November 17, 2018 1:35 PM

Linear Regression

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

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

The 1-D case

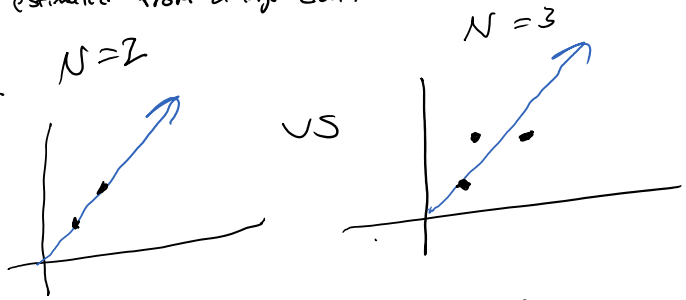
Our goal is to learn a mapping $y = f(x)$, where x & y are both real-valued scalars (i.e $x \in \mathbb{R}$, $y \in \mathbb{R}$)
We will take f to be a linear function

$y = wx + b$ (weight, bias)

The weight (w) and bias (b) are two scalar parameters of the model, which we would like to learn from the training data. In particular, we wish to estimate the weights and bias from the N training pairs. $\{(x_i, y_i)\}_{i=1}^{N}$. Then, once we have values for the weights and bias (w & b), we can compute/predict outputs y for some new input x.

Given two training data points (i.e ... $N=2$), we can exactly solve for the unknown slope w and offset b. Unfortunately, this approach is really sensitive to the noise in the training data measurements, as a result you can't usually trust the resulting model. Instead, we can find a much better model when the two parameters are estimated from a larger data set.

possible example



When $N > 2$ we will not be able to find a unique parameter values for which $y_i = wx_i + b$ for all i, since we have many more constraints than parameters. That is, given noisy data, it is unlikely that 3 or more points will lie on the same line. Rather the best we can hope for is to find a line that is as close to the training point as possible. To this end we want to minimize the errors between the data points and our hypothetical linear model.

in other words: $y_i \approx (wx_i + b)$

The most common way to estimate the parameters of a line in a problem like this is through 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 (a.k.a the squared loss).

In more detail, for the i^{th} training point, the model gives a prediction for y_i namely $wx_i + b$. So let's define the error for the i^{th} training point given the model line with parameters weight and bias to be the vertical distance from the line to the training point

$$e_i = y_i - (wx_i + b)$$
$$= y_i - wx_i - b$$

Empirical loss / energy function / objective function

The sum of the squared errors over all training points is used to measure the quality of the fit.

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

By squaring the errors we obtain a function that is non negative and zero if and only if all the errors are zero. Bonus (shown below) it yields a very straightforward computational solution.

Finding the line that minimizes the squared error is equivalent to solving for the ws and b that minimize $E(w, b)$. This can be done by setting the derivative of E with respect to these parameters to zero and then solving i.e

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

Take derivative

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

optimize (set derivative = 0)

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

optimize (set derivative = 0)

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

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

$$0 = \sum_{i=1}^{N} y_i - w\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} b \quad \text{, summation properties}$$

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

$$Nb = \sum_{i=1}^{N} y_i - w\sum_{i=1}^{N} x_i$$

$$b^* = \frac{\sum_{i=1}^{N} y_i}{N} - w\frac{\sum_{i=1}^{N} x_i}{N}$$

$\frac{\sum_{i=1}^{N} y_i}{N}$: sample mean of y

$\frac{\sum_{i=1}^{N} x_i}{N}$: sample mean of x

$$\boxed{b^* = \hat{y} - w\hat{x}}$$ ← estimate of b

Here we defined $\hat{x}$ and $\hat{y}$ as the averages of the x's and y's respectively. This equation for b^* still depends on w, but don't worry fam, we can substitute this estimate for b in the original energy function then solve for w. The substitution produces the following energy:

$$E(w,b) = \sum_{i=1}^{N}(y_i - (wx_i + b))^2 \quad \text{, recall } b^* = \hat{y} - w\hat{x}$$

