choose a hypothesis space $\mathcal{H}$ --- a set of possible models for $\mathcal{D}$.
- Learning: Find the best hypothesis $h \in \mathcal{H}$ according to some objective.
- Model selection: Select the best model according to some criteria.

➢ Two important factors: modeling + optimization (learning process).

Prediction/Inference

- The goal of machine learning is to predict the output of the future unseen data based on the trained model.

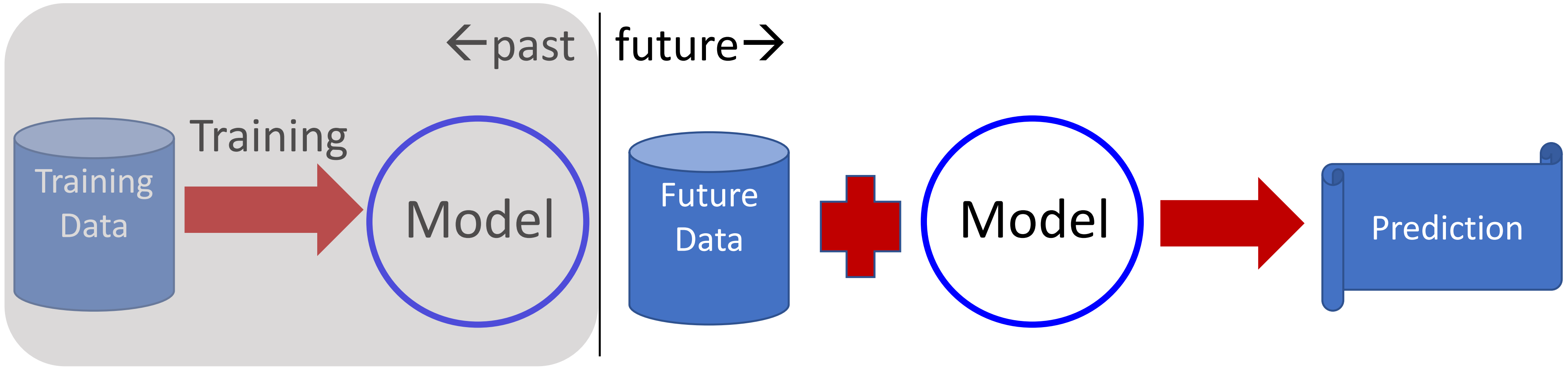


Figure generated by Liyan Song.

Generalization: Prediction Ability

- Generalization (泛化): the ability to produce reasonable outputs for inputs not encountered during the training process.

In other words: No PANIC when never-seen-before data are given to predict.

Supervised Learning: Classification (分类)

- Supervised learning: given some labeled examples.
- Classification: output is a finite set of labels.

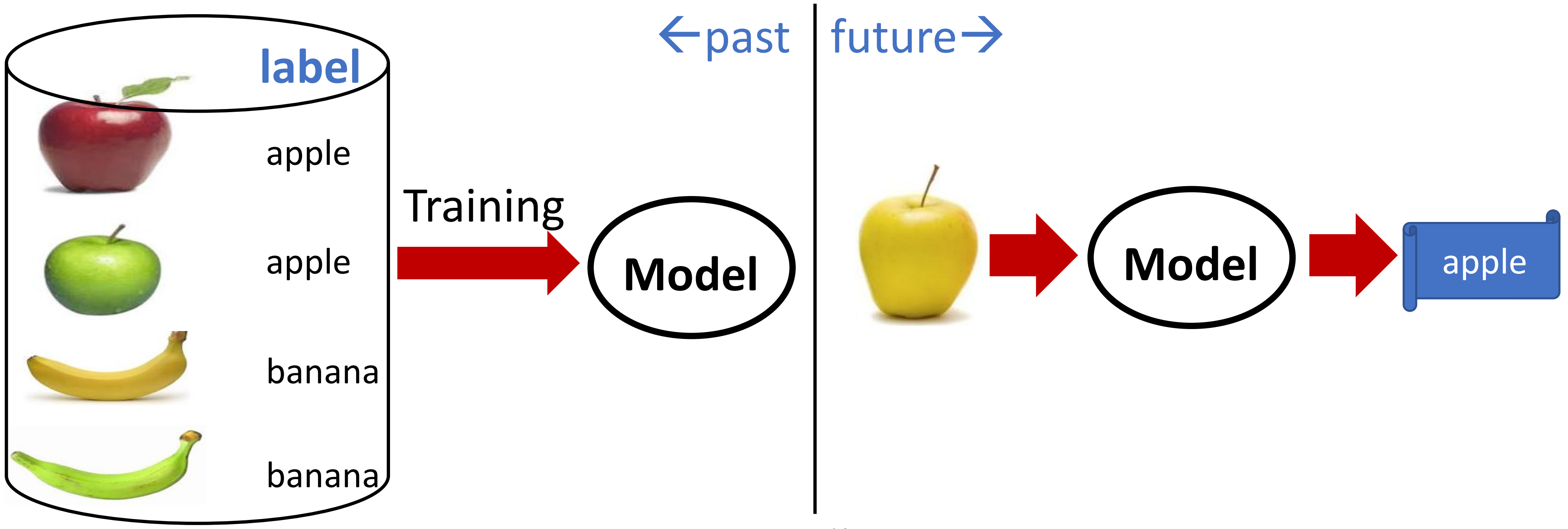


Figure generated by Liyan Song.

Supervised Learning: Regression (回归)

- Supervised learning: given some pairwise points.
- Regression: output is a real-valued number.

