Introduction to Machine Learning

Lecture 2

GuangBing Yang, PhD

yguangbing@gmail.com, Guang.B@chula.ac.th January 29th, 2021

Supervised Learning 1: Linear Models for Regression

What is regression problem

- It is a kind of supervised learning problem.
- It learns a mapping from inputs X to output Y, given a labeled set of pairs—**training sets.** $D = \{(x_i, y_i)\}_{i=1}^{N}$.
- Y is real-valued scalar or continuous, like 0.1234, 123, etc.
- Simply say, learning a continuous function is called **regression**.
- Based on above, this type of supervised learning problem is known as the regression.
- The goal of regression is:
  - To predict the value of one or more continuous output (sometimes also called targets or responsible variable) variables Y given the value of input variables X.

yguangbing@gmail.com, Guang.B@chula.ac.th January 29th, 2021

What does the linear mean here?

- Given the simplest linear regression model:
  - $y = f(\mathrm{x}, \mathrm{w}) = w_0 + w_1 x_1 + \ldots + w_D x_D$,

  where **w** are parameters, and **x** are input variables in D-dimensions, $\mathrm{x} = (x_1, \ldots, x_D)^T$.
- The 'linear' is regard to **parameters w** rather than inputs X.
- The above linear regression model is the simplest one because the both functions of parameters and inputs are linear.
- The linearity of input variables imposes significant limitations on the model.
- However, the functions of inputs x can be (normally it must be) nonlinear ones.

yguangbing@gmail.com, Guang.B@chula.ac.th January 29th, 2021

What does the linear mean here?

- That means the input variables **x** can be some nonlinear function's outputs.
- Use a function, $\Phi(\mathrm{x})$, to replace the **x**, where $\Phi(\,.\,)$ is called **basis function**.
  - For example, in the above the simplest linear regression model, the basis function is $\Phi(\mathrm{x}) = \mathrm{x}$.
- Now, the linear regression model can be represented as:
  - $y = f(\mathrm{x}, \mathrm{w}) = w_0 + w_1\Phi(x_1) + \ldots + w_D\Phi(x_D)$, Or,
  - $y = f(\mathrm{x}, \mathrm{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathrm{x})$,
  - where $\phi_j(\mathrm{x})$ are known as **basis functions** corresponding to the jth parameter w_j. Note that $\phi_i(\mathrm{x})$ and $\phi_j(\mathrm{x})$ are not necessary to be the same function to inputs X.
  - In realistic machine learning problem, these basis functions are often defined as feature functions.

yguangbing@gmail.com, Guang.B@chula.ac.th January 29th, 2021

What is regression problem

- The parameter w_0 is a fixed offset in the data and is called "**intercept**" in statistics and "**bias parameter**" in machine learning.
- For convenient, let's define an additional ***dummy basis function*** $\phi_0(\mathrm{x}) = 1$, so that
  - $y = f(\mathrm{x}, \mathrm{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathrm{x}) = \mathrm{w}^T \Phi(\mathrm{x})$,
  - where $\mathrm{w} = (w_0, \ldots, w_{M-1})^T$ and $\Phi = (\phi_0, \ldots, \phi_{M-1})^T$, and there are M parameters and M basis functions.
- Note that in realistic machine learning, pattern recognition, or actually any of AI related applications, the basis functions $\phi_j(\mathrm{x})$ can be expressed in some form of ***fixed pre-processing*** or ***feature extraction*** to the original data variables — this kind of processing is called ***feature engineering***.

What is regression problem

- Some examples of basis functions:
  - *Gaussian* basis function: $\phi_j(x) = exp\{-\frac{(x-\mu_j)^2}{2s^2}\}$
  - The *logistic sigmoid* basis function: $\phi_j(x) = \sigma(\frac{x-\mu_j}{s})$, where the sigma function is defined as: $\sigma(a) = \frac{1}{1+exp(-a)}$
  - The *hyperbolic tangent* function: $\phi_j(x) = tanh(x)$, where $tanh(a) = 2\sigma(2a) - 1$.

Linear Regression Model

- What can we do to know the parameters w and why does the approach work?
  - Remember that we say a supervised learning is about knowing some of x and y to find out a function f` to approximately represent the true function f to satisfy the relation: y=f(x)
  - Previously we defined: $y = f(\mathrm{x}, \mathrm{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathrm{x}) = \mathrm{w}^T \Phi(\mathrm{x})$.
  - One can see the only unknown matters in the *f* is the parameters W. Therefore, the problem of finding the function *f* transfers to the problems of getting the values of W.
- How can we get the values of W? The answer is the cost function, also called error function or loss function.

