September 29, 2018 10:15 AM

Introduction to Machine Learning

A set of tools that allow us to teach computers by providing examples of how stuff should be done.

Ex/ program to filter spam?

ways to solve?

- flag terms such as Nigerian gold, viagara, and doctors hate this guy.

con: writing too many rules is hard to do. Scammers are dumb but smarter than before, as a result they could pass by filters by using words such as "Vi@gara".

ML has provided a solution as modern spam filters are "learned" from examples

ML is very diverse and there is multiple ways of defining it.

AI view: - build intelligence in computers by programming all rules cannot be done; automatic learning is crucial.

- Think of a baby: They don't know to speak or understand english but they learn it.

The Software Engineering View: - ML allows us to program computers by example which can be easier than writing code the traditional way.

The stats view: ML is a marriage of comp sci and stats. The application of computational techniques to statistical problem.

ML methods are broken down into two phases:

1. Training: A model is learned from a collection of training data.
2. Application: The model is used to make decisions about some new test data.

Ex / spam Filtering example

training data contains email messages labeled as "ham" or "spam", and each new email message that we receive (and wish to classify) is test data.

Main types of ML:

1) Supervised Learning: In which training data is labeled with the correct answers. The most common types of supervised learning is classification and regression
   - Classification: outputs are desirete labels
   - Regression: outputs are real valued

2) Unsupervised Learning: In which we are given a collection of unlabled data and we wish to analyze and discover patterns within. The two most important examples are dimension reduction and clustering

3) Reinforcement Learning: In which an agent (a robot or controller) seeks to learn optimal actions to take based on the out comes of past actions.

Other ones, not in the course include:

1. semi supervised learning: in which only a subset of the training data is labelled.

2. Active learning: in which obtaining data is expensive, so an algorithm must determine which training data to acquire.

3. Transfer learning:

4. Structured prediction:

5. Time series forecasting

6. Anomaly detection

7. Deep learning.

1.1 - Simple Regression problem

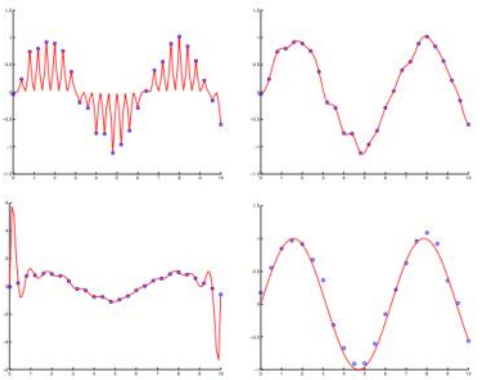


Figure 1: A simple regression problem. The blue circles are measurements (the training data), and the red curves are possible fits to the data. There is no one "right answer." The solution we prefer depends on the problem. Ideally we want to find a model that provides good predictions for new inputs (i.e., locations on the x-axis for which we had no training data). We will often prefer simple, smooth models like the one shown in the lower right.

Goal: To fit 1D curve to a few points

ML requires us to make certain choices to fit the data.

1) How do we parameterize the model we fit?
- quadratic
- linear
- sinusoidal curve

2) What criteria do we use to judge the quality of the fit?

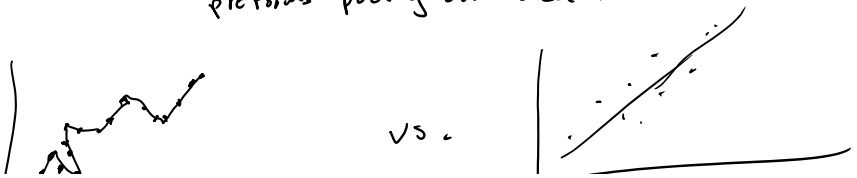
Ex/ Noisy data → then fit in terms of the squared error between the data we are given and the fitted curve.

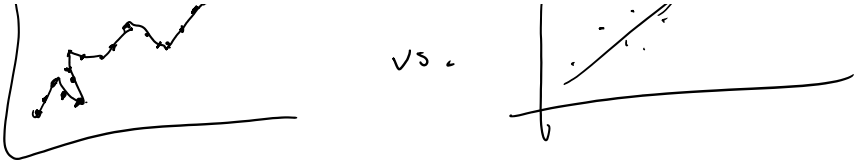
- when minimizing the squared error, the resulting fit is usually called a least-squared estimate.

3) Some types of models and some model parameters can be very expensive to optimize well. How long are we willing to wait for a solution, or can we use an approximation instead.

