Traing versus Testing and Linear Regularization

Tong Zhang

Rutgers University

Supervised Learning Problem

- Prediction problem
  - Input X: known information.
  - Output Y: unknown information.
  - Goal: to predict Y based on X
    find a function (prediction rule) $f(X)$ such that $Y \approx f(X)$
- Observe historical data $S_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$
  - S_n called training data
- Learning: find a good prediction rule $f(x)$ based on S_n

Supervised Learning Problem

- Prediction problem
  - Input X: known information.
  - Output Y: unknown information.
  - Goal: to predict Y based on X
    find a function (prediction rule) $f(X)$ such that $Y \approx f(X)$
- Observe historical data $S_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$
  - S_n called training data
- Learning: find a good prediction rule $f(x)$ based on S_n
- Example: least squares regression
  - $f(x) = w^T x$: linear prediction rule
  - learning algorithm: find $\hat{w}$ to minimize squared error on training data

$$\hat{w} = \arg\min_w \sum_{i=1}^n (f(X_i) - Y_i)^2$$

Training versus Testing

- Training data: $S_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$
- Learning: find a prediction rule on training data
  - find a rule to fit well on S_n

Training versus Testing

- Training data: $S_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$
- Learning: find a prediction rule on training data
  - find a rule to fit well on S_n
- Test data: future observations that may be different from training data
  - purpose: learned prediction rule should work well on test data

Training versus Testing

- Training data: $S_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$
- Learning: find a prediction rule on training data
  - find a rule to fit well on S_n
- Test data: future observations that may be different from training data
  - purpose: learned prediction rule should work well on test data
- Learning process:
  - learn prediction rule on training data
  - evaluate performance on test data
- Why separate training and testing?
  - training error is usually unrealistically low (overfitting)
  - error on test data (data not used to fit model) is more realistic

Overfitting

- Goal: predict well on future unseen data (test data)
- Two aspects:
  - rule should fit training data well
    - requires a more complex model.
  - behavior of rule on test data should match that on training data
    - requires a less complex (more stable) model.

Overfitting

- Goal: predict well on future unseen data (test data)
- Two aspects:
  - rule should fit training data well
    - requires a more complex model.
  - behavior of rule on test data should match that on training data
    - requires a less complex (more stable) model.
- Model complexity:
  - more complex model: smaller training error but larger difference between test and training error
  - less complex model: larger training error but smaller difference between test and training error
- Related concepts:
  - bias variance trade-off
  - regularization

Overfitting: Simple Example

- Binary classification example:
  - Input X is one dimensional and uniformly distributed in $[-1, 1]$
  - True class label
$$Y = \begin{cases} 1 & X \geq 0 \\ -1 & X < 0 \end{cases}$$
- Given training data (X_i, Y_i) $(i = 1, \ldots, n)$
  - with probability one X_i are all different
- Assume you don't know the relationship of X and Y but want to learn it from data.

Overfitting Method

The following prediction rule fits training data perfectly

$$f(X) = \begin{cases} Y_i & \text{if } X = X_i \text{ for some } i \\ 1 & \text{otherwise} \end{cases}$$

- Has no prediction capability when X not in the training data.
- Very complex prediction rule – arbitrary flexibility.
- Characteristics of overfitting procedure:
  - training error: small (0)
  - difference of test error and training error: large (0.5)
  - test error: large (0.5)

Underfitting method

Simply return the following prediction rule independent of the data

$$f(X) = 1.$$

- Ignore the data: fits training data poorly
- Simplistic prediction rule — no flexibility.
- Characteristics of underfitting procedure:
  - training error: large (≈ 0.5)
  - the difference of test error and training error: small (≈ 0)
  - test error: large (0.5)

Balanced fit

We pick the class of functions

$$f(x) = \mathrm{sign}(x - \theta)$$

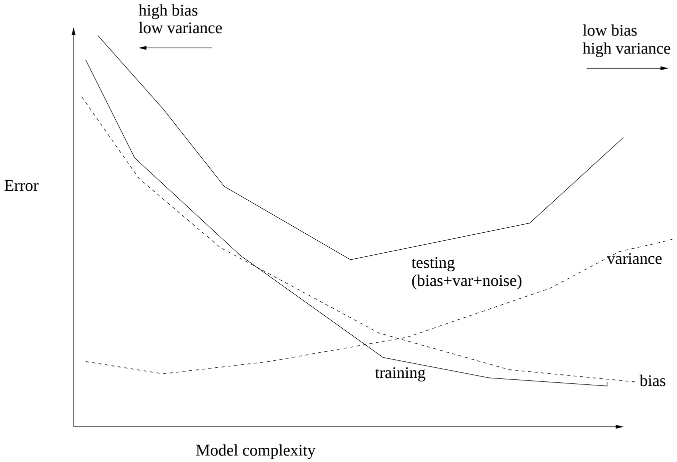
with unknown θ; then find θ to minimize training error.
Fits training data well with good prediction capability.

