Machine Learning

Author: Pannenets.F

Date: September 2, 2020

Je reviendrai et je serai des millions. *«Spartacus»*

Introduction

Nothing.

Pannenets F September 2, 2020

Contents

Part I

part

Chapter 1 welcome

Definition 1.1 (Machine Learning) *Get computers to learn without being explicitly programmed.*

Neural network just mimics human's brain's working.

Applications:

- Database mining
- Application that human cannot do (too immediate, big)
- Self-customizing programs
- Understand human activities.

Chapter 2 What is Machine Learning?

2.1 Introduction

Definition 2.1 (Machine Learning) *Field of study that gives computers thr ability to learn without being explicitly programmed. (older, informal)*

Improve performance from task experience. (more modern)

- *A **task***
- *Some **experience***
- *Some way to **measure***

Two board type of ML algorithms:

- Supervised Learning
- Unsupervised Learning

Others:

- Reinforcement Learning
- Recommender Systems

2.1.1 Supervised Learning

Supervised learning has some known relationship between input and output.

Supervised learning problems are categorized into **regression** and **classification** problems.

Fitting a straight line or quadratic or 2nd-order curve to a function is a regression. The main idea is to give a continuous solution based on data we have.

Example 2.1 Estimate weekly income of a company.

Estimating some possbility to be certain type or classification is also a supervised learning.

Example 2.2 Classify tumor type based on its size.

When the scale of classes turns to infinity, we need **Support Vector Machine**.

2.1.2 Unsupervised Learning

Unsupervised learning has no specific relationship between input and output. Data has no lables, but the data could be devided into sevornal clusters.

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

Example 2.3 Two recordings of audio are mixed, and computer should get them departed.

2.2 Linear Regression with One Variable

2.2.1 Model and Cost Function

For a training set, Notation:

- m: number of training example
- x: input variable / features
- y: output variable / target variable

Learning algorithm applis to training set to find a **hypothesis** that could gives back a estimated answer.

The goal of supervised learing is to learn a function (hypothesis) from the training set.

2.2.2 Cost Function

In a univariate problems, we have:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

And the θ_i is so called **Parameters**.

Regression is to minimize the difference between $h(x)$ and y. The cost function defined as:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=0}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Of course, for each parameters's vector we can compute its J value. Then we can get the global minimum to get the best fit curve. Later, we will learn an efficient algorithm to find the minimum cost.

2.3 Parameter Learning

Have some function $J(\theta_0, \theta_1)$ and want to minimize it. Here is the outline:

- start with some θ_0, θ_1
- change the parameters until the loss has been minimized

We need a way to minimize loss function.

2.3.1 Gradient Descent

At a parameters' place, take a little baby step to change. And choose the step direction that has the quickest reduction speed.

And the process is to repeat the equantion until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1), \text{where} j = 0, 1$$

And α is called **learning rate** which controlls the step of learning. The partical or derivative part controlls the direction.

If α is too small, gradient descent can be really slow. But if it's too large, gradient descent may not converge or even diverge.

With this algorithm and appropriate learning rate, the parameters are always towards an (local) optimal valley.

2.3.2 Gradient Descent for Linaer Regression

With gradient descent, we can handle with the cost function provided in previous sections.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=0}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=0}^{m} \frac{\partial}{\partial \theta_i} h_\theta(x^{(i)}) \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

Informally, **convex function** has no local optimum but only one global optimum.

Batch gradient descent: each step takes all the training examples.

2.4 Linear Algebra Review

2.4.1 Matrices and Vectors

Definition 2.2 (Matrix) *Rectangular array of numbers, dimension of matrix is # of rows times # of columns. For a matrix A, its (i, j) entry is writen as A_{ij}.*

Definition 2.3 (Vector) *Vector can be treated as a $n \times 1$ matrix in $\mathbb{R}^n$.*

2.4.2 Addition and Scalar Multiplication

Matricx addition is to add elements at the same place of their matrix, of course, the matrices should have the same size. (element wise)

Scalar multiplication is to multiply the scalar number to every element in the matrix.

2.4.3 Matrix Vector Multiplication

For a matrix and a vector to be multiplied, they should respectively has size of $m \times n$, $n \times 1$. This relationship of size accords to that the answer is the linear combination of columns of matrix with coefficient of respected vector elements.