4) Overfitting vs underfitting. We care mainly about how well it works on future data.

overfit: when a model fits well but preforms poorly on test data




vs.

Linear Regression

In regression our goal is to learn a mapping from one real-valued space to another.

After learning, we can use the regression model to predict an output given some new test input.

1-D case

Our goal is to learn a mapping

$y = f(x)$, where x & y are both real-valued scalar. ($x \in \mathbb{R}, y \in \mathbb{R}$)

Let f be a linear function. Then

$y = \underset{\text{weight}}{w}x + \underset{\text{bias}}{b}$

weight and bias are scalar parameters which we would like to learn from the training data. In particular, we wish to estimate w, b from the N training pairs

$\{(x_i, y_i)\}_{i=1}^{N}$

Afterwards, once we have values for the weight and bias, we can compute or predict outputs for y given some input x.

Given two training data points

$\Rightarrow N = 2$

we can solve for the unknown slope/weight w and offset/bias b.

con: this approach is extremely sensitive to noise in the training data measurements

When $N > 2$ or the number of training data points are greater than 2, we will not be able to find unique parameter values for which $y_i = wx_i + b \; \forall i$, since we have more constraints than parameters.

The best we can hope is to find a line that is as close to the training points as possible.

The most common way to estimate the parameter of a line in a problem like this is by least squares regression.

Least Squares Regression

In least-squares regression, the quality of the fit between the model and the training data is specified in terms of the squared error / squared loss.

In more detail, for the i^{th} training point, given the model line with parameters w & b, to be the vertical distance from the line to the training point is

$e_i = y_i - (wx_i + b)$

The sum of squared errors over all train points is then ... of the fit is called

used to measure the quality ... of ... is

the objective function, energy function, empirical loss:

$$E(w,b) = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} (y_i - (wx_i+b))^2$$

By squaring the errors, we obtain a function that is ≥ 0, and zero iff all errors are $=0$.

To find a line that minimizes the squared error is equivalent to solving for the w and b that minimizes $E(w,b)$.

Take derivative of E with respect to the parameter

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial b} \sum_{i=1}^{N} (y_i - (wx_i+b))^2$$

$$= \sum_{i=1}^{N} \frac{\partial E}{\partial b} (y_i - (wx_i+b))^2$$

$$= \sum_{i=1}^{N} 2(y_i - (wx_i+b)) \cdot -1$$

$$= -2 \sum_{i=1}^{N} (y_i - (wx_i+b))$$

set it to zero

$$0 = -2 \sum_{i=1}^{N} (y_i - (wx_i+b))$$

$$0 = \left(\sum_{i=1}^{N} y_i - (wx_i+b) \right)$$

$$0 = \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} wx_i - \sum_{i=1}^{N} b$$

$$\sum_{i=1}^{N} b = \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} wx_i$$

$$b^* = \hat{y} - w\hat{x}$$

b^*: estimate; $\hat{y}$: average of y; $\hat{x}$: average of x

, this equation for b^* depends on w but we can substitute this estimate for b in the original energy function

$$\sum_{i=1}^{N} (y_i - (wx_i+b))$$

$$\sum_{i=1}^{N} (y_i - (wx_i + \hat{y} - w\hat{x}))^2$$

$$\sum_{i=1}^{N} (y_i - \hat{y} - wx_i + w\hat{x})$$

$$\sum_{i=1}^{N} ((y_i - \hat{y}) - (wx_i - w\hat{x}))^2$$

$$\sum_{i=1}^{N} ((y_i - \hat{y}) - w(x_i - \hat{x}))^2$$

Take partial derivative

$$\frac{\partial E}{\partial w} = \sum_{i=1}^{n} -2 \cdot \left(y_i - \hat{y} - w(x_i - \hat{x}) \right) \cdot (x_i - \hat{x})$$

$$= -2 \sum_{i=1}^{n} \left(y_i - \hat{y} - w(x_i - \hat{x}) \right)(x_i - \hat{x})$$

set it = 0

$$0 = -2 \sum_{i=1}^{n} \left(y_i - \hat{y} - w(x_i - \hat{x}) \right)(x_i - \hat{x})$$

$$0 = \sum_{i=1}^{n} \left((y_i - \hat{y})(x_i - \hat{x}) - w(x_i - \hat{x})^2 \right)$$

$$0 = \sum_{i=1}^{n} (y_i - \hat{y})(x_i - \hat{x}) - w \sum_{i=1}^{n} (x_i - \hat{x})^2$$

$$w \sum_{i=1}^{n} (x_i - \hat{x})^2 = \sum_{i=1}^{n} (y_i - \hat{y})(x_i - \hat{x})$$

$$w^* = \frac{\sum_{i=1}^{n} (y_i - \hat{y})(x_i - \hat{x})}{\sum_{i=1}^{n} (x_i - \hat{x})^2}$$

∴ the values of w^* and b^* are the least-squares estimates for the parameters of the linear regression.