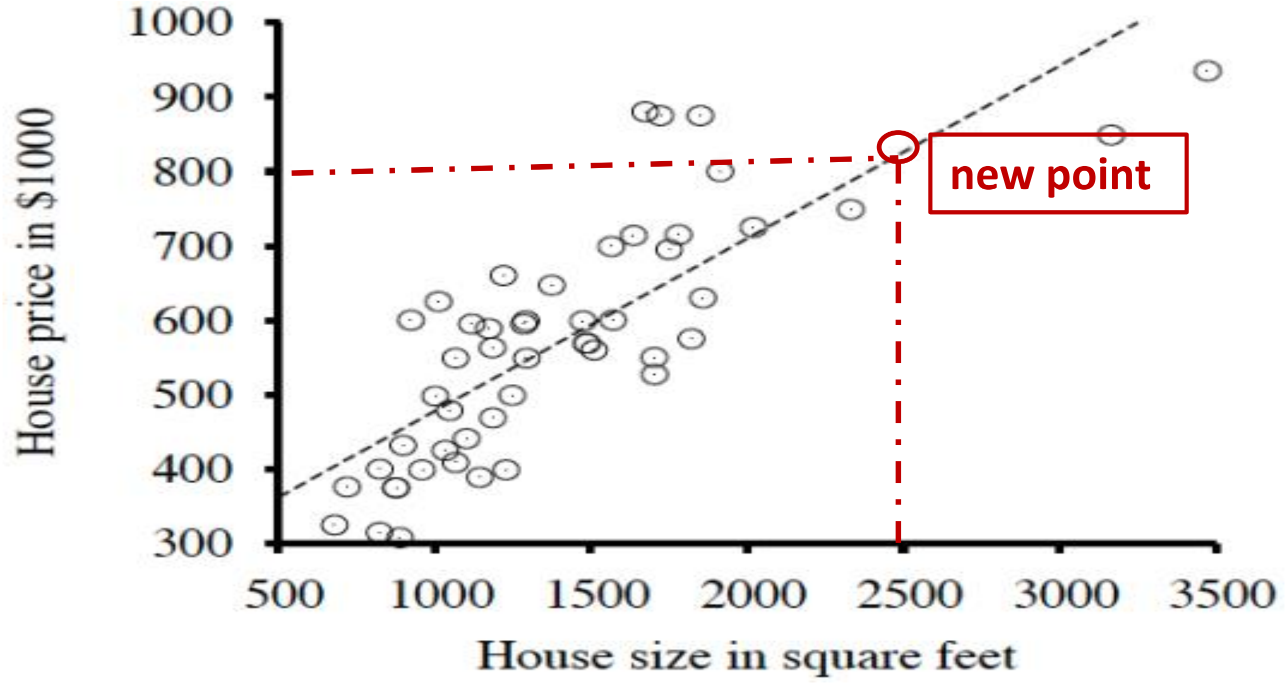


Figure 18.13 of the AI book by S. Russell & P. Novig.

Supervised Learning Formulation

- Given a set of pairwise examples $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\right)\right\}_{n=1}^{N}$ sampled i.i.d. from $\mathcal{X} \times \mathcal{Y}$. Each $\boldsymbol{y}^{(n)} = f\left(\boldsymbol{x}^{(n)}\right)$ is generated by an unknown function $f: \mathcal{X} \to \mathcal{Y}$.
  - $\boldsymbol{x}^{(n)} \in \mathcal{X} \subset \mathbb{R}^d$ is the input feature space.
  - $\boldsymbol{y}^{(n)} \in \mathcal{Y} \subset \mathbb{R}^1$ for regression, and $\boldsymbol{y}^{(n)}$ is discrete for classification.
  - $\mathcal{D}$: training data set.
- Find a function h^* from hypothesis space (假设空间) $\mathcal{H}$ to best approximate f

$$h^* = \arg\max_{h \in \mathcal{H}} p(h|\mathcal{D}).$$

Alternative Formulation

- Let
  - $y = f(x) + \varepsilon$, the *true* model with $\varepsilon \sim (0, \sigma^2)$
  - $\hat{y} = \hat{f}(x)$, the *estimated* model
- GOAL: Find $\hat{f}$ such that the discrepancy between y and $\hat{y}$ is minimized

Supervised Learning Process

- Learning: Learn a model $h \in \mathcal{H}$ using training set $\mathcal{D}$

  $\Rightarrow$ training error (训练误差) or empirical error (经验误差)

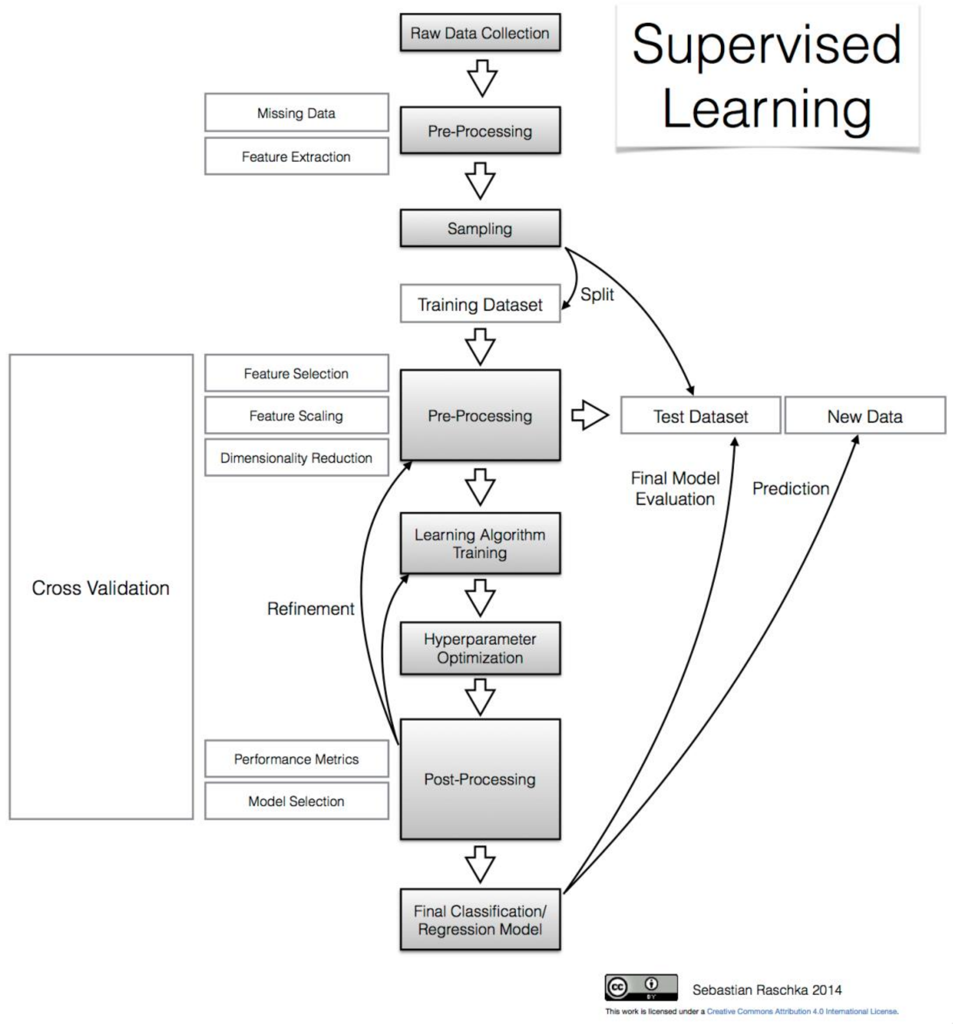
- Evaluation: Evaluate h on validation set $\mathcal{T} = \left\{\left(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}\right)\right\}$

  $\Rightarrow$ validation (test) error (测试误差)

  - Classification:
    - (e.g.) error rate $\frac{1}{T}\sum_t [y^{(t)} \neq h(x^{(t)})]$
  - Regression:
    - (e.g.) mean-square-error $\frac{1}{T}\sum_t \left[y^{(t)} - h\left(x^{(t)}\right)\right]^2$

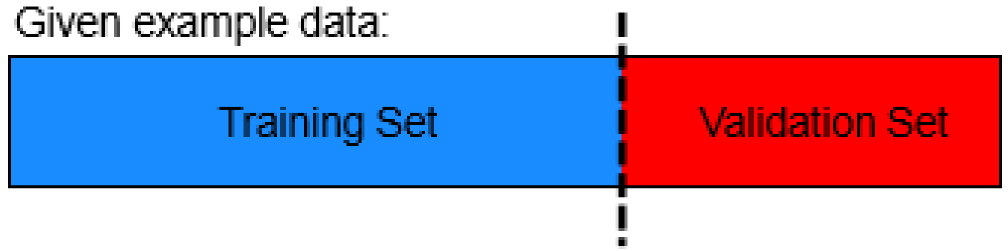
Learning process: major steps

- Iterative process
- If the model gives mediocre results on the testing/validation set, something needs to be changed, and then repeat the process
- Many possibilities for changes



Training, Validation and Testing Sets

- Divide the data into training, validation and testing sets, e.g.,
  - 60/20/20% for smaller datasets
  - 90/5/5% for large datasets
- Estimate parameters using the training set
- Check performance on the validation set ,and if unsatisfactory, make modifications to the model and re-compute the parameters using the training set
- Repeat if necessary
- Check the performance of the FINAL model on the test set (this is not part of training)



Cross Validation

- The hold out method
- K-fold cross validation
- Leave-one-out (LOO) cross validation

- Where/when do you use cross validation?