2.4.4 Matrix Matrix Multiplication

For 2 matrices to be multiplied, they should respectively has size of $m \times n$, $n \times p$.

2.4.5 Matrix Multiplication Properties

For matrix multiplication, it's a way to pack a series of hypothesis which has the same structure and different parameters.

2.4.6 Inverse and Transpose

Not all numbers have an inverse. So do matrices.

Only square matrices **may** have inverse. Else matrices could have pseudo inverse.

Words

quadratic

ellipse

contour

by convention

Chapter 3 Multivariate Linear Regression and Octave Tutorial

3.1 Multivariate Linear Regression

With **Multivariate Linear Regression**, we can do predictions with more infomation. It appears that the expression will get more inputs or features.

Note $x_j^{(i)}$ *refers to the ith training example's j_{th} feature value.*

For example, for a n-varible hypothesis, its math form should be like:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

Its vector form:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}, \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}, \text{where } x_0 = 1$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta^T$$

3.2 Gradient Descent for Multiple Varibles

For n-varible hypothesis, the cost function expresses like:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

So the gradient descent performs like:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}, \text{where } \theta = 0, 1, 2, \cdots, n$$

3.3 Gradient Descent in Practice

After the basic knowledge introduced, we are going to learn something in practice.

3.3.1 Feature Scaling

Make sure features are on a similar scale and the cost function can converge more quickly. If $x_1 \in (0, 2000)$ and $x_2 \in (1, 5)$, the step of gradient descent may fit with x_2 but be too small to quickly converge.

A typical and useful opration is to scale feature into $(0, 1)$. More generally, get every feature into approximately a $[-1, 1]$ range.

Or take mean normalization. Replace x_i with $x_i - \mu_i$ or $(x_i - \mu_i)/(\max(x) - \min(x))$ to make features have approximately zero mean.

3.3.2 Learning Rate

Make sure that gradient descent is working correctly, that is $J(\theta)$ should decrease after each iteration.

We can do automatic convergence test that if the decrease of $J(\theta)$ is less that a threshold, we declare the convergence.

If the α is too large, the loss may not converge or even diverge. If too small, it will take too long time to end the task.

3.4 Features and Polynomial Regression

Sometimes a straight line model would not fit a curve well, so that we choose polynomial regression to fit it with a polynomial.

For example, a 3-order polynomial like $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$, we can do mapping like

$$x \rightarrow x_1, x^2 \rightarrow x_2, x^3 \rightarrow x_3$$

Of course, in this example it's important to do scaling.

3.5 Computing Parameters Analytically

3.5.1 Normal Equation

Normal equantion is a method to solve for θ analytically.

Using calculus we can solve for the global optimal for equation (single or multiple varibles).

Or, we can display our data in matrix form, then use the least square method.

For features, use X; for outputs, use Y:

$$X = \begin{bmatrix} (x^{(0)})^T \\ (x^{(1)})^T \\ \vdots \\ (x^{(n)})^T \end{bmatrix} \in \mathbb{R}^{n+1}, Y = \begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

With least square method, get Θ like:

$$\Theta = (X^T X)^{-1} X^T Y$$

In Octave, it could compute by `pinv(x'*t)*x'*y`.

With normal equation, you do not need to choose a α and to iterate. BUT, when the dimesion is large, the computation $(X^T X)^{-1}$ could be very slow (it's a $O(n^3)$ algorithm). With gradient descent, it could work well even dimesion is large.

3.5.2 Normal Equation Noninvertibility

What will happen if $X^T X$ is not invertible? The pseudo inverse is used when

- redundant features / linearly dependent
- too many features

3.6 Octave Tutorial

Octave is a good language for algorithm prototyping.

3.6.1 Basic Operations

In fact, it's just so similar to MATLAB. For example, the `ones`, `zeros`, `eye`, `rand`, and the vector form.

```
1   w = rand(1, 10000) + (-6) % uniform distribution
2   hist(w, 100)
```

```
1   w = randn(1, 10000) + (-6) % normal distribution
2   hist(w, 100)
```

3.6.2 Moving Data Around

It's about how to move data into Octave for certain tasks.

For data file, use the `load` and `save`. For matrix, use the column and row index with ":", like `vec(col1:col2)` and matrix concat could be rather easy, like `c = [a b]`.