Linear Regression Model — Cost Function

- The **cost function** is defined as the difference between expected values and actual values of outputs.
- The most common and the basis cost function in machine learning is the **least squares error** function, also called the **sum-of-squares error** functions:

$$E_D(\mathrm{w}) = ||y - \hat{y}||^2$$

or,

$$E_D(\mathrm{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y_n - \mathrm{w}^T \Phi(\mathrm{x}_n)\}^2$$

which substituted $\hat{y}$ with $\mathrm{w}^T \Phi(\mathrm{x})$, and brought in the summation processing of all training data points

Linear Regression Model — Cost Function

- Now, having the cost function, how to get the parameters W's values?
- Solution: minimize the cost function, get expected parameters W. By setting the gradient of the $E_D(\mathrm{w})$ over W to zero gives the minimization of the sum-of-squares error function.

$$0 = \frac{\partial E_D(\mathrm{w})}{\partial \mathrm{w}} = \frac{2}{2}\sum_{n=1}^{N}\{y_n - \mathrm{w}^T\Phi(\mathrm{x}_n)\}\Phi(\mathrm{x}_n)^T$$

expanded it, we have: $\mathrm{w}^T(\sum_{n=1}^{N}\Phi(\mathrm{x}_n)\Phi(\mathrm{x}_n)^T) = \sum_{n=1}^{N} y_n\Phi(\mathrm{x}_n)^T,$

one more step,

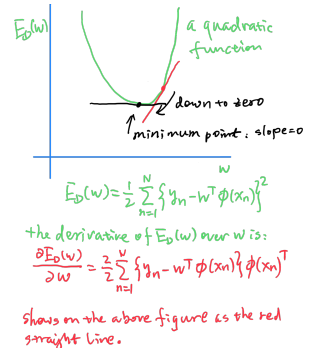
$$\mathrm{w} = \frac{\sum_{n=1}^{N} y_n\Phi(\mathrm{x}_n)^T}{(\sum_{n=1}^{N}\Phi(\mathrm{x}_n)\Phi(\mathrm{x}_n)^T)} = (\sum_{n=1}^{N}\Phi(\mathrm{x}_n)\Phi(\mathrm{x}_n)^T)^{-1}\sum_{n=1}^{N} y_n\Phi(\mathrm{x}_n)^T,$$

A compact expression: $\mathrm{w}_{ML} = (\Phi^T\Phi)^{-1}\Phi^T\mathrm{y}$, also called ***Normal Equations*** when $\Phi^T\Phi$ is not singular, the optimal W_{ML} is unique. Why?

Where Φ is called *design matrix* (also the basis function of x)

Linear Regression Model — Cost Function

- Why calculate the gradient of the cost function over the parameter? An optimization problem.
- The sum-of-square error function is actually a quadratic function.
- It has a convex curve, the minimum point can be found using the slope.
- when Tangent slope =o, the point is the minimum point, and corresponding w is W_{ML}.

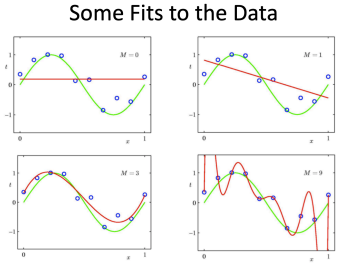


Over-fitting and under-fitting problems

- Machine learning uses some approximation approaches to estimate the parameters.
- This approximation simplifies the learning processes, but
- It brings a significant problem—over-fitting and under-fitting, particular the over-fitting problem.
- The over-fitting is about the trained model *perfectly* matches the training data.
- In other words, the trained model has *memorized* the details of training data, even any noise signals in the training data.
- The consequence is that the trained model performs very poorly in predictions of new data, which the model never sees before.
- A serious over-fitting error can make the model lose the predicative capability totally.

Over-fitting and under-fitting problems

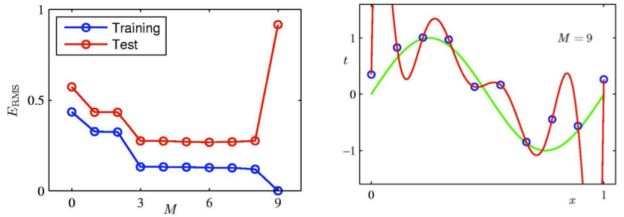
- Use the polynomial curve fitting from the text book, *Pattern Recognition and Machine Learning:*
-



For M=9, we have fitted the training data perfectly.

Over-fitting and under-fitting problems

- Testing the model on 100 data points, which were sampled using the same procedure used for generating the training data.
-

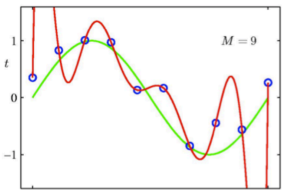


For M=9, the training error is zero! The parameters **w** can be fitted exactly to the data points. But the test error is huge, why?