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

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

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

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

Take derivative with respect to w to solve for optimal weight

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

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

Optimize

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

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

$$0 = \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i\bar{y} - w\sum x_i(x_i - \bar{x}) - \sum \bar{x}y_i + \sum \bar{x}\bar{y} + w\sum \bar{x}(x_i - \bar{x})$$

, summation property

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

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

$$w^* = \frac{\sum_{i=1}^{n}\left(x_i y_i - x_i\bar{y} - \bar{x}y_i + \bar{y}\bar{x}\right)}{\sum x_i(x_i - \bar{x}) - \bar{x}(x_i - \bar{x})}$$

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

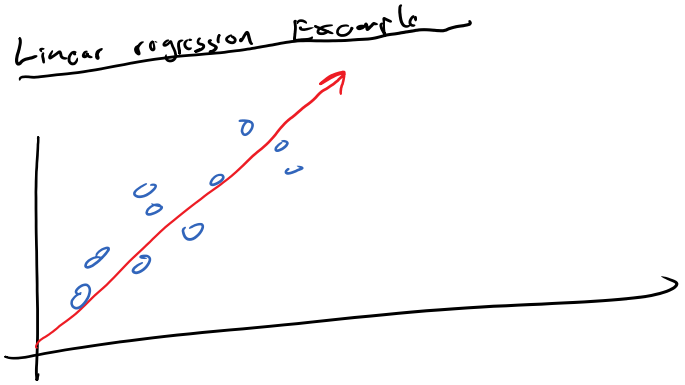
$$w^* = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

factor dawg! :)

The value w^* and b^* are the least squares estimate for the parameters of the linear regression.

November 17, 2018 2:34 PM

Linear regression example



An example of a linear regression! The red line is fit to the blue data points by minimizing the sum of the squared errors. The error of a data point is defined to be the vertical distance from the point to the model line.

Multidimensional inputs

Suppose we wish to learn a mapping from D-dimensional inputs to scalar outputs $x \in \mathbb{R}^D$ $y \in \mathbb{R}$. To this end we will learn a vector of weights w, so 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

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

This mapping can be written as

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

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

If we stack outputs in

Given [illegible] objective function is

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

$$= \|y - \bar{X}\bar{w}\|^2$$

$$= (y - \bar{X}\bar{w})^T (y - \bar{X}\bar{w})$$

$$= (y^T - \bar{w}^T \bar{X}^T)(y - \bar{X}\bar{w})$$

$$= y^T y - y^T \bar{X} w - \bar{w}^T \bar{X}^T y + \bar{w}^T \bar{X}^T \bar{X} \bar{w}$$

Optimize
take derivative

, if we stack outputs in a vector and inputs in a matrix, then we can also write this as $E(\bar{w}) = \|y - \bar{X}\bar{w}\|^2$ where

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \quad \bar{X} = \begin{bmatrix} x_1^T & 1 \\ \vdots & \vdots \\ x_N^T & 1 \end{bmatrix}$$

and $\|\cdot\|$ is the usual Euclidean norm i.e $\|v\|^2 = \sum_{i=1}^{N} v_i^2$

$$\frac{\partial E}{\partial w} = 0 - y^T \bar{X} - \bar{X}^T y + 2 x^T x w$$

set the derivative to zero

$$0 = -y^T \bar{X} - \bar{X}^T y + 2 x^T x w$$

$$y^T \bar{X} + \bar{X}^T y = 2 x^T x w$$

$$2 \bar{X}^T y = 2 x^T x w$$

$$\bar{X}^T y = x^T x w$$

$$(\bar{X}^T X)^{-1} (x^T y) = w^*$$

$$\therefore w^* = (\bar{X}^T X)^{-1} (x^T y)$$

Recall

$$\frac{\partial x^T A x}{\partial x} = (A + A^T) x$$

$$\Rightarrow \bar{w}^T \bar{X}^T \bar{X} \bar{w}$$

let $\bar{X}^T \bar{X} = A$

$$= \bar{w}^T A \bar{w}$$

$$\frac{\partial w^T A w}{\partial w} = (A + A^T) w$$

$$= (\bar{X}^T \bar{X} + (\bar{X}^T \bar{X})^T) w$$ using A back in

$$= (\bar{X}^T \bar{X} + X^T X) w$$

$$= 2 x^T \bar{X} w$$

Multidimensional outputs

In most general case, both inputs and outputs may be multidimensional. For instance, one might want to predict expected crop yields of several different crops, as a function of different growing conditions, such as soil concentration, temperature, and precipitation. With D- dimensional [illegible] from

soil concentration, temperature, and precipitations. With D- dimensional inputs, and K-dimensional outputs $y \in \mathbb{R}^K$, a linear mapping from input to output can be written as:

