Regression

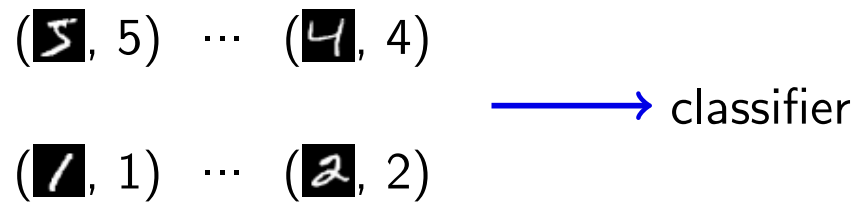
Nan Ye

School of Mathematics and Physics
The University of Queensland

Learning Problems

Supervised Learning

- Fit a model relating x and y given a dataset $(x_1, y_1), \ldots, (x_n, y_n)$
- In a classification problem, the output is discrete.



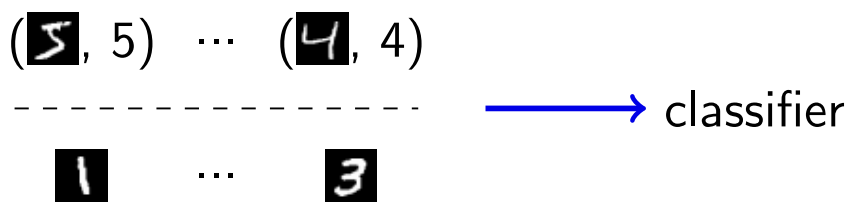
- In a regression problem, the output is real-valued.

Terminology

- *x: input, independent variables, covariate vector, observation, predictors, explanatory variables, features.*
- *y: output, dependent variable, response.*

Semi-supervised learning

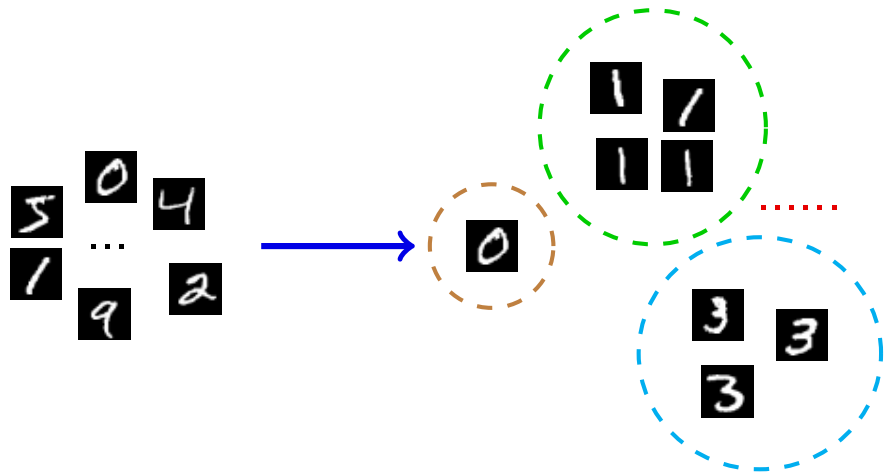
- Same as supervised learning, except that the dataset additionally contain unlabeled inputs $x'_1, x'_2, \ldots, x'_m$.



- Useful when it is expensive to label the inputs.

Unsupervised Learning

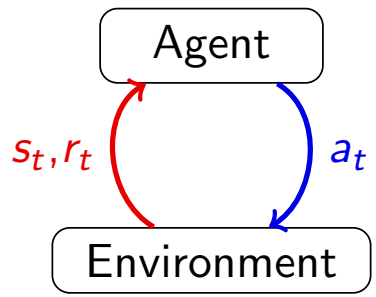
- Only the inputs are given, but not the outputs
- Unsupervised learning methods are used for various purposes, e.g.
  - Clustering: divide data points into groups



  - Density estimation: estimate a distribution given a sample
  - Dimension reduction: find a low-dimensional representation of data

Reinforcement learning

- In reinforcement learning, the agent learns how to act in an unknown environment by interacting with the environment.
- At time t, the agent executes an action a_t, and the environment provides its state s_t and a reward r_t as the feedback.



