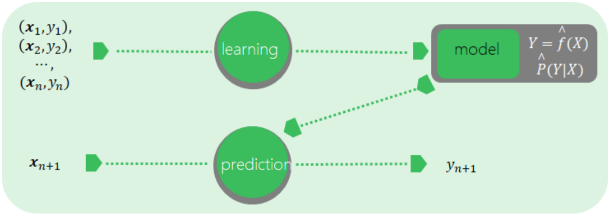
Representation of Data

- Input space $\mathcal{X} = \{\text{All possible samples}\}$; $\mathbf{x} \in \mathcal{X}$ is an input vector, also called feature, predictor, independent variable, etc. ; typically multi-dimensional ; e.g., $\mathbf{x} \in \mathbb{R}^p$ is a weight vector or coding vector
- Output space $\mathcal{Y} = \{\text{All possible results}\}$; $y \in \mathcal{Y}$ is an output vector, also called response, dependent variable, etc. ; typically one-dimensional ; e.g., $y = 0$ or 1 for classification problems, $y \in \mathbb{R}$ for regression problems.
- For supervised learning, assume that $(\mathbf{x}, y) \sim \mathcal{P}$, a joint distribution on the sample space $\mathcal{X} \times \mathcal{Y}$

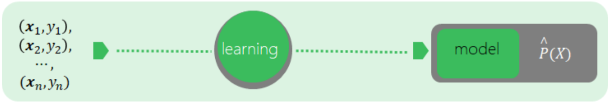
Supervised Learning

- Goal : given $\mathbf{x}$, predict what is y ; in deterministic settings, find the dependence relation $y = f(x)$; in probabilistic settings, find the conditional distribution $P(y|\mathbf{x})$ of y given $\mathbf{x}$
- Training dataset : $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n} \overset{i.i.d.}{\sim} \mathcal{P}$, used to learn an approximation $\hat{f}(x)$ or $\hat{P}(y|\mathbf{x})$
- Test dataset : $\{(\mathbf{x}_j, y_j)\}_{j=n+1}^{n+m} \overset{i.i.d.}{\sim} \mathcal{P}$, used to make a prediction $\hat{y}_j = \hat{f}(\mathbf{x}_j)$ or $\hat{y}_j = \arg\max_{y_j} \hat{P}(y_j|\mathbf{x}_j)$, and verify how accurate the approximation is



Unsupervised Learning

- Goal : in probabilistic settings, find the distribution $P(\mathbf{x})$ of $\mathbf{x}$ and approximate it ; there is no y
- Training dataset : $\{\mathbf{x}_i\}_{i=1}^{n} \overset{i.i.d.}{\sim} \mathcal{P}$, used to learn an approximation $\hat{P}(\mathbf{x})$; no test data in general



Learning Models

- Decision function (hypothesis) space : $\mathcal{F} = \{f_\theta | f_\theta = f_\theta(\mathbf{x}), \theta \in \Theta\}$ or $\mathcal{F} = \{P_\theta | P_\theta = P_\theta(y|\mathbf{x}), \theta \in \Theta\}$
- Loss function : a measure for the "goodness" of the prediction, $L(y, f(\mathbf{x}))$
  - 0-1 loss : $L(y, f(\mathbf{x})) = I_{y \neq f(\mathbf{x})} = 1 - I_{y = f(\mathbf{x})}$
  - Square loss : $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$
  - Absolute loss : $L(y, f(\mathbf{x})) = |y - f(\mathbf{x})|$
  - Cross-entropy loss : $L(y, f(\mathbf{x})) = -y \log f(\mathbf{x}) - (1 - y) \log(1 - f(\mathbf{x}))$
- Risk : in average sense, $R(f) = E_P[L(y, f(\mathbf{x}))] = \int_{\mathcal{X} \times \mathcal{Y}} L(y, f(\mathbf{x})) P(\mathbf{x}, y) \mathrm{d}\mathbf{x}\mathrm{d}y$
- Target of learning : choose the best f^* to minimize $R(f)$, $f^* = \min_f R(f)$