$y = \bar{W}^T \bar{x}$, where $\bar{W} \in \mathbb{R}^{(D+1) \times K}$. It is convenient to express $\bar{W}$ in terms of its column vectors i.e

$$\bar{W} = [\bar{w}_1, \ldots \bar{w}_K] \equiv \begin{bmatrix} w_1 & \ldots & w_K \\ b_1 & \ldots & b_K \end{bmatrix}$$

In this way, we then express the mapping from the input $\bar{x}$ to the jth element of y as $y_j = \bar{w}_j^T x$. Now given N training samples denoted $\{\bar{x}_i, y_i\}_{i=1}^N$ a natural energy function to minimize, to estimate $\bar{W}$, is just the squared residual error over all training samples and all output dimensions.

i.e

$$E(\bar{W}) = \sum_{i=1}^{N} \sum_{j=1}^{K} (y_{ij} - \bar{w}_j^T \bar{x}_i)^2$$

There are several ways to conveniently vectorize this energy function. One is to express E solely as a sum over output dimensions. That is let y'_j be the N-dimensional vector comprising the j^{th} component of each output training vector.

i.e., $y'_j = [y_{1j}, y_{2j}, \ldots, y_{Nj}]^T$ then we can write

$$E(\bar{W}) = \sum_{j=1}^{K} \| y'_j - \tilde{X} \bar{w}_j \|^2$$

where $\tilde{X}^T = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_N]$

Non Linear Regression

November 18, 2018 10:19 AM

Linear models are often insufficient to capture the real world phenomena. That is, the relation between the inputs and outputs we want to be able are not linear.

As a consequence, non linear models are often required.

Basically, linear models usually do not fit real world events. Relationship between x & y are not linear. So we need non-linear models. Ya feel homie?

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

① $y = f(x)$

Recall, In linear regression we used

$$y = f(x) = wx + b$$

(w: weight, b: bias)

where the weight (w) and bias (b) were the parameters. Previously, we estimated the parameters to fit the model to the data. We did this by computing b^* and w^* by taking the taking the derivative with respect to the parameter and setting it equal to zero. (check linear regression notes)

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

What non linear function should we choose?

- linear?
- trig?
- summation?

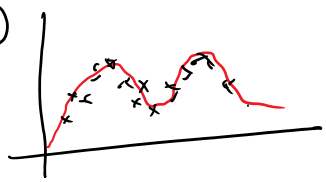
In principle $f(x)$ could be anything.

However, the form we choose will have a major impact on the effectiveness of the regression.

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

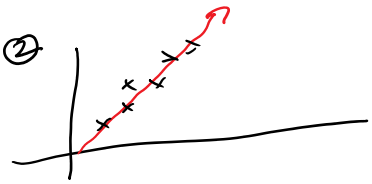
Eg.

(1)



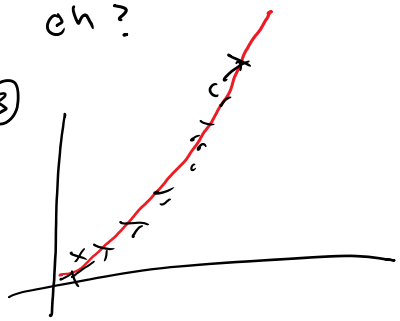
Looks like a trig function eh?

(2)



Looks like a straight line boi

(3)



hmm... looks cubic or quadratic. Somewhat exponential.

Ideally, the form of the model would be matched exactly to the underlying phenomenon.

∴ if we were to model a linear process, then we would want to use linear regression.

if we were measuring a physical process, we could, in principle, model $f(x)$ using the appropriate equations from physics and so on.