- The goal is to learn a policy (mapping from state to action) that maximizes the expected rewards.
- Reinforcement learning is hard because the feedback is limited and rewards may be delayed.

The Machine Learning Approach

- Formulating and solving a problem as a machine learning problem
- Example: learning a Bernoulli distribution
- Example: learning a Gaussian distribution

How A Learning Algorithm Works

- Given some training data, we choose a model class.
- We then choose a model that fits the training data well according to some measure.
  - Usually, the model has certain number of parameters, and choosing the values of these parameters can be cast as a numerical optimization problem.
- Once we have a trained model, we can use it to make predictions on new data.

Learning a Bernoulli Distribution

I pick a coin with the probability of heads being θ. I flip it 100 times for you and you see a dataset D of 70 heads and 30 tails, can you learn θ?

Maximum likelihood estimation

The likelihood of θ is

$$P(D \mid \theta) = \theta^{70}(1-\theta)^{30}.$$

Learning θ amounts to maximizing the likelihood.

$$\begin{aligned}
\theta_{ml} &= \operatorname*{argmax}_{\theta} \; P(D \mid \theta) \\
&= \operatorname*{argmax}_{\theta} \; \ln P(D \mid \theta) \\
&= \operatorname*{argmax}_{\theta} \; (70 \ln \theta + 30 \ln(1-\theta)).
\end{aligned}$$

Note that we have switched to log-likelihood, which is typically easier to work with.

$$\theta_{ml} = \operatorname*{argmax}_{\theta} \left(70 \ln \theta + 30 \ln(1-\theta)\right).$$

Set derivative of log-likelihood to 0,

$$\frac{70}{\theta} - \frac{30}{1-\theta} = 0,$$

we have

$$\theta_{ml} = 70/(70+30).$$

Regularization

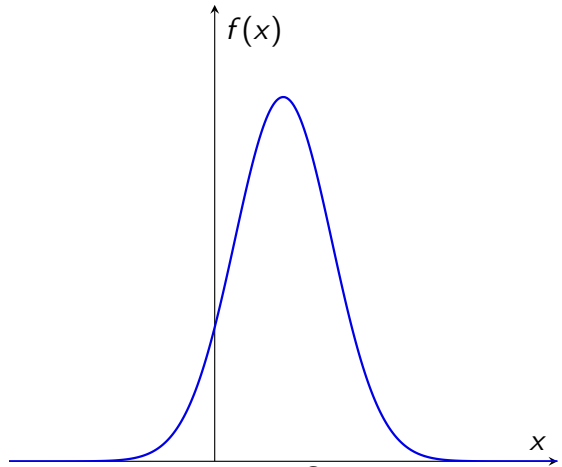
If your friend told you that he has played the game with me and his estimate is 0.6, and you want to make use of this prior knowledge...

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \left(70 \ln \theta + 30 \ln(1-\theta) - \overbrace{\lambda(\theta - 0.6)^2}^{\text{regularizer}} \right)$$

- A regularizer is a term used to prevent fitting to irregularities in data.
- Here the regularizer tries to rely on your friend's estimate to combat irregularities.
- A larger $\lambda > 0$ means you have more trust for your friend's estimate.

Learning a Gaussian distribution

I pick a Gaussian $N(\mu, \sigma^2)$ and give you a bunch of data $D = \{x_1, \ldots, x_n\}$ independently drawn from it. Can you learn μ and σ.