3.6.3 Computing on Data

For coresponding or elemenet-wise operations, there is a dot sign befor the opration, like `c = a .* b`, where a and b get the same size. Without the dot sign, it will be a matrix multiplication. And transpose is important to matrix which is bind to quatation mark, like `AT = A'`.

For most functions, they could be applied to vector or matrix form, like `abs([1,-2,3])`, `a = [1 3 5 -9]; a_b = a > 2`.

Some functions:

1. `magic`: return a n by n magic square matrix
2. `sum`: return sum of matrix
3. `prod`: return prod of matrix
4. `A(:)`: turn A into a vector

3.6.4 Plotting Data

In implying algorithm, plot is a very important tools. Just like in MATLAB, use `vectors`, `plot`, `figure`, `subplot`, `imagesc`. Some controll command like `colorbar`, `colormap`, `hold on`, `xlable`, `ylable`, `legend`, `title`,

3.6.5 Control Statements and Functions

In most programming languages, there are `for`, `while if` statements for controll.

Besides, we can define our own functions in seperate files and use them by `addpath` or `cd`

3.6.6 Vectorization

With vectorization, code can be more efficient and quick to imply.

For example, we can use vector transpose to compute the inner product rather than for-loop.

Words

prototyping

clunky

Chapter 4 Classification

4.1 Classification and Representation

4.1.1 Classification

Classification is to predict the varible y into discrete values, in other workds, an assignment to classify features into classes. For example, divide eamils by spam or not.

We can set a list of threshold of some parameters of hypothesis to devide the dataset into different classes.

But if we apply linear regression, some extreme data will have a nonnegligible effect on the hypothesis.

As we all know, the hypothesis usually gives a continuous value, while the classification problem gives discrete values. But apply hypothesis and map values into discrete values sometimes cannot work well.

Here we will focus on the binary classification which has a postive class and a negetive class.

4.1.2 Hypothesis Representation

A logistic regression model want it's $h_\theta(x) \in [0, 1]$. We could apply **Sigmoid Function**:
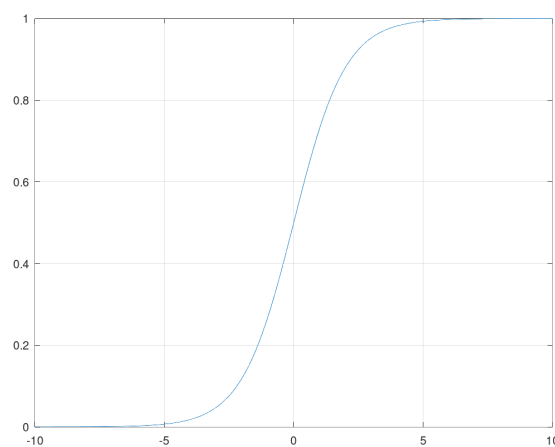
$$g(z) = \frac{1}{1 + e^{-z}}$$

It's like

Figure 4.1: Sigmoid Function

Then, we get:

$$h_\theta(x) = g(\theta^T x)$$

Let's interpret the hypothesis' output: $h_\theta(x)$ is the estimated probability that $y = 1$ on input x. That is **probability that $y = 1$, given a x parameterized by** θ:

$$h_\theta(x) = P(y = 1|x; \theta)$$

4.1.3 Decision Boundary

Why Sigmoid Function makes sense? When $h_\theta(x) > 0.5$ we predict $y = 1$ and predict $y = 0$ in else condition. And it means that $\theta^T x > 0$ or not.

$$h_\theta(x) = g(\theta^T x)$$

By Sigmoid Function we can classify the hypothesis by it's values.

When we get 2 varibles, like $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ and we still make binary classification, wo we can still predict $y = 1$ if $h_\theta(x) \geq 0$ and $y = 0$ if $h_\theta(x) \leq 0$.

Further more, how about we need a non-linear decision boundaries, like a circle or a a ellipse? Just use non-linear functions! For a ellipse:

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

And we define the boundary like:

$$\begin{cases} y = 1, \text{ if } x_1^2 + x_2^2 \geq 1 \\ y = 0, \text{ else} \end{cases}$$

4.2 Logistic Regression Model

4.2.1 Cost Function

For a classification problem, we need the cost function, too. This is related to how to choose the parameters.