But basically, modelling the process precisely is too difficult. In these cases, we typically turn to a few models in Machine Learning that are widely used and effective for many problems.

These methods include

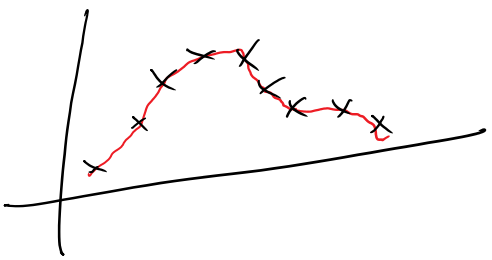
1) Radial Basis Function (RBF)
2) Artificial Neural Networks (ANN)

3) K-Nearest Neighbours (K-NN)

% btw, there is another important choice to be made, namely, the choice of an objective function for learning, or, equivalently, the underlying noise model.

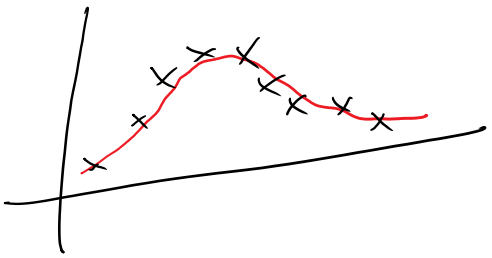
Basically a smoothness term, to help avoid overfitting

w/o smoothness



vs

with smoothness



Notice one overfits, while the other one helps avoid overfitting.

3.1 Basis function Regression

A common choice for the function $f(x)$ is a basis function representation:

$$y = f(x) = \sum_{k=1} w_k b_k(x)$$

for the case of 1D inputs x. The functions $b_k(x)$ are called basis functions. Often it will be convinient to express this model in vector form, for which we define

$$b(x) = [b_1(x), \ldots, b_M(x)]^T$$
$$w = [w_1, \ldots, w_M]^T$$

where M is the number of basis functions. We can then rewrite the model as

$$\boxed{y = f(x) = b(x)^T w}$$

Note

$$b(x^T) w$$
$$= [b_1(x), b_2(x), \ldots b_M(x)] \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix}$$

the model ...

$$y = f(x) = b(x)^T w$$

Note

$b(x^T)w$

$= [b_1(x), b_2(x), \cdots b_m(x)] \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$

$= w_1 b_1(x) + w_2 b_2(x) \cdots w_m b_m(x)$

$= \sum_{i=1}^{m} w_i b_i(x)$

$\Rightarrow \sum_{k} w_k b_k(x)$

Two common choices of basis functions are polynomials and Radial Basis Functions (RBF)

A simple, common basis for polynomials are monomials i.e

$b_0(x) = 1$

$b_1(x) = x$

$b_2(x) = x^2$

$b_3(x) = x^3$

$\vdots$

$b_n(x) = x^n$

With such a monomial basis, the regression model has the form

$$f(x) = \sum w_k b_k(x)$$

$$= \sum w_k x^k$$

Radial Basis Function

$$b_1(x) = \exp\left(-\frac{(x - c_1)^2}{2\sigma^2}\right)$$

$$b_2(x) = \exp\left(-(x - c_2)^2\right)$$

$$b_2(x) = \exp\left(-\frac{(x-c_2)^2}{2\sigma^2}\right)$$

$$\vdots$$

$$b_K(x) = \exp\left(-\frac{(x-c_K)^2}{2\sigma^2}\right)$$

where $\exp a \equiv e^a$

$$f(x) = \sum w_k b_k(x)$$

$$= \sum w_k \exp\left(-\frac{(x-c_k)^2}{2\sigma^2}\right)$$

where c_k is the center (i.e, the location) of the basis function and σ^2 determines the width of the basis function. Both of these are parameters of the model that must be determined (or estimated) from the training data.