The probability density function (PDF) of $N(\mu, \sigma^2)$ is

$$f(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

Maximum likelihood estimation

$$\begin{aligned}\ln f(D \mid \mu, \sigma) &= \ln \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \exp \left(- \sum_i \frac{(x_i - \mu)^2}{2\sigma^2} \right) \\ &= -n \ln(\sigma\sqrt{2\pi}) - \sum_i \frac{(x_i - \mu)^2}{2\sigma^2}.\end{aligned}$$

Set derivative w.r.t. μ to 0,

$$\sum_i \frac{x_i - \mu}{\sigma^2} = 0 \quad \Rightarrow \quad \mu_{ml} = \frac{1}{n} \sum_i x_i$$

Set derivative w.r.t. σ to 0,

$$-\frac{n}{\sigma} + \frac{(x_i - \mu)^2}{\sigma^3} = 0 \quad \Rightarrow \quad \sigma^2_{ml} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu_{ml})^2.$$

Regularization

If your friend told you that he has played the game with me and he knows σ and his estimate of μ is c, and you want to make use of this prior knowledge...

$$\hat{\mu} = \operatorname*{argmax}_{\mu} \left(\ln f(D \mid \mu) - \overbrace{\frac{1}{2\sigma^2}(\mu - c)^2}^{\text{regularizer}} \right) = \frac{1}{n+1}(c + \sum_i x_i).$$

- The regularized estimate $\hat{\mu}$ has a smaller variance than μ_{ml}.
- However, $\hat{\mu}$ has a larger bias (expected difference between the estimate and the true mean) than μ_{ml}.

Summing Up...

Learning is...

- Collect some training data, e.g. coin flips.
- Choose a hypothesis class, e.g. Bernouli distribution.
- Choose a loss function, e.g. negative log-likelihood.
- Choose an optimization procedure, e.g. set derivative to 0.

Remarks

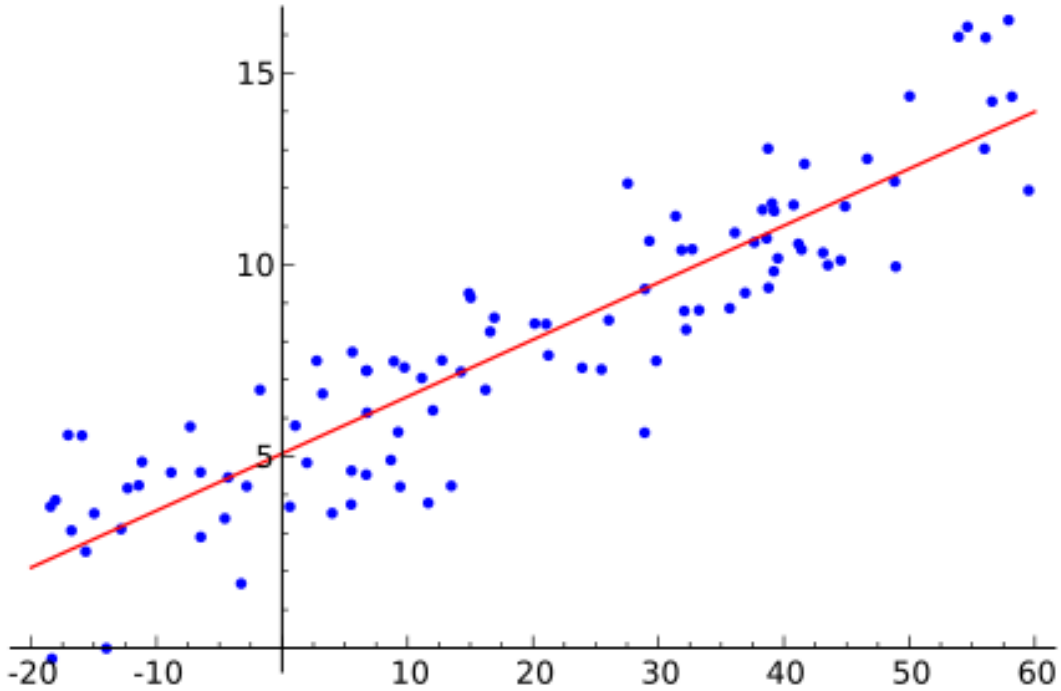
- Just like human learning, we want to learn something generally applicable beyond what we have seen.
  - *i.e., the model should work well not only on the training data, but on new test data too*
- Regularization may be used to encode prior knowledge.

Statistics, optimization and regularization provide powerful tools for formulating and solving machine learning problems.