2.2 Multi-dimensional inputs

Suppose we wish to learn a mapping from D-dimensional inputs $x \in \mathbb{R}^d$, $y \in \mathbb{R}$.

w : vector of weights

The mapping has the form

$$f(x) = w^T x + b = \sum_{j=1}^{D} w_j x_j + b$$

for convenience, we can fold the bias b into the weight vector, if we augment the input vector x with an additional 1. i.e if we define

$$\tilde{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_D \\ b \end{bmatrix}, \quad \tilde{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{bmatrix}$$

The error of a data point is defined to be the vertical distance from the point to the model line. Then the mapping can be written

$$f(x) = \tilde{w}^T \tilde{x}$$

Given N training input-output pairs, the least square objective function is:

$$E(\tilde{w}) = \sum_{i=1}^{N} (y_i - \tilde{w}^T \tilde{x}_i)^2$$

If we stack outputs in a vector and the inputs in a matrix, then we can also write this as

$$E(\tilde{w}) = \sum_{i=1}^{N} (y_i - \tilde{w}^T x_i)^2$$

$$= \| \quad \|^2$$

$$E(\tilde{w}) = \sum_{i=1}^{n} (y_i - w x_i)$$

$$= \sum_{i=1}^{n} \left(y_i - \underset{1\times n}{[w_1 \cdots w_0\ b]} \underset{n\times 1}{\begin{bmatrix} x_1 \\ \vdots \\ x_0 \\ 1 \end{bmatrix}}\right)^2$$

$$= \sum_{i=1}^{n} \left(y_i - [\tilde{x}_1\tilde{w}_1 + \tilde{x}_2\tilde{w}_2 \cdots \tilde{x}_0\tilde{w}_0 + b]\right)^2$$

$$= \sum_{i=1}^{n} (y_i - \tilde{x}_i w_i)^2$$

$$= \| y - \tilde{x}\tilde{w} \|^2$$

$$E(w) = \| y - \tilde{x}\tilde{w} \|^2$$

$$= (y - \tilde{x}\tilde{w})^T (y - \tilde{x}\tilde{w})$$

$$= (y^T - \tilde{w}^T\tilde{x}^T)(y - \tilde{x}\tilde{w})$$

how?

$$= y^T y - y^T\tilde{x}\tilde{w} - \tilde{w}^T\tilde{x}^T y + \tilde{w}^T\tilde{x}^T\tilde{x}\tilde{w}$$

$$= \tilde{w}^T\tilde{x}^T\tilde{x}\tilde{w} - 2y^T\tilde{x}\tilde{w} + y^T y$$

Optimize by setting $\frac{\partial E}{\partial w_i} = 0$

P $\frac{\partial}{\partial w_i}$.

Nonlinear Regression

Linear models are often insufficient to capture real world phenomena

=> The relationship between inputs and outputs we want to be able to predict is not linear.

As a result, non linear models are often required.

However, we are still interested in parameterized functions of the form.

$y = f(x)$

Recall

In linear regression, we used $f(x) = Wx + b$, the parameters of which were W & b. We estimate the parameters to fit the model data.

In the case of non linear regression $f(x)$ is a non linear function. In principle $f(x)$ could be anything.

The form that we will choose will have a major impact on the effectiveness of the regression.

A more general model will require more data to fit it, and different models are appropriate for different problems.

In many situation, we do not know much about the underlying nature of the process being modeled. In many others, modeling the modeling process precisely is too difficult.

In these cases, we typically turn to a few models in ML that are widely used and effective for many problems.

Includes:

- Basis function Regression
  - → Radial Basis Function Regression (RBF)

- Artificial Neural Networks
- k-Nearest Neighbor

Important choice to be made: objective function for learning / underlying noise

October 3, 2018 9:49 AM

Classification

In classification, we are trying to learn a map from an input space to some finite output space.

Simplest case: Binary Classification

determine whether an input has a property or not

eg. isSpam()
- yes?
- No?

eg. containsImage()
- True
- false

eg. hasDisease()
- True
- false

Multiclass classification

determining which of the multiple categories the input belongs.

Eg. recorded voice signal

gjucn