To fit these model, we again use least-squares regression, minimizing the sum of squared residual error between model predictions and the given training outputs

$$E(w) = \sum (y_i - f(x_i))^2$$

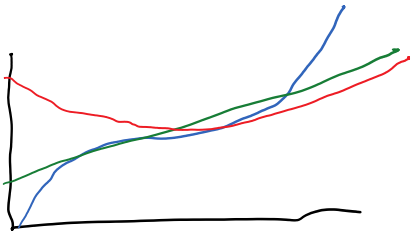
$$= \sum \left(y_i - \sum_k w_k b_k(x)\right)^2$$

Note: the objective function has the same form as the linear regression model in the previous chapter, except that inputs are now the $b_k(x)$ values.

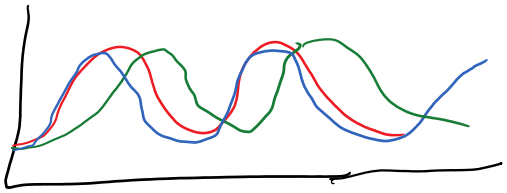
In particular, E is still quadratic in the weight parameters, w, and hence they can be estimated the same way. We therefore rewrite the objective function in matrix-vector form as follows.

$$E(w) = \|y - Bw\|^2$$

where $\|\cdot\|$ denotes the Euclidean (L^2) norm, and the elements of the matrix B are given by $B_{i,j} = b_j(x_i)$ (for row i and column j). In Matlab the least-squares estimate can be computed as $w^* = B \backslash y$



polynomial basis



Radial Basis

Picking other parameters

Finding optimal positions and widths of the RBF basis function is not so simple.

For example, if we were to optimize these parameters to minimize the squared-error, then we would end up with one basis function centered at each data point, and with tiny width. These RBF functions would exactly fit the data, but won't be a good model as it overfits, ∴ won't be able to predict future data well.

Common Heuristic used to determine these parameters, without overfitting the training Data.

Picking Basis centers

1. Place the centers uniformly spaced in the region containing the data.
2. Place one center at each data point. (Expensive with large data)
3. Cluster the data, and use one center for each cluster

Picking width

1. manually try (trial and error) ≈ LOL me in A1
2. Use the average squared distances (or median distances) to neighboring centers scaled by a constant, to be the width. This approach also allows you to use different widths for different basis functions, it allows the basis function to be spaced non-uniformly.

November 20, 2018 10:46 AM

Overfitting and Regularization

As mentioned earlier, directly minimizing the squared-error can lead to an effect called overfitting, wherein we fit the training data extremely well (i.e with low error), yet we obtain a model that produces very poor predictions on future test data whenever the test inputs differ from the training inputs.

Overfitting can be understood in many ways, all of which are variations on the same underlying pathology:

1. Problem is not sufficiently constrained: For example, if we have 10 measurements and we need ten model parameters, then we can obtain a perfect fit to the data. Usually you need much more data than there are model parameters.

2. Fitting noise: Overfitting can occur when the model is so powerful that it can fit the data and also the random noise in the data.

3. Discarding Uncertainty: The posterior probability distribution of the unknowns is insufficiently peaked to pick a single estimate.

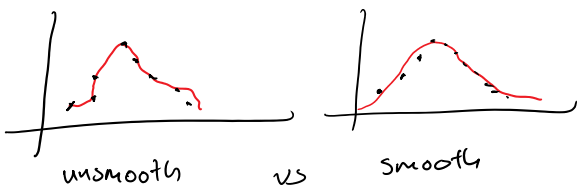
Two main solutions to overfitting problems:

add prior knowledge and handling uncertainty.

In many cases there is some sort of prior knowledge we can leverage.

A very common assumption is that the underlying is likely to be smooth

For instance, we might believe that lacking any other information, our function should have small derivatives.



unsmooth vs smooth

Smoothness also reduces complexity

It is also easier to estimate smooth models from small data sets.

In the extreme, if we make no prior assumptions about the nature of the fit then it is impossible to learn and generalize at all.

Smoothness assumptions are one way of constraining the space of models so that we have any hope of learning from small data sets.

One way to add smoothness is to parameterize the model in a smooth way (e.g., making the width parameter for RBFs larger, using only low order polynomial basis functions). But this limits the expressiveness of the model. In particular, when we have lots and lots of data, we would like the data to be able to "override" the smoothness assumptions. With large widths, it is impossible to get highly curved models no matter what the data says.