Regression

- Ordinary least squares
- Ridge regression
- Basis function method
- Nearest neighbor regression
- Kernel regression

Ordinary Least Squares



Find a best fitting hyperplane for $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathbf{R}^d \times \mathbf{R}$.

- OLS finds a hyperplane minimizing the sum of squared errors (SSE)

$$\beta_n = \operatorname*{argmin}_{\beta \in \mathbf{R}^d} \sum_{i=1}^{n} (\mathbf{x}_i^\top \beta - y_i)^2.$$

- The solution to OLS is

$$\beta_n = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y},$$

  where $\mathbf{X}$ is the $n \times d$ matrix with $\mathbf{x}_i$ as the i-th row, and $\mathbf{y} = (y_1, \ldots, y_n)^\top$.

The formula holds when $\mathbf{X}^\top \mathbf{X}$ is non-singular (also see slide 20). When $\mathbf{X}^\top \mathbf{X}$ is singular, there are infinitely many possible values for β_n. They can be obtained by solving the linear systems $(\mathbf{X}^\top \mathbf{X})\beta = \mathbf{X}^\top \mathbf{y}$.

Proof. The sum of squared error can be written as

$$R_n(\beta) = \sum_{i=1}^{n} (\mathbf{x}_i^\top \beta - y_i)^2 = ||\mathbf{X}\beta - \mathbf{y}||_2^2.$$

Set the gradient of R_n to 0

$$\nabla R_n = 2\mathbf{X}^\top (\mathbf{X}\beta - \mathbf{y}) = 0,$$

we have

$$\beta_n = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

Least Squares as MLE

- Consider the class of conditional distributions $\{p_\beta(Y|X) : \beta \in \mathbf{R}^d\}$, where

$$p_\beta(Y \mid X = \mathbf{x}) = N(Y; \mathbf{x}^\top \beta, \sigma^2) \doteq \frac{1}{\sqrt{2\pi}\sigma} e^{-(Y - \mathbf{x}^\top \beta)^2/2\sigma^2},$$

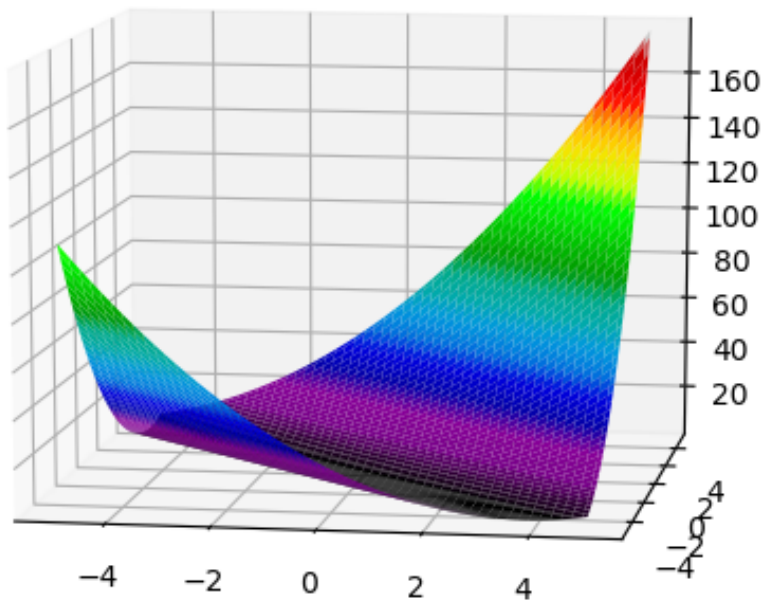
  with σ being a constant.
- The (conditional) likelihood of β is

$$L_n(\beta) = p_\beta(y_1 \mid \mathbf{x}_1) \dots p_\beta(y_n \mid \mathbf{x}_n).$$

- Maximizing the likelihood $L_n(\beta)$ gives the same β_n as given by the method of least squares.