Risk Minimization Strategy

- Empirical risk minimization (ERM) : given training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $R_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i))$
  - By law of large number, $\lim_{n\to\infty} R_{emp}(f) = R(f)$
  - Optimization problem : $\min_{f\in F} \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i))$
- Structural risk minimization (SRM) : given training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and a complexity functional $J = J(f)$, $R_{srm}(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$
  - $J(f)$ measures how complex the model f is, typically the degree of complexity
  - $\lambda \geqslant 0$ is a tradeoff between the empirical risk and model complexity
  - Optimization problem : $\min_{f\in F} \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$

Algorithms

- Computational methods to solve the problem for f
- Numerical methods to solve the optimization problems
  - Gradient descent method, including coordinate descent, sequential minimal optimization (SMO), etc.
  - Newton's method and quasi-Newton's method
  - Combinatorial optimization
  - Genetic algorithms
  - Monte Carlo methods
  - ...

Model Assessment

Assume we have learned the model $y = \hat{f}(\mathbf{x})$, what is the error ?

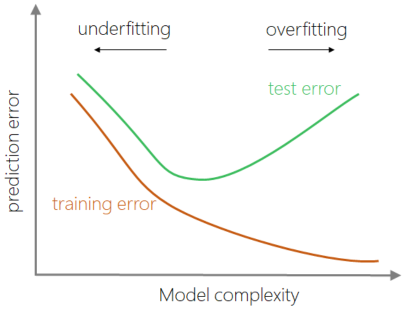
- Training error : $R_{emp}(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}(\mathbf{x}_i))$, tells the difficulty of learning problem
- Test error : $e_{test}(\hat{f}) = \frac{1}{m}\sum_{j=n+1}^{n+m} L(y_j, \hat{f}(\mathbf{x}_j))$, tells the capability of prediction ; in particular, if 0-1 loss is used
    - Error rate : $e_{test}(\hat{f}) = \frac{1}{m}\sum_{j=n+1}^{n+m} I_{y_j \neq \hat{f}(\mathbf{x}_j)}$
    - Accuracy : $r_{test}(\hat{f}) = \frac{1}{m}\sum_{j=n+1}^{n+m} I_{y_j = \hat{f}(\mathbf{x}_j)}$
    - $e_{test} + r_{test} = 1$

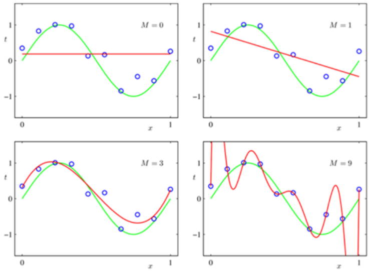
Model Assessment (Cont')

- Generalization error : $R_{exp}(\hat{f}) = E_P[L(y, \hat{f}(\mathbf{x}))] = \int_{\mathcal{X}\times\mathcal{Y}} L(y, \hat{f}(\mathbf{x}))P(\mathbf{x}, y)\mathrm{d}\mathbf{x}\mathrm{d}y$, tells the capability for predicting unknown data from the same distribution, its upper bound M defines the generalization ability
  - As $n \to \infty$, $M \to 0$
  - As F becomes larger, M increases

Overfitting

- Too many model parameters
- Better for training set, but worse for test set

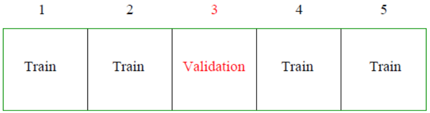




fitting of degree M polynomial, green curve is the ground truth

Model Selection

- Regularization : $\min_{f \in F} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \underbrace{\lambda J(f)}_{penalty}$, choose λ to minimize empirical risk and model complexity simultaneously
- Cross-validation (CV) : split the training set into training subset and validation subset, use training set to train different models repeatedly, use validation set to select the best model with the smallest (validation) error
    - Simple CV : randomly split the data into two subsets
    - K-fold CV : randomly split the data into K disjoint subsets with the same size, treat the union of $K - 1$ subsets as training set, the other one as validation set, do this repeatedly and select the best model with smallest mean (validation) error
    - Leave-one-out CV : $K = n$ in the previous case


1
2
3
4
5
Train
Train
Validation
Train
Train