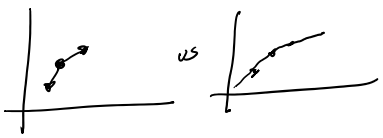
Instead, we can add regularization (smoothness term), in which a extra terms are added to the learning objective function, often to prefer (encourage) smooth models.

RBF regression with scalar outputs, and with many other type of basis functions or multidimensional outputs can be done with an objective function of the form:

$$E(w) = \|y - Bw\|^2 + \lambda \|w\|^2$$

data term; smoothness term

This objective function has two terms. The first term, called the data term, measures the model fit to the training data. The second term, often called the smoothness term, penalizes non-smoothness (rapid changes or $\|w\|$). This particular smoothness term ($\|w\|$) is called weight decay, because it tends to make the weight smaller.



Note: Estimation with $E(w) = \|y - Bw\|^2 + \lambda \|w\|^2$ is also known as ridge regression.

Overfitting vs Underfitting

Overfitting: fits training data too well and performs poor on test

Underfitting: does not fit training data too well.

Regularized least squares regression

$$E(w) = \|y - Bw\|^2 + \lambda \|w\|^2$$

$$= (y - Bw)^T (y - Bw) + \lambda (w^T w)$$

$$= y^T y - y^T Bw - w^T B^T y + w^T B^T Bw + \lambda w^T w$$

$$= y^T y - 2 w^T B^T y + w^T B^T Bw + \lambda w^T w$$

$$= y^T y - 2 w^T A + w^T A w + \lambda w^T w$$

$$(A^T)^T = A$$

$= (y - Bw)^T (y - Bw) + \lambda w^T w$

$= (y^T - w^T B^T)(y - Bw) + \lambda w^T w$

$= y^T y - y^T Bw - w^T B^T y + w^T B^T Bw + \lambda w^T w$

$E(w) = y^T y - y^T Bw - w^T B^T y + w^T B^T Bw + \lambda w^T w$

$\frac{\partial E}{\partial w} = 0 - y^T B - B^T y + 2B^T Bw + 2\lambda w$

$0 = -y^T B - B^T y + 2B^T Bw + 2\lambda w$

$0 = (-y^T B)^T - B^T y + 2B^T Bw + 2\lambda w$

$0 = -B^T y - B^T y + 2B^T Bw + 2\lambda w$

$0 = -2B^T y + 2B^T Bw + 2\lambda w$

$0 = -B^T y + B^T Bw + \lambda w$

$B^T y = w(B^T B + \lambda)$

$(B^T B + \lambda)^{-1} B^T y = w^*$

$= w \cdot I \cdot w$

$A^T) = (A + w$

$= (B^T B + (B^T B)^T) w$

$B^T B = (I + B^T B) w$

$= 2B^T B w$

November 20, 2018 8:17 PM

Basic Probability Theory

Bayesian reasoning provides a formal and consistent way to reason in the presence of uncertainty. Bayesian probability theory is distinguished by defining probabilities as degrees of belief.

Frequentist statistic provides the probability of an event is defined as

---

In classical logic, we have some statements that may be true or false, we have a set of rules which allow us to determine the truth or falsity of new statements. For example, suppose we introduce two statements named A & B:

where

A ≡ "My car was stolen"

B ≡ "My car is not in the parking spot where I remember leaving it"

Moreover, let us assert the rule $A \Rightarrow B$

Then, if A is true, we deduce logically that B must also be true.

Alternatively, if I find my car where I left it then I may infer that it was not stolen ($\bar{A}$) by contrapositive $\bar{B} \Rightarrow \bar{A}$

Classical logic provides a model of how humans might reason, and a model of how we might build an "intelligent" computer.

Classical logic shortcomings:
- assumes that all logic is absolute
- logic requires that we know some facts about the world with absolute certainty, and then, we may deduce only those facts which must follow with absolute certainty.
- real world, almost no facts that we know with absolute certainty.
- what we know about the world is uncertain and is only based on our experience.
- However, we can make predictions with great confidence

Basic Definition & Rules

1. $0 \leq P(A) \leq 1$
2. $P(A) = 1$, certain