Ridge Regression

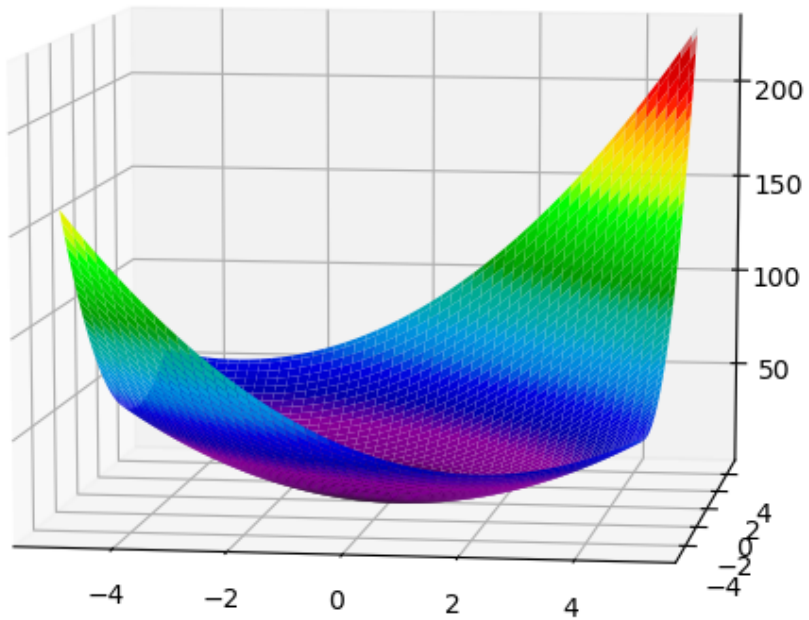
- When collinearity is present, the matrix $\mathbf{X}^\top\mathbf{X}$ may be singular or close to singular, making the solution unreliable.
- We see a valley if we plot the SSE, or a "ridge" in the log-likelihood (recall the MLE interpretation for least squares)



- Ridge regression fixes the "ridge" by adding a *regularizer* $\lambda||\beta||_2^2$ to OLS objective, where $\lambda > 0$ is a fixed constant.

$$\beta_n = \underset{\beta \in \mathbf{R}^d}{\operatorname{argmin}} \left(\sum_{i=1}^{n} (\mathbf{x}_i^\top \beta - y_i)^2 + \overbrace{\lambda||\beta||_2^2}^{\text{quadratic}/\ell_2 \text{ regularizer}} \right).$$

- If we plot the regularized SSE, we see a unique minimizer



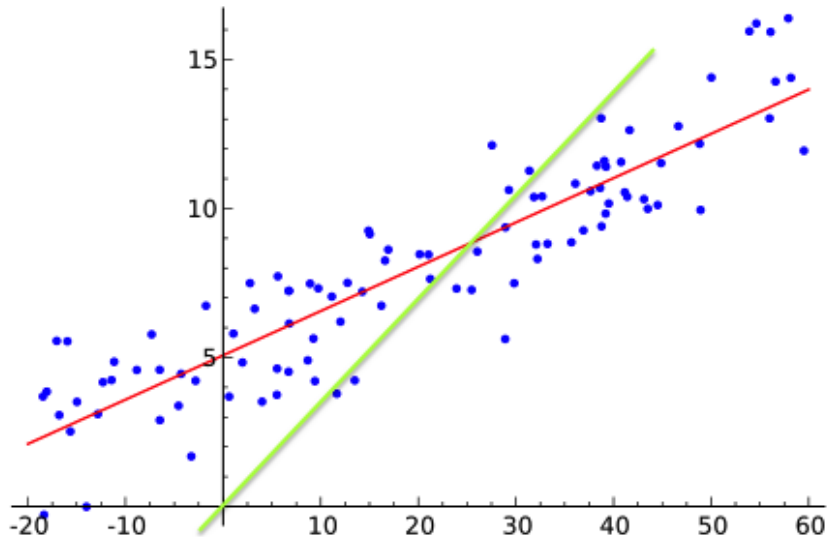
- By setting the gradient of the objective function to 0, we obtain

$$\beta_n = (\lambda I + \mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

  The matrix $\lambda I + \mathbf{X}^\top \mathbf{X}$ is non-singular, and thus there is always a unique solution.
- When λ is large, the model fits less well on training data, thus the regularizer helps in preventing the model from fitting to irregularities in the training data.