II Supervised Learning

1. What is Supervised Learning (formulation)
2. **Hypothesis Space $\mathcal{H}$ for Curve Fitting**

Example: Curve Fitting (曲线拟合)

- Fit a function to the points in the (x, y) plane, where $y = f(x)$.
  - Do not know the true function f.
  - Can only approximate f with an h from hypothesis space $\mathcal{H}$.

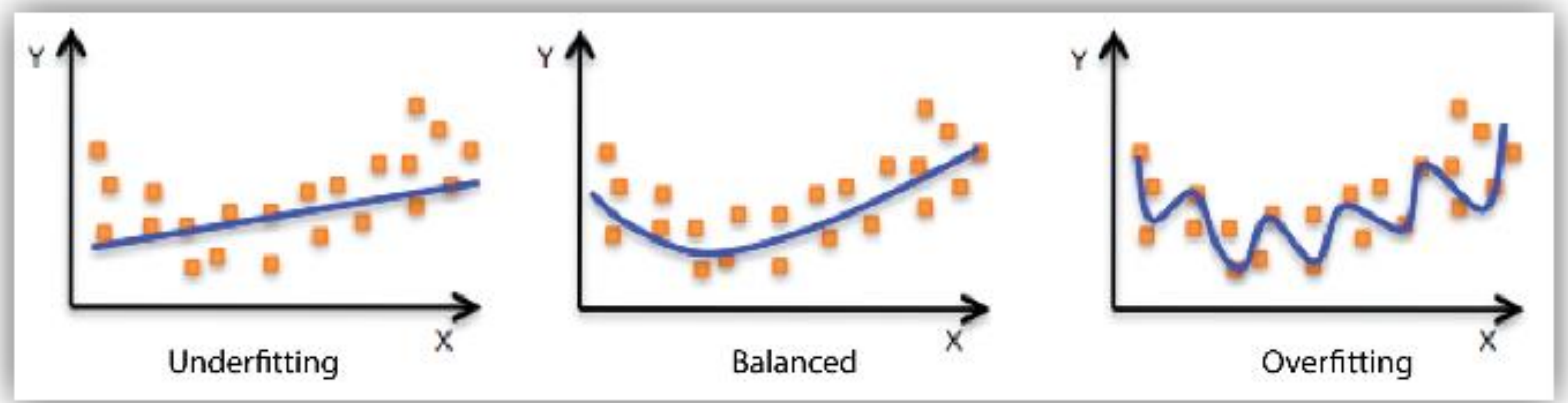


Image source: https://docs.aws.amazon.com/machine-learning/latest/dg/images/mlconcepts_image5.png。

Perfect Fit from Hypothesis Space

- $\mathcal{H}$(a): linear function, and $\mathcal{H}$(b): degree-7 polynomials.
  - Both can perfectly fit all the points.
- Question: Which $\mathcal{H}$ is better when both have the perfect fit?
- Answer: $\mathcal{H}$(a).
- Ockham's razor (奥卡姆剃刀): prefer simpler $\mathcal{H}$.
  - Fact: $\mathcal{H}$(a) linear function is simpler than $\mathcal{H}$(b) degree-7 polynomial.
- Define simplicity of $\mathcal{H}$ is NOT easy.
- Related to model selection.

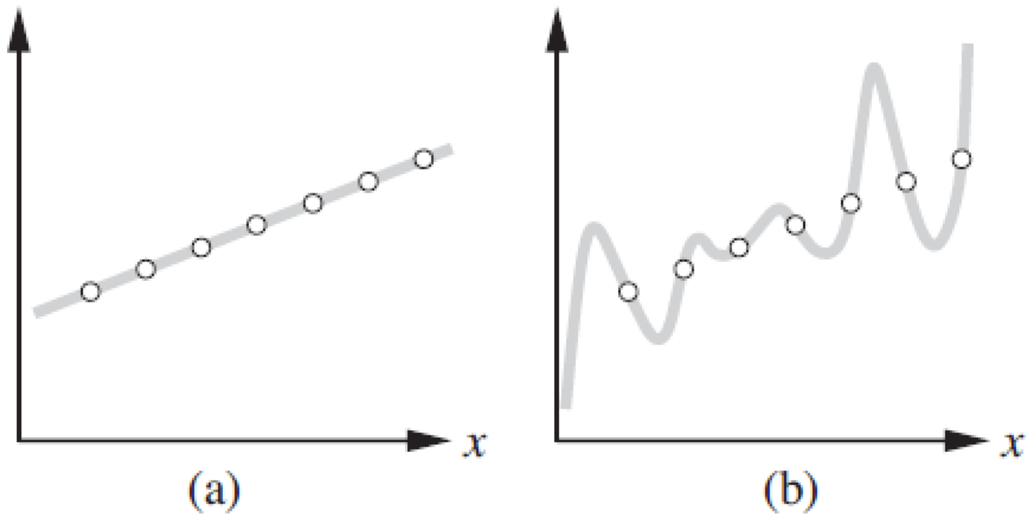


Figure 18.11 of the AI book by S. Russell & P. Novig.

Imperfect Fit from Hypothesis Space

- The points on the graph:
  - Cannot perfectly fit by any linear function from $\mathcal{H}$(a).
  - Can perfectly fit by a degree-6 polynomial $h_b^* \in \mathcal{H}$(b).
- Question: Which $\mathcal{H}$ is better?
- Consideration: fit on seen vs generalize on unseen
  - $\mathcal{H}$(a): not perfectly fit but might generalize well.
  - $\mathcal{H}$(b): perfectly fit but might not generalize well.
- Answer: Trade-off between complex $\mathcal{H}$ that fit the training data well and simpler $\mathcal{H}$ that may generalize better.