- small number of parameters: one dimensional.
- some flexibility but not overly complex.
- Characteristics of balanced fitting
    - training error: small (0)
    - difference of test error and training error: small ($\approx$ 0)
    - test error: small ($\approx$ 0)

Regularization

- Model complexity
  - how complex a target function can be represented by the model family
  - more complexity model:
    - represent more complexity and less smooth functions
    - prone to overfitting
  - less complexity model:
    - represent less complexity and smoother functions
    - prone to underfitting
- Regularization: limit model complexity to achieve good test error
  - restrict parameter space to control model complexity
- Typical learning algorithms:
  - have tuning parameters to control model complexity
  - tuning parameters should be adjusted to achieve good balance.

Bias-variance trade-off



Linear Model

- Consider training data: $S_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$
- Linear prediction rule:
$$f(X) = w^T X.$$
- Linear Least Squares Method:
$$\hat{w} = \arg\min_w \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - Y_i)^2$$

Linear Model

- Consider training data: $S_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$
- Linear prediction rule:
$$f(X) = w^T X.$$
- Linear Least Squares Method:
$$\hat{w} = \arg\min_w \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - Y_i)^2$$
- Complexity:
  - dimension d: the large d is the more complex the model is

Generalization and Model Complexity

- Training error:

$$\text{training error}(f) = \frac{1}{n}\sum_{i=1}^{n}(f(X_i) - Y_i)^2.$$

- Test error:

$$\text{test error}(w) = \mathbf{E}_{(x,y)}(f(x) - y)^2$$

- Learning algorithm: find $\hat{w}$ to minimize training error
- Model complexity: measuring difference of training and test error
- Generalization error: with large probability we have

$$\text{test error}(\hat{w}) \leq \text{training error}(\hat{w}) + \underbrace{O(d/n)}_{\text{difference of training test}}$$

  - only useful when $d \ll n$
  - what to do when d is large? need other form of regularization

Linear Regularization

- Regularization: restrict the model expressiveness when d is large.
- Restrict the linear family size: $g(w) \le A$.
  - example: $g(w) = \|w\|_2^2 = \sum_{j=1}^{d} w_j^2$.
  - A: tuning parameter (estimated through cross-validation).
- Model complexity: measured by A.
- Benefit of regularization:
  - statistical: robust to large number of features.
  - numerical: stabilize solution.

Linear Regularization Formulation

- Goal: minimize average squared loss on unseen data.
- A practical method: minimize observed loss on training:

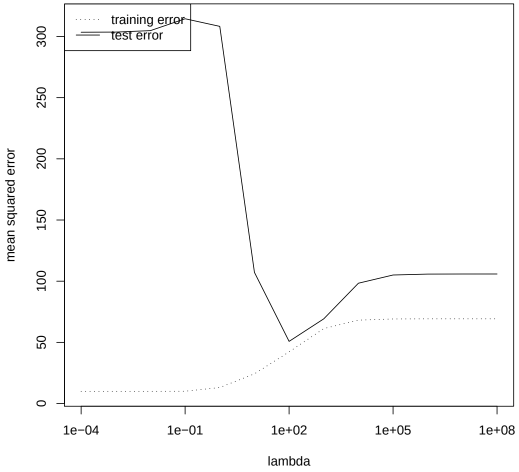
$$\hat{w} = \arg\min_{w} \frac{1}{n}\sum_{i=1}^{n}(w^T X_i - Y_i)^2,$$
$$\text{such that} \quad g(w) \leq A.$$

- Equivalent formulation ($\lambda \geq 0$):

$$\hat{w} = \arg\min_{w} \frac{1}{n}\sum_{i=1}^{n}(w^T X_i - Y_i)^2 + \lambda g(w)$$

- Small A corresponds to larger λ and vice versus
- Complexity: A large complexity large; λ small complexity large