Regression with a Bias

- So far we have only considered hyperplanes of the form $y = \mathbf{x}^\top \beta$, which passes through the origin (green line).



- Usually, hyperplanes with a bias term (red line), that is, hyperplanes of the form $y = \mathbf{x}^\top \beta + b$ is more appropriate.
- The bias may need to be treated differently from the linear coefficients.

OLS with a bias term

- For OLS, we can simply choose the best fitting hyperplane with a bias term

$$(b_n, \beta_n) = \operatorname*{argmin}_{b \in \mathbf{R}, \beta \in \mathbf{R}^d} \sum_{i=1}^{n} (\mathbf{x}_i^\top \beta + \overbrace{b}^{\text{bias}} - y_i)^2.$$

- We can reduce it to regression without a bias term by adding a dummy feature: simply replace $\mathbf{x}^\top \beta + b$ with $\begin{pmatrix} 1 & \mathbf{x}^\top \end{pmatrix} \begin{pmatrix} b \\ \beta \end{pmatrix}$.

Ridge regression with a bias term

- For ridge regression, we again have a regularizer for the linear coefficients, but we *do not* regularize the bias term.

$$(b_n, \beta_n) = \underset{b \in \mathbf{R}, \beta \in \mathbf{R}^d}{\operatorname{argmin}} \left(\sum_{i=1}^{n} (\mathbf{x}_i^\top \beta + b - y_i)^2 + \lambda ||\beta||_2^2 \right).$$

- Again, we can solve it by reducing it to ridge regression without a bias term.
  - Let $\hat{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$, and $\hat{y}_i = y_i - \bar{y}$, where $\bar{\mathbf{x}} = \sum_{i=1}^{n} \mathbf{x}_i / n$, and $\bar{y} = \sum_{i=1}^{n} y_i / n$.
  - Then we can show that

$$\beta_n = \underset{\beta \in \mathbf{R}^d}{\operatorname{argmin}} \left(\sum_{i=1}^{n} (\hat{\mathbf{x}}_i^\top \beta - \hat{y}_i)^2 + \lambda ||\beta||_2^2 \right),$$

$$b_n = \bar{y} - \bar{\mathbf{x}}^\top \beta_n.$$

Basis Function Method

We can use linear regression to learn nonlinear functions

- Choose some *basis functions* $g_1, \ldots, g_k : \mathbf{R}^d \to \mathbf{R}$.
- Transform each input $\mathbf{x}$ to $(g_1(\mathbf{x}), \ldots, g_k(\mathbf{x}))$.
- Perform linear regression on the transformed data.

Examples

- Linear regression: use basis functions $g_1, \ldots, g_d$ with $g_i(\mathbf{x}) = x_i$, and $g_0(\mathbf{x}) = 1$.
- Quadratic functions: use basis functions of the above form, together with basis functions of the form $g_{ij}(\mathbf{x}) = x_i x_j$ for all $1 \leq i \leq j \leq d$.