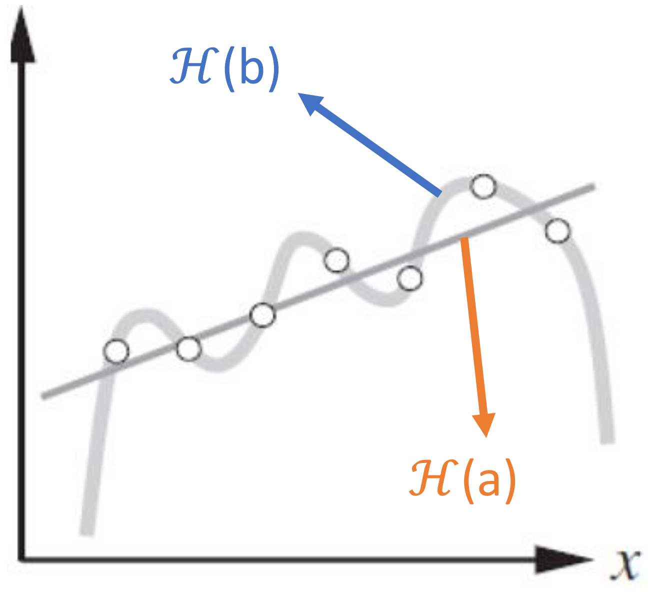
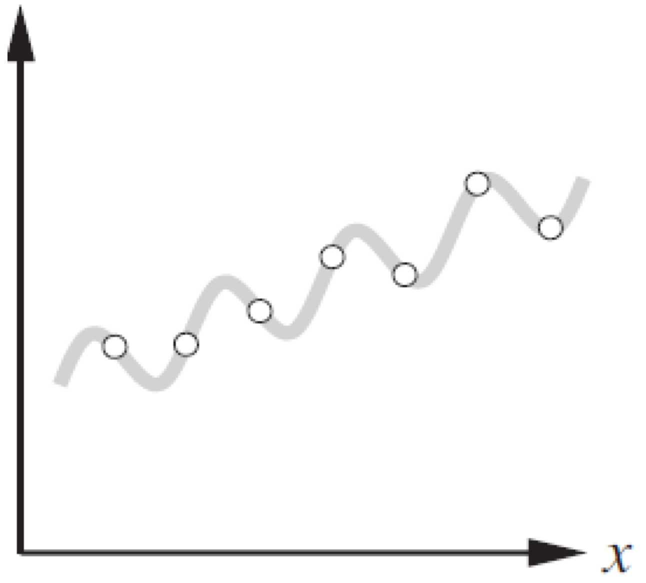


Figure 18.11 of the AI book by S. Russell & P. Novig.

Enriched Hypothesis Space $\mathcal{H}$(c)

- $\mathcal{H}$(c): polynomials over both x and $\sin(x)$.
- $h_c^* = ax + b + c \cdot \sin(x)$ can perfectly fit.
- h_c^* is 'simpler' than h_b^* (only 3 parameters).

➢ Choice of hypothesis space $\mathcal{H}$ is important.



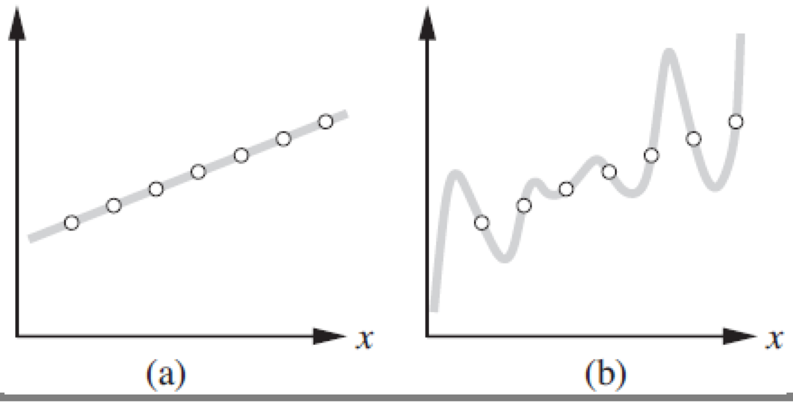


Figure 18.11 of the AI book by S. Russell & P. Novig.

Underfitting vs Overfitting (Regression)

- Underfitting (欠拟合): too simple $\mathcal{H}$

  $\Rightarrow$ cannot capture the underlying pattern.
- Overfitting (过拟合): too complex $\mathcal{H}$

  $\Rightarrow$ perfect on the given data, but much worse on unseen data.

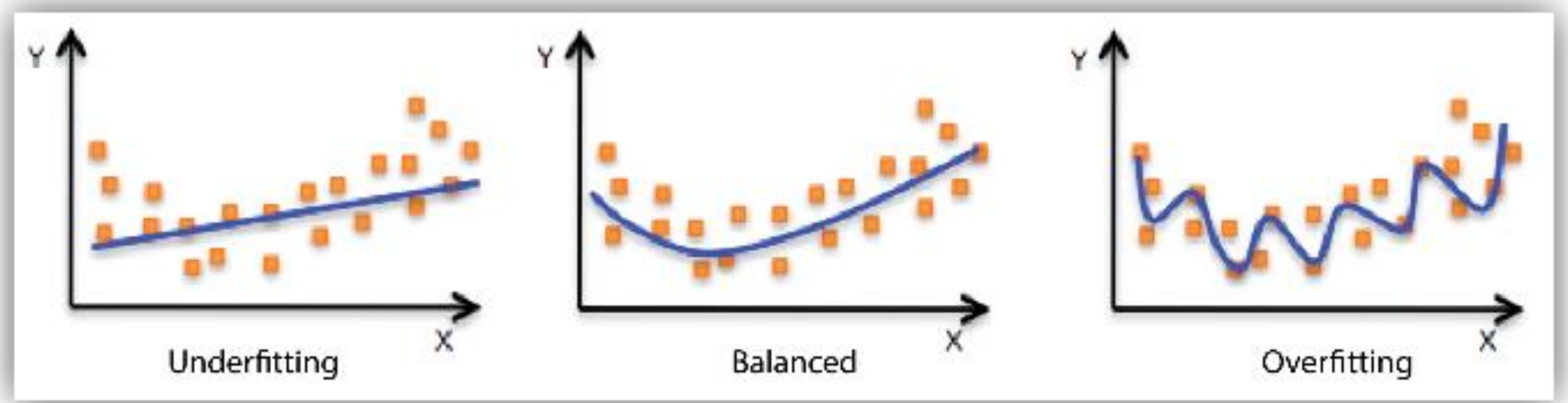


Image source: https://docs.aws.amazon.com/machine-learning/latest/dg/images/mlconcepts_image5.png。

Underfitting vs Overfitting (Classification)

- Underfitting (欠拟合): too simple $\mathcal{H}$

  $\Rightarrow$ cannot capture the underlying pattern.

- Overfitting (过拟合): too complex $\mathcal{H}$

  $\Rightarrow$ perfect on the given data, but much worse on unseen data.