Over-fitting and under-fitting problems

- As M increases, the magnitude of coefficient becomes larger.
- For M=9, the coefficients have become finely tuned to the data.
- Between data points, the function exhibits large oscillations.
- The consequence is that more flexible polynomials with larger M tune to the random noise on the target values.

	$M=0$	$M=1$	$M=3$	$M=9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43



Over-fitting and under-fitting problems

- For a complex model (having many parameters), more training data can make over-fitting problem less serious.
- For a few of training data, a complex model is highly likely over-fitted.
- So, solutions to overcome the over-fitting problem are:
  1. using more training data.
  2. simplify the model, but this is not work for most of cases since a complex problem needs a complex model.

Thus, a more general solution is needed: **Generalization/regularization**

Generalization/regularization

- The purpose of a good generalization is to let the model to make accurate predictions for new test data that is not known during learning.
- The cost function with regularization takes the form:

$$\hat{E_D} = E_D(\mathrm{w}) + \lambda E_W(\mathrm{w})$$

where λ is the *regularization coefficient* that controls the relative importance of the **data-dependent error** $E_D(\mathbf{w})$ and the regularization term $E_W(\mathrm{w})$.

- In the case of the sum-of-square error function, the above formula can take this form:

$$\frac{1}{2}\sum_{n=1}^{N}\{y_n - \mathrm{w}^T\phi(\mathrm{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

Generalization/regularization

- In machine learning, when $q = 1$, we call the regularization as L1 or **lasso** regularization, if $q = 2$, it is called L2 or **weight decay** regularization. They are most common regularization methods.
- Thus, the error function with L1 regularization is:

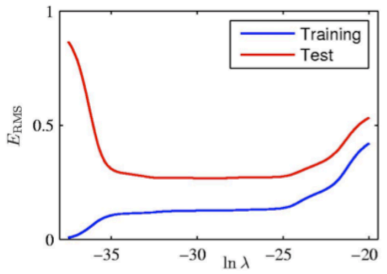
$$\frac{1}{2}\sum_{n=1}^{N}\{y_n - \mathrm{w}^T\phi(\mathrm{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|$$

- and the error function with L2 regularization is:

$$\frac{1}{2}\sum_{n=1}^{N}\{y_n - \mathrm{w}^T\phi(\mathrm{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^2$$

Generalization/regularization

- After applying the regularization, the training and testing cost functions of the polynomial curve fitting model basically match to each other.



	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Regression examples in real-world

- Predict tomorrow's stock market price given current market information.
- Predict tomorrow's temperature in Bangkok, or amount of rain will dump tomorrow before night.
- Predict how many people may be affected by Coronavirus till the end of this month, given current conditions.
- Predict when will the Covid-19 pandemics end if given the progress information of vaccine development and current situations of the pandemics.
- And more examples..

Build a Linear Regression Model

- To build a linear regression model, we need following components:
- The data, that will be split randomly into three portions: **training**, **validation**, and **testing** datasets.
  - The training dataset is used to train the model,
  - the validation dataset is used to tune the model or doing model selections, and
  - the testing dataset is only used to evaluate the model performance and predictive capability.
- Determine a basis function $\phi(x)$, either the simplest linear, or a non-linear basis function, such as a polynomial, sigmoid, tanh, or exponential functions as discussed previously.
- Define a cost function with regularization method: L1 or L2.
- Train the model and update the parameter values continuously until convergence or reaching the threshold of the minimization of the loss (error) or the end of the loop of the training.
- Validate and tune the model, compare the loss values obtained using validation data and training data. Make sure there are no extreme variances between these two loss values to avoid the over-fitting issues. By adjusting the regularization coefficient and regularization methods (L1 or L2) to minimize the over-fitting issues.
- Evaluate the model performance using testing data. Record the metrics and report the model performance.

Recap

- Linear regression is a type of supervised learning
- The training data consists of x and y, the labeled data, and output y is real-valued scalar or continuous
- The 'linear' is regard to **parameters w** rather than inputs X.
- The basis function of x can be nonlinear.
- The **cost function** is defined as the difference between expected values and actual values of outputs.
- The most common cost function is the **least squares error** function, also called the **sum-of-squares error**.
- The over-fitting is about the trained model *perfectly* matches the training data.
- Regularization is the approach to solve the over-fitting problem.
- To build a linear regression model needs training/testing data, cost function, regularization methods, and training/evaluation processing.

Q & A

Any questions?

Lets go to the lab section