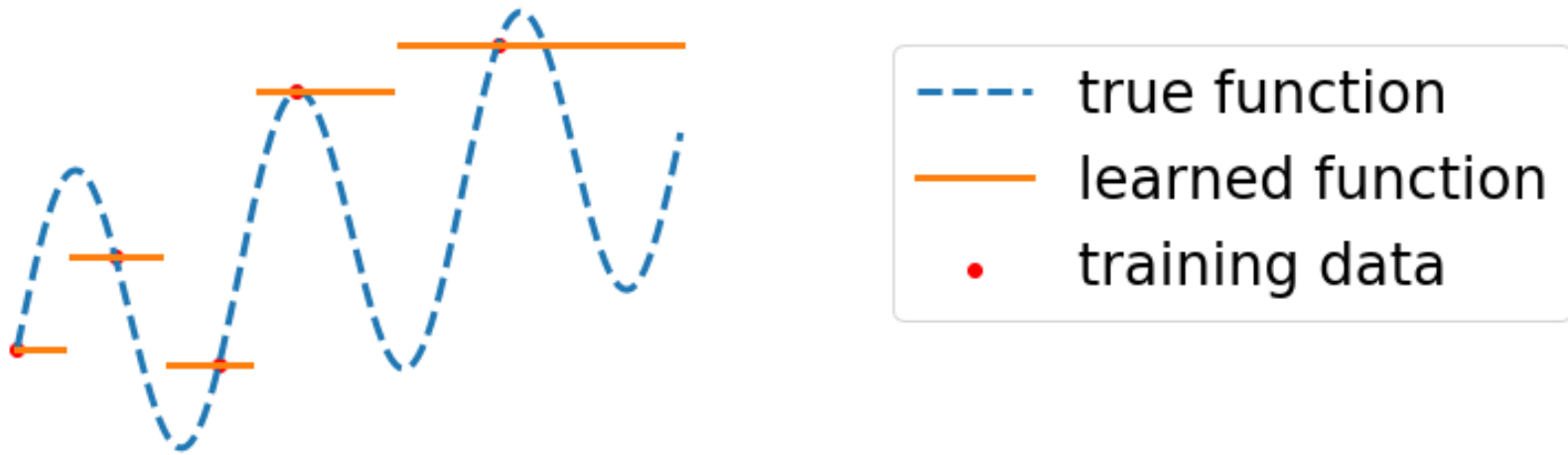
k nearest neighbor (kNN)

Given a test example $\mathbf{x}$, kNN predicts

$$h_n(\mathbf{x}) = \text{avg}\{y_i : \mathbf{x}_i \in N_k(\mathbf{x})\},$$

that is, the average of the values of the set $N_k(\mathbf{x})$ of the k nearest neighbors of $\mathbf{x}$ in the training data.

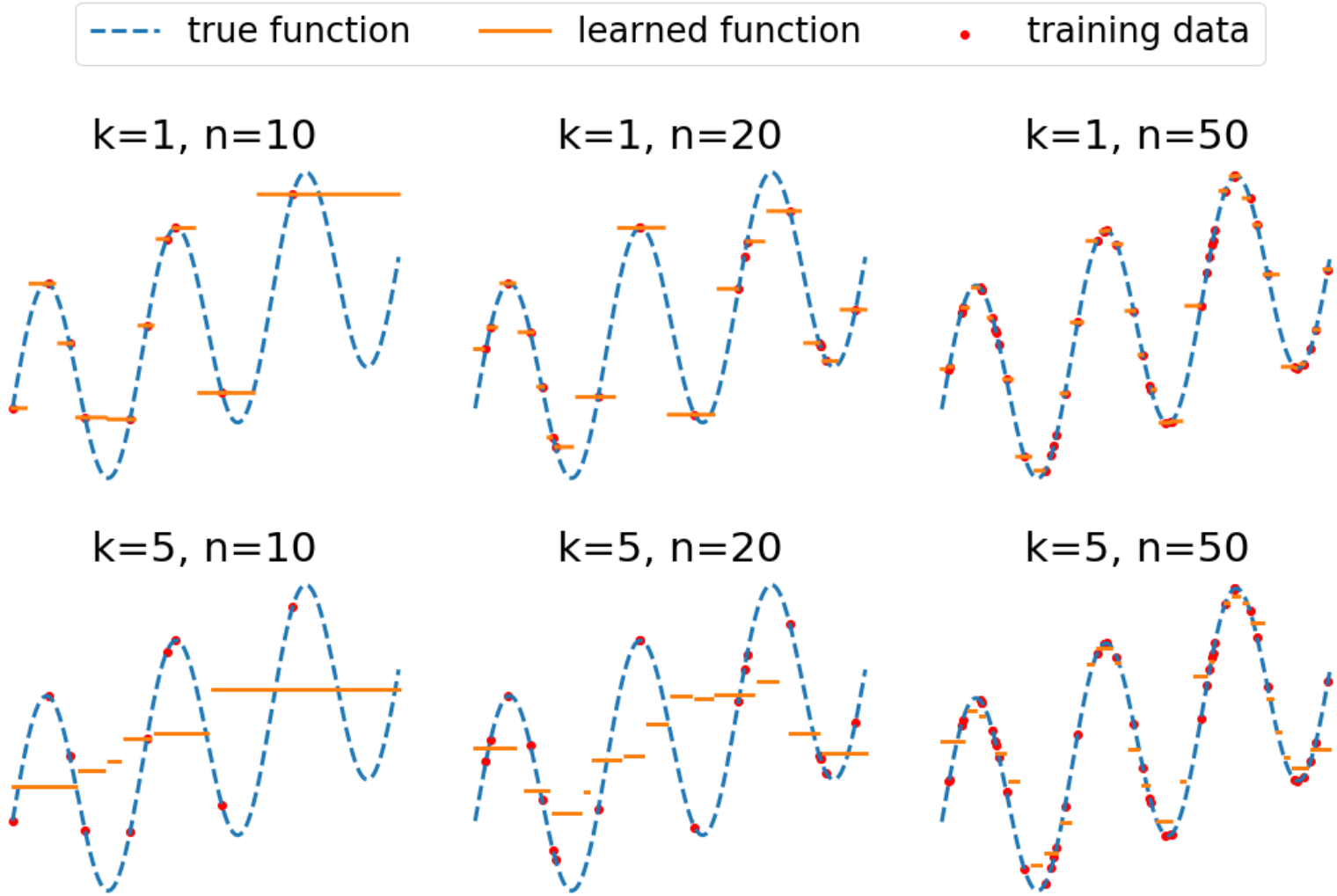
- (Curse of dimensionality) The number of samples required for accurate approximation is exponential in the dimension.
- kNN is a *non-parametric* method, while linear regression is a *parametric* method.