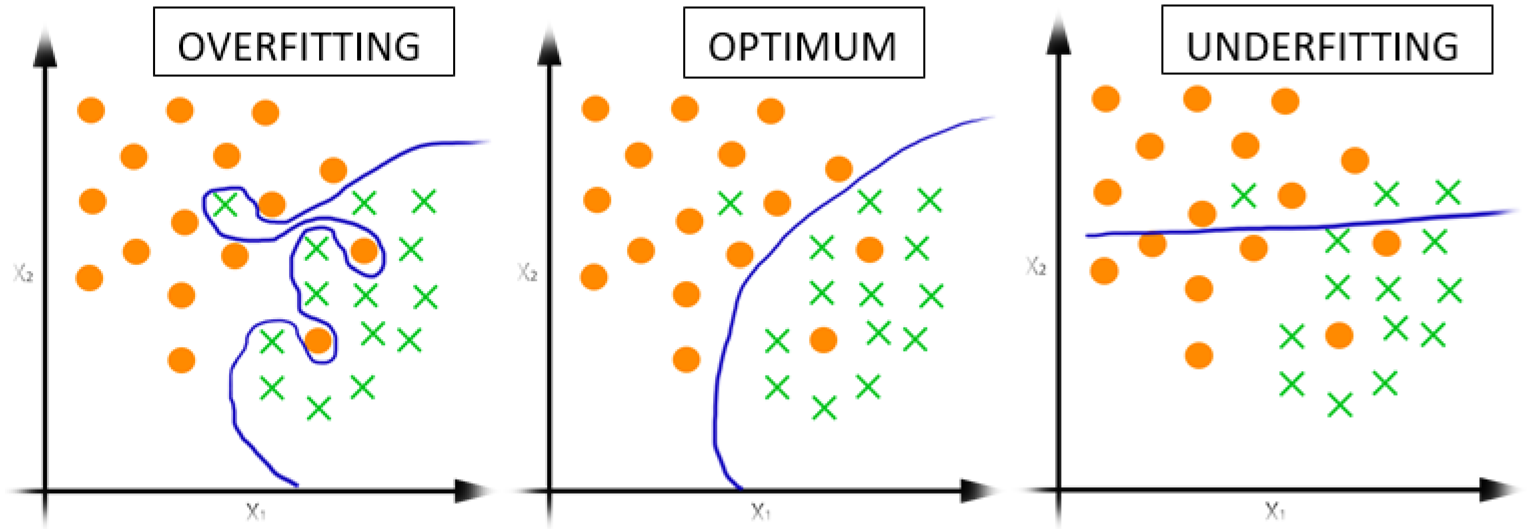


Figure by Sachin Joglekar@Google.

Underfitting and Overfitting

- Underfitting:
  - How to recognize? High training error and high test error.
  - How to deal with it? Use a more complex $\mathcal{H}$.

- Overfitting:
  - How to recognize? Low training error but high test error.
  - How to deal with it?
    - (1) Use a less complex $\mathcal{H}$.
    - (2) Regularization (正则化): penalize certain parts of the parameter space or introduce additional constraints to constrain the hypothesis space.
    - (3) Get more training data.

III Computational Learning Theory

1. **Bias-variance trade-off**
2. Model selection
3. Probably approximately correct (PAC) learning

Issues

- How should we understand supervised learning?
- What are the fundamentals of supervised learning?

Bias-variance Trade-off

- Let
  - $y = f(x) + \varepsilon$, *true* model with $\varepsilon \sim (0, \sigma^2)$
  - $\hat{y} = \hat{f}(x)$, the *estimated* model
- GOAL: Find $\hat{f}$ such that the discrepancy between y and $\hat{y}$ is minimized
- For any variable Z

$$\mathrm{var}(Z) = E\left[(Z - E[Z])^2\right] = E\left[Z^2\right] - (E[Z])^2$$

$$E[Z^2] \; = \mathrm{var}(Z) + (E\;[Z])^2$$

- Objective (loss) function

$$L(x) = E_{\mathcal{D}}(y - \hat{f}(x))^2 = E[y^2] - 2E_{\mathcal{D}}[y\hat{f}(x)] + E_{\mathcal{D}}[\hat{f}(x)^2]$$

Bias-variance Trade-off

- Individual terms

$$E\left[y^2\right] = \mathrm{var}(y) + (E[y])^2 = \sigma^2 + f(x)^2$$
$$-2E_{\mathcal{D}}[y\hat{f}(x)] = -2f(x)E_{\mathcal{D}}[\hat{f}(x)]$$
$$E_{\mathcal{D}}[\hat{f}(x)^2] = \mathrm{var}\,(\hat{f}(x)) + (E_{\mathcal{D}}[\hat{f}(x)])^2$$