Training and Test Error versus λ



model complexity decreases as λ increases

Model Complexity: generalization bound

- Least squares with square regularization:

$$\hat{w} = \arg\min_w \frac{1}{n} \sum_i (w^T X_i - Y_i)^2 + \lambda w^2.$$

- Predictive power of $\hat{w}$:

$$\text{test error} \leq \left(1 + \frac{1}{\lambda n} \max_x \|x\|_2^2\right)^2 \underbrace{\min_w E_{x,y} \left((w^T x - y)^2 + \lambda \|w\|_2^2\right)}_{\text{best possible regularized loss}}$$

- Robust to large feature set:
  - square reg $g(w) = \frac{1}{2} w^2$
  - generalization depends on $\|x\|_2^2/\lambda$, not on dimension d

L_p Regularization

- L_p-regularization: (constrained version)

$$\hat{w}^{L_p} = \arg\min_{w \in R^d} \sum_{i=1}^{n} (w^T X_i - Y_i)^2 \text{ subject to } \sum_{j=1}^{d} |w_j|^p \leq A.$$

- Equivalent formulation: $\lambda \geq 0$ is regularization parameter (penalized version)

$$\hat{w}^{L_p} = \arg\min_{w \in R^d} \sum_{i=1}^{n} (w^T X_i - Y_i)^2 + \lambda \sum_{j=1}^{d} |w_j|^p.$$

- $p = 0$: sparsity; $p = 1$:Lasso; $p = 2$: ridge regression.
  - subset selection: $p = 0$ (non-convex, non-smooth)
  - $p \in (0, 1)$ (non-convex, smooth)
  - $p \geq 1$: convex ($p = 1$ is the closest convex approximation to $p = 0$)
  - $p = 2$: ridge regression; $p = 1$: Lasso

Ridge regression

Regularization formulation

$$\hat{w}^{L2} = \arg\min_{w \in R^d} \sum_{i=1}^{n} (w^T X_i - Y_i)^2 + \lambda \sum_{j=1}^{d} |w_j|^2.$$

- Implicit dimension reduction effect (principal components).
- More stable (smaller weights) than least squares
- It does not generally lead to sparse solution.
- Closely related to kernel methods.

Ridge regression solution

- X: $n \times d$ data matrix (n training data and d features)
- Solution of ridge regression:

$$\hat{w}^{L2} = (X^T X + \lambda I)^{-1} X^T Y.$$

- Compare to standard least squares regression solution:

$$\hat{w} = (X^T X)^{-1} X^T Y.$$

  disadvantage: $X^T X$ may not be invertible
- Advantage: ridge regression allows $d > n$
  - stable: $X^T X + \lambda I$ is always invertible
  - Implicit dimension reduction effect.

The Effect of Correlated Feature

- Boston Housing data: 13 variables ($X[\cdot, 1 \cdots 13]$)
- Adding another feature: $X[\cdot, 14] = X[\cdot, 13] +$ random-noise
  - smaller noise: $X[\cdot, 13]$ is more correlated to $X[\cdot, 14]$
- weight $|w_{13}|$ under various noise size

noise size	0.01	0.1	0.5	1
least squares	358	25.5	1.77	14.7
ridge ($\lambda = 1$)	3.90	3.99	3.39	7.31

- least squares is not stable when there are correlated features

Lasso

- Find w with 1-norm $\leq s$ to minimize squared error:

$$\hat{w}^{L1} = \arg\min_{w \in R^d} \sum_{i=1}^{n} (w^T X_i - Y_i)^2$$

$$\text{subject to } \sum_{j=1}^{d} |w_j| \leq s.$$

- Equivalent to ($\lambda \geq 0$ is regularization parameter):

$$\hat{w}^{L1} = \arg\min_{w \in R^d} \sum_{i=1}^{n} (w^T X_i - Y_i)^2 + \lambda \sum_{j=1}^{d} |w_j|.$$

Solution property

- Convex optimization problem, but solution may not be unique.
- Global solution can be efficiently found.
- Solution is sparse: some w_j will be zero: achieves feature selection.
- Solution is not necessarily stable.

Summary

- Training versus test:
  - find prediction rule to best fit training data
  - performance evaluated on test data
- Model complexity: difference between training and test error
  - increase model complexity decreases training error but increases difference between training and test
- Regularization:
  - control model complexity by restricting parameter space
- Linear regularization: restrict weight using $\|w\|_p \leq A$
  - $p = 2$: ridge regression — stable solution and allow $d \geq n$
  - $p = 1$: Lasso (convex) — sparse solution but may be unstable.