- In the figure above, we fit 1-NN regressor given five training points.
- The learned function is a piecewise-linear function passing through all the training points.

Your turn

- Given three training examples with (x, y) being $(1, 1)$, $(2, 2)$, $(4, 4)$, draw the function learned by 1NN.
- How does the function change if an additional training example $(5, 5)$ is provided?
- How does the function change if an additional training example $(3, 3)$ is provided?


true function
learned function
training data
k=1, n=10
k=1, n=20
k=1, n=50
k=5, n=10
k=5, n=20
k=5, n=50

Kernel Regression

Given a test example $\mathbf{x}$, kernel regression predicts

$$h_n(\mathbf{x}) = \sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x}_i) y_i \Big/ \sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x}_i),$$

where $K(\mathbf{x}, \mathbf{x}')$ is a function measuring the similarity between $\mathbf{x}$ and $\mathbf{x}'$, and is often called a kernel function.

Example kernel functions

- Gaussian kernel $K_\lambda(\mathbf{x}, \mathbf{x}') = \frac{1}{\lambda} \exp(-\frac{||\mathbf{x}' - \mathbf{x}||_2^2}{2\lambda})$.
- kNN kernel $K_k(\mathbf{x}, \mathbf{x}') = I(||\mathbf{x}' - \mathbf{x}|| \leq \max_{\mathbf{x}'' \in N_k(\mathbf{x})} ||\mathbf{x}'' - \mathbf{x}||)$. Note that this kernel is data-dependent and non-symmetric.

Parametric vs. Nonparametric Methods

- Parametric methods use models with fixed number of parameters
  - e.g. OLS, ridge regression
- Nonparametric methods use data-dependent models with a increasing number of parameters as more data becomes available.
  - e.g. kNN, kernel regression
- Nonparametric methods can possibly represent more complex functions, but they often requires a lot of data and computation.

Your turn

Which of the following statement is correct? (Multiple choice)

(a) In ridge regression, we should penalize a large bias term.

(b) Ridge regression always has a unique solution.

(c) A non-parametric regression method has no parameters.

(d) In machine learning, model learning is often formulated as an optimization problem.

What You Need to Know...

- Types of learning problems
- The machine learning approach
- Parametric regression: OLS, ridge regression, basis function method.
- Non-parametric regression: kNN, kernel regression.