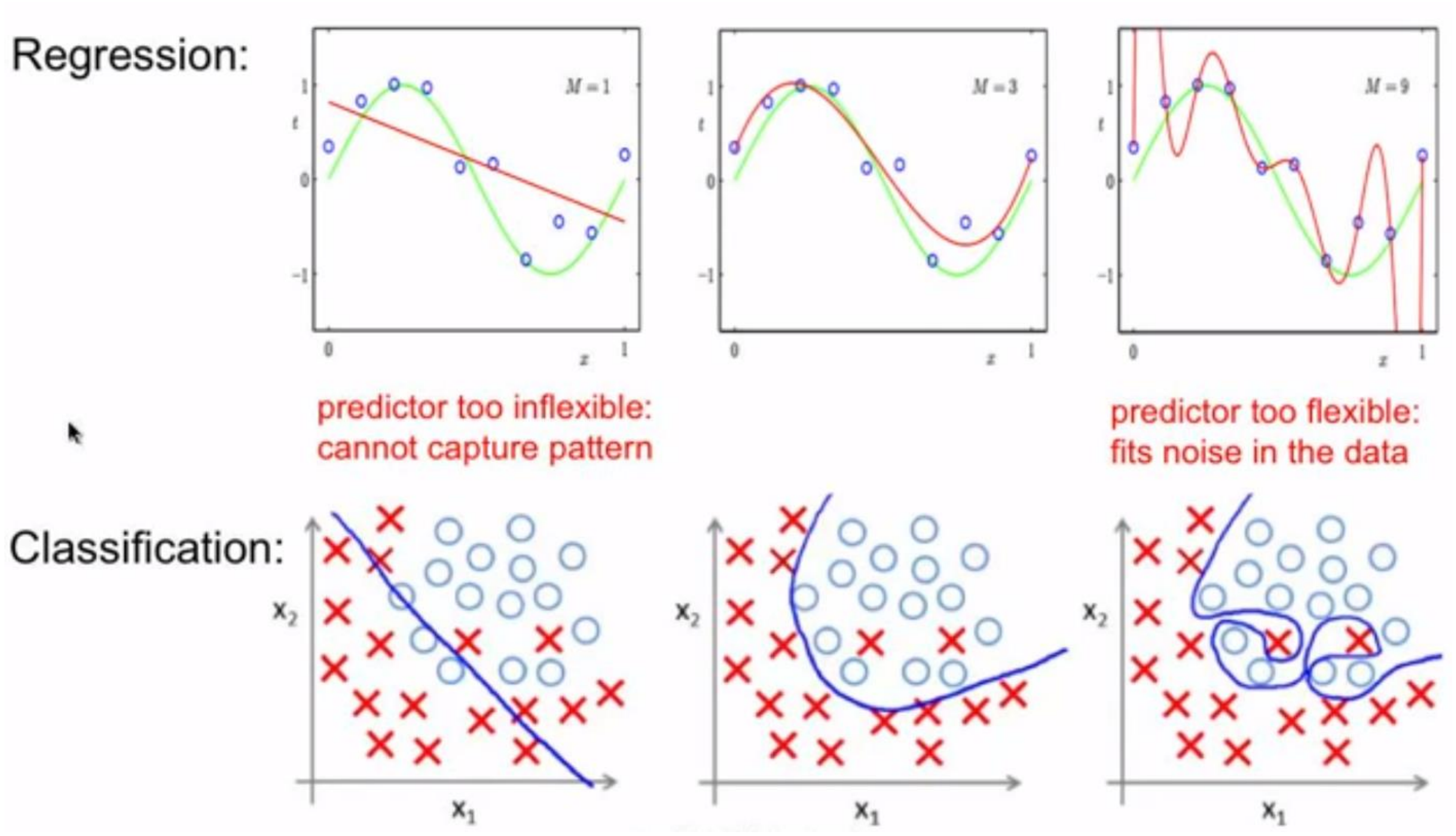
- All terms combined

$$\begin{aligned} L(x) &= \sigma^2 + \mathrm{var}\,(\hat{f}(x)) + f(x)^2 - 2f(x)E_{\mathcal{D}}[\hat{f}(x)] + (E_{\mathcal{D}}[\hat{f}(x)])^2 \\ &= \sigma^2 + \mathrm{var}\,(\hat{f}(x)) + (f(x) - E_{\mathcal{D}}[\hat{f}(x)])^2 \\ &= \sigma^2 + \mathrm{var}\,(\hat{f}(x)) + \mathrm{bias}^2\hat{f}(x) \end{aligned}$$

Bias-variance Trade-off

- The loss function consists of three terms
  - Irreducible error: σ^2
    - Bound on the algorithm performance
    - Uncertainty in the data
  - Squared bias: $\mathrm{bias}^2 \hat{f}(x)$
    - Error due to simplification in the model
    - Difference between $f(x)$ and $E\hat{f}(x)$
    - Performance on the training data
  - Variance: $\mathrm{var}\, \hat{f}(x)$
    - How much the estimate “jumps” around its mean
    - How well the method generalizes on different testing data

Connection to Overfitting and Underfitting



Figures generated by Xuantian Lin (Taiwan University).

Complexity Choice

- With the increasing model complexity, the error
  - On the training set decreases
  - On the testing set first decreases and then starts increasing

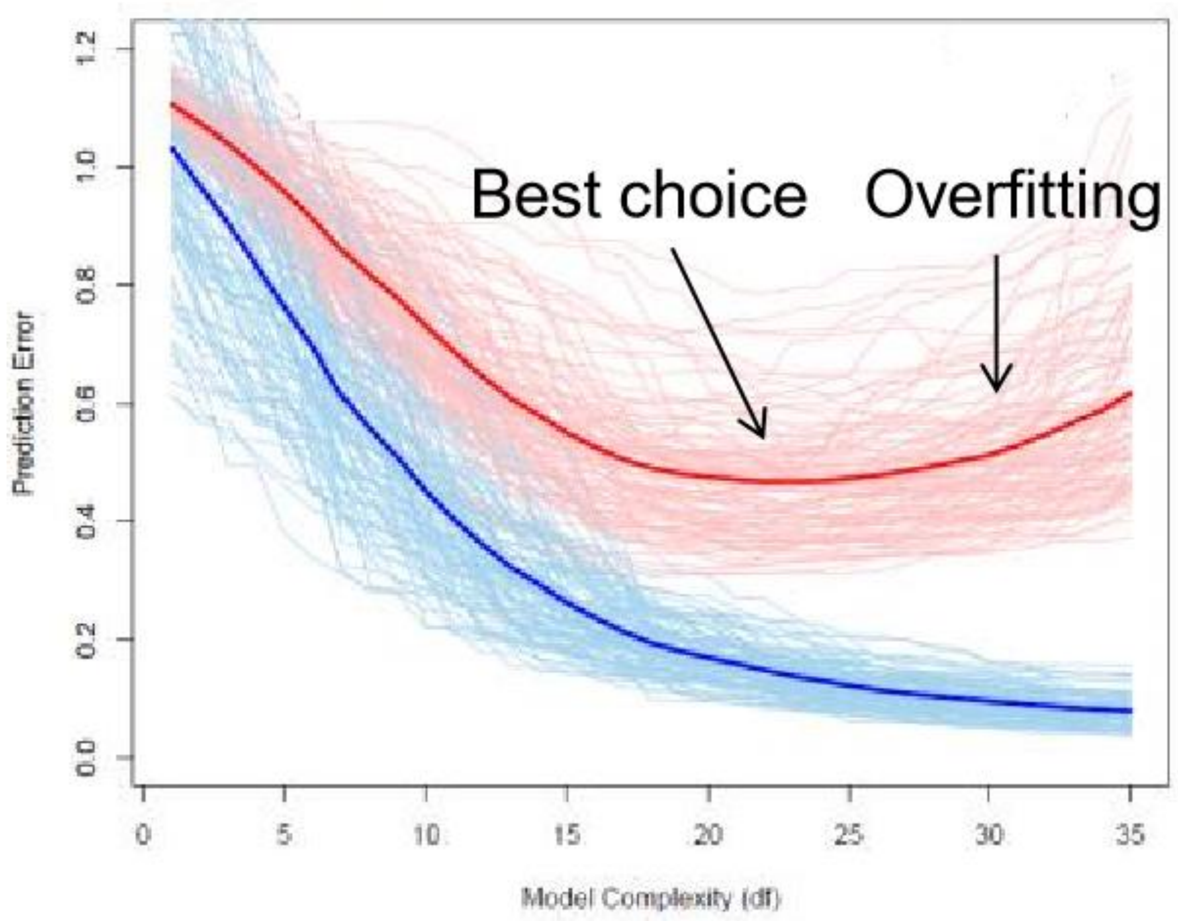


Image source: Hastie, Tibshirani, Friedman (2009).

Complexity Choice

- Optimal model complexity
- Underfitting: to the left
- Overfitting: to the right

- Trade-off!