In chapters before, we defined as a sum of squared error:

$$\text{Cost}\left(h_\theta(x^{(i)}, y^{(i)})\right) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\text{Cost}(h_\theta(x^{(i)}, y^{(i)}))$$

For a simple cost function, it's non-convex so it does not guarantee to converge.

In logistic regression cost function, we define:

$$\text{Cost}\,(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) \text{ if } y = 1 \\ -\log(1 - h_\theta(x)) \text{ if } y = 0 \end{cases}$$

If the hypothesis gives an approximate output as y, the cost is rather small; but when it goes to the other end, the cost will grow in a very fast speed.

And this cost function could be convex in our problem.

4.2.2 Simplified Cost Function and Gradient Descent

Let's re-write the cost function in a more compact way as:

$$\text{Cost}\,(h_\theta(x), y) = -y \log(h_\theta(x), y) - (1 - y) \log(1 - h_\theta(x))$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}, y^{(i)})$$

$$= -\frac{1}{m} \left[y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

To fit parameters θ, we need to minimize the $J(\theta)$. Then we can apply the gradient descent as in previous chapters.

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}, \text{where } \theta = 0, 1, 2, \cdots, n$$

Note: the $h_\theta(x)$ is different! Now we have $h_\theta(x) = 1/(1 + e^{-\theta x})$.

4.2.3 Advanced Optimization

With this section, we can get logistic regression run more quickly. Like, gradient descent, conjugate gradient, BFGS[1], L-BFGS[2]

We can just simply call `fminunc` to get gradient descent.

4.3 Multi-class Classification

For multiclass problem, we can encode each class with some num like 1, 2, 3, 4 and so on. One-verse-all algorithm is to seperate different class to a special postive class in turn. Or: train a logistic regersss classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$. Finally, pick the $\max_i h_\theta^{(i)}(x)$.

---

[1]Broyden-Fletcher-Goldfarb-Shanno algorithm,

[2]Limited-memory BFGS

4.4 Solve Overfitting: Regularization

4.4.1 The Problem of Overfitting

What's overfitting? If a little bit more complicate dataset cannot be fitted with a line, the linear fit will have a high preconception or bias(underfitting). If we apply a polynomial regression with too high order, hte learned hypothesis will fit the dataset very well but fall to have the ablity to generalize the problem(overfitting).

If we want to address the overfitting, we can

- reduce number of features
  - manually select features to keep
  - model selecttion algorithm
- aplly regularization
  - keep all features, but reduce magnitude or value of some parameters θ_j
  - works well when get lots of features

4.4.2 Cost Function for Regularization

If we want to make some parameters really small, we can add some terms like:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\theta_3^2 + 1000\theta_4^2$$

Small parameters will turn to simpler hypothesis and get more smoth and less prone to overfit. If we want to shrink all parameters, the cost could be implemented as[3]:

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{i=1}^{m} \theta_i^2 \right]$$

The latter term is called regularization term. The regularization will put off a more general hypothesis.

4.4.3 Regularized Linear Regression

If we do gradient descent, the gradient will be different for θ_0 because the extra term does nothing with it.

$$\theta_j := \begin{cases} \theta_j - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j, \text{ where } \theta = 1, 2, \cdots, n \\ \theta_j - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}, \text{ where } \theta = 0 \end{cases}$$

---

[3]remeber that: θ starts with 0, but we just ignore the first term

Similarly, we can still apply the normal equation in conditions:

$$\theta = \left(X^T X + \lambda M\right)^{-1} X^T y, \text{ where } M = \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

4.4.4 Regularized Logistic Regression

$$J(\theta) = -\frac{1}{m}\left[y^{(i)}\log(h_\theta(x^{(i)}), y^{(i)}) + (1 - y^{(i)})\log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\theta_i^2$$

$$\theta_j := \begin{cases} \theta_j - \alpha\dfrac{1}{m}\displaystyle\sum_{i=0}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}, \text{ where } \theta = 1, 2, \cdots, n \\[4mm] \theta_j - \alpha\left[\dfrac{1}{m}\displaystyle\sum_{i=0}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \dfrac{\lambda}{m}\theta_j\right], \text{ where } \theta = 0 \end{cases}$$

Words

ameliorate

preconception

contort

penalize

prone

Chapter 5 chap

5.1 sec