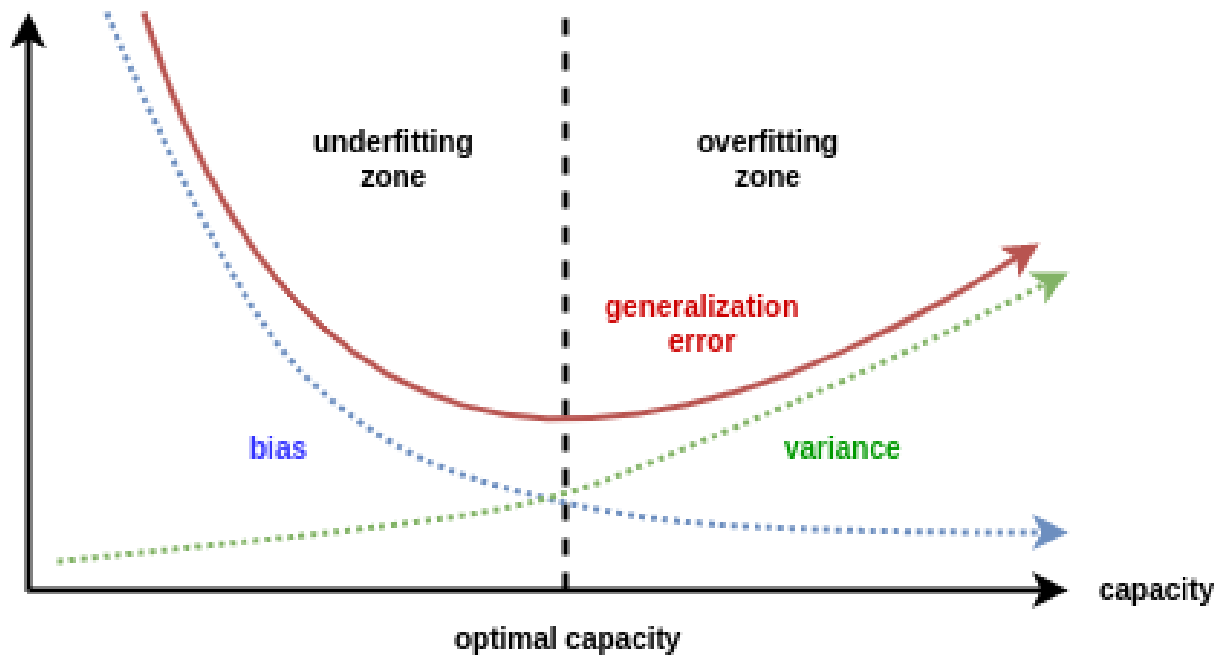


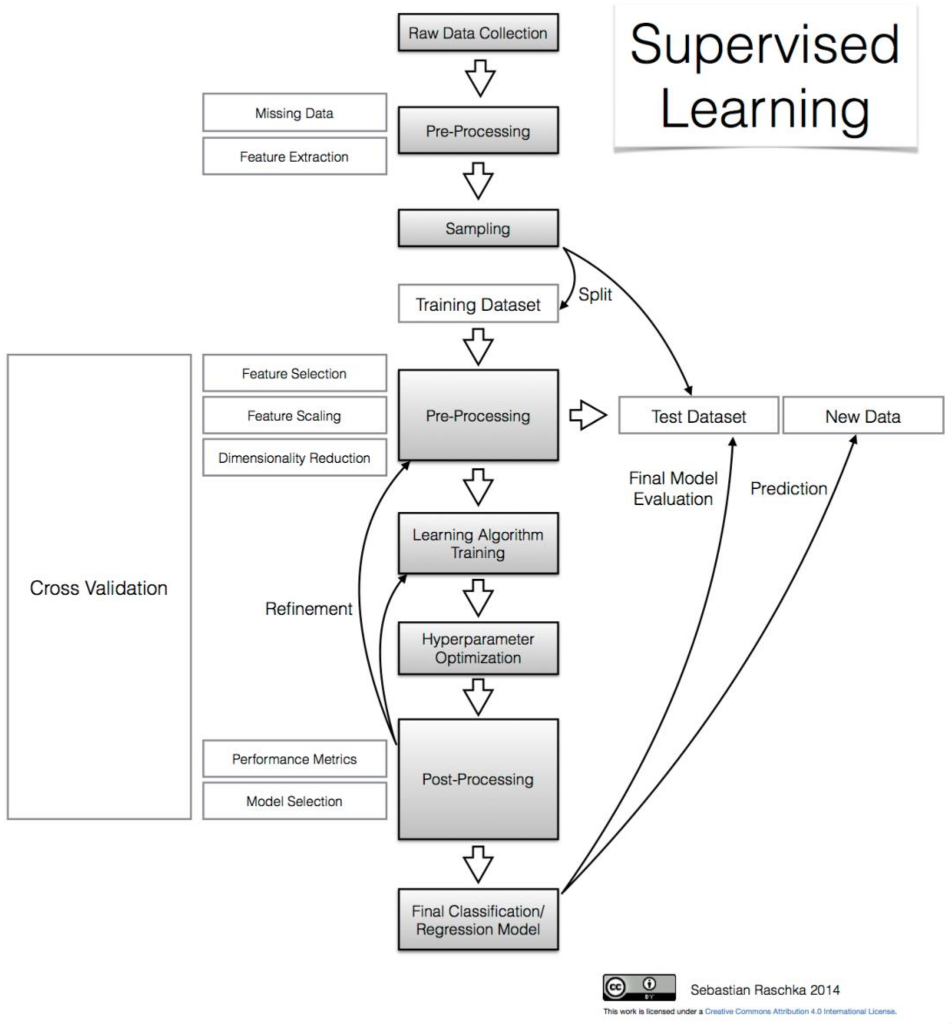
Image source: https://djsaunde.wordpress.com/2017/07/17/the-bias-variance-tradeoff/

III Computational Learning Theory

1. Bias-variance trade-off
2. **Model selection**
3. Probably approximately correct (PAC) learning

Learning process: major steps

- Iterative process
- If the model gives mediocre results on the testing/validation set, something needs to be changed, and then repeat the process
- Many possibilities for changes



What to do when something is wrong?

- Bad performance on the training set (high bias)
  - More complex model, different model, change hyperparameters, normalize inputs, train longer, change starting points, more complex optimization procedure, …
- Good performance on the training set, bad performance on the validation set (high variance)
  - Simpler model, more data in the training set, regularization, feature selection, …
- Good performance on the training set, good performance on the validation set
  - Done

Input normalization

- Instead of $(x_1, \dots, x_n)$ consider $(z_1, \dots, z_n)$

$$z_i = \frac{x_i - \mu}{\sigma}$$

$$\mu = \frac{1}{n} \sum x_i$$

$$\sigma^2 = \frac{1}{n-1} \sum (x_i - \mu)^2$$

- Then $Z \sim (0, I)$
- Eliminates strong impact of one variable
- IMPORTANT: Use the same coefficients μ and σ for the training and testing data

Increasing training set size

- Individual observations have smaller impact (reduces outliers)
- Difference with increasing model complexity:
  - Increase training set size: Optimization algorithm (mostly) unchanged
  - Increase model complexity: Optimization algorithm takes longer to converge. Convergence issues may occur (later)
- How to add observations?
  - Gather them: May be costly
  - Add artificial observations/Data augmentation: for example flip or rotate images. May improve the behaviour but the benefit is not so great as for the first option. It is cheap.

Regularization

- Reduces the model complexity by forcing some parameters to approach zero
- For linear regression, modifies the objective by adding regularization

$$\text{minimize } \|y - Xw\|^2 + \lambda R(w)$$

- Possible regularizations
  - $R(w) = \|w\|_0$ (sparse). Number of non-zeros. Discontinuous. Difficult
  - $R(w) = \|w\|_1$ (l_1 penalization, LASSO). Piecewise linear. Simpler
  - $R(w) = \|w\|_2^2$ (l_2 penalization, Ridge regression, Tikhonov regularization, …). Differentiable. Simplest
- Regularization parameter (weight) $\lambda \geq 0$
  - $\lambda = 0 \Rightarrow$ no effect
  - $\lambda \approx \infty \Rightarrow w \approx 0$

Feature selection

- GOAL: Reduce the number of features, set some w_j to zero
- It can be shown that
  - l_2 regularization shrinks w_j towards zero
  - l_1 regularization sets w_j to zero if below a certain threshold
  - l_0 regularization counts non-zeros in w_j and tries to set as many of them to zero as possible
- Another idea for feature selection
  - Filter methods: Run the model and select the active features based on some criterion (correlation, largest values of w_j, …)
  - Wrapper methods: Run the model repeatedly and subsequently add or remove features

Feature selection: Wrapper methods

- Wrapper methods include forward and backward stepwise methods
- Forward method
  - Start with empty feature set $F = \emptyset$
  - For all $j \notin F$ evaluate performance for active features $F \cup \{j\}$
  - Add the best performing j to F and repeat until a stopping criterion is satisfied (maximal number of features, the performance is stable, …)
- Backward method
  - The same but start with full set $F = \{1, \dots, d\}$
  - And removes features one after another
- Combination of forward and backward stepwise method (adding or removing more features at the same time) is possible

Training v.s. Testing

- The basic requirement is that the training, validation and test sets follow the same distribution
- Otherwise, the model is trained for a behavior but tested for another one
- Sometimes a discrepancy between distributions happens naturally
  - You test your app for cat recognition on many pictures downloaded from the Internet (good quality pictures) + few pictures uploaded by users (bad quality pictures)
  - But you are only interesting in the performance on the second category
- Possible solutions
  - Add artificial blurring effect to good pictures
  - Increase weights in the objective function for bad quality pictures

Design of Training and Validation Sets

- Given a set of samples, how to design a training set and a test set?
  - Hold-out (留出法) cross-validation: separate to 2 compliment subsets.
    - Random sampling (随机采样): no bias, no knowledge is used.
    - Stratified sampling (分层采样): biased sampling based on the labels of the given set.
  - k-fold cross-validation (k折交叉验证法): separate to k subsets.

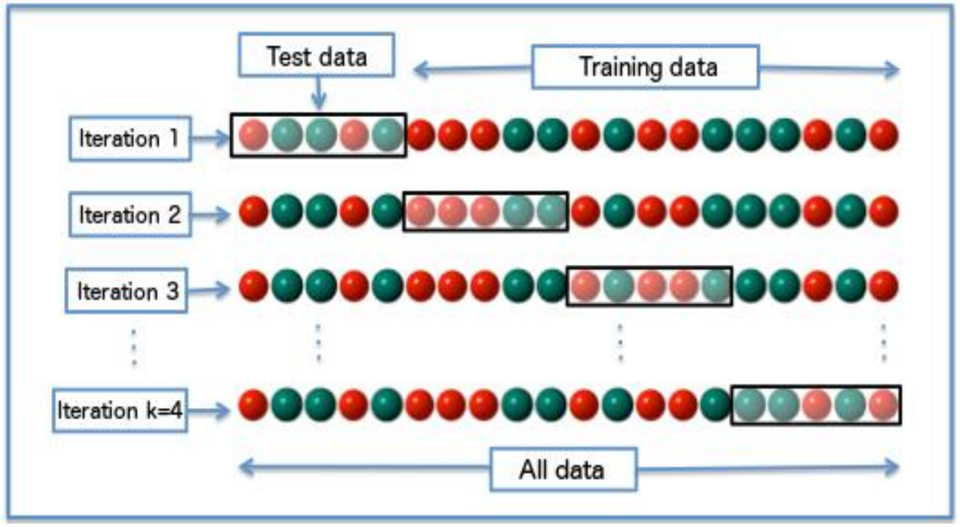


Image source: https://en.wikipedia.org/wiki/Cross-validation_(statistics)#/media/File:K-fold_cross_validation_EN.jpg

III Computational Learning Theory

1. Bias-variance trade-off
2. Model selection
3. **Probably approximately correct (PAC) learning**

PAC Learning

- PAC (probably approximately correct) learning studies how many examples are needed to show that all consistent hypotheses are "good"
- Consider a hypothesis space $\mathcal{F}$, its subset $\mathcal{F}_{bad}$ of bad hypotheses and assume that observations follow distribution D
- For $\hat{f} \in \mathcal{F}$ define the error function
$$e(\hat{f}) = P(\hat{f}(x) \neq f(x) | x \sim D)$$
- We say that a hypothesis $\hat{f} \notin \mathcal{F}_{bad}$ is approximately correct if
$$e(\hat{f}) \leq \varepsilon$$

PAC Learning

- For incorrect hypothesis $\hat{f} \in \mathcal{F}_{bad}$ we have
$$P(\hat{f}(x) = f(x) |\ x \sim D) \leq 1 - \varepsilon$$
- Considering n observations
$$P(\hat{f}(x_i) = f(x_i) |\ x_1, \dots, x_n \sim D \text{ iid}) \leq (1 - \varepsilon)^n$$
- Thus the chance that $\mathcal{F}_{bad}$ contains at least one hypothesis which is consistent on the n observations is bounded above by $|\mathcal{F}|(1 - \varepsilon)^n$
- If our algorithm returns a hypothesis which is consistent on n examples, we want to minimize the chance that it is probably incorrect, thus we require
$$|\mathcal{F}|(1 - \varepsilon)^n \leq \delta$$

PAC Learning

- Assume that
$$n \geq \varepsilon^{-1}(\log|\mathcal{F}| - \log\delta)$$
- Then from $e^x \geq 1 + x$ and the assumption above
$$n\log(1-\varepsilon) \leq n\log e^{-\varepsilon} = -n\varepsilon \leq \log\delta - \log|\mathcal{F}|$$
- This is equivalent to
$$|\mathcal{F}|(1-\varepsilon)^n \leq \delta$$
- Unfortunately, if $\mathcal{F}$ is a set of all Boolean functions of k attributes, we have $|\mathcal{F}| = 2^{2^k}$, and thus approximately $n \geq \varepsilon^{-1}(2^k - \log\delta)$ has the exponential growth in this case

Summary of this lecture

- What is Machine Learning
- Supervised Learning
  - Formulation
  - Hypothesis Space $\mathcal{H}$ for Curve Fitting
- Computational Learning Theory
  - Bias-variance trade-off
  - Model selection
  - Probably approximately correct (PAC) learning

Reading Materials for This Lecture

- [1] Gradient Descent: http://ruder.io/optimizing-gradient-descent/
- [2] AI book (P693-704 & P708-718).
- [3] Hastie, Tibshirani, Friedman, The Elements of Statistical Learning (Chapters 2, 7, 10)
- [4] L.G. Valiant, A Theory of the Learnable, *Communications of the ACM* 27(11):1134–1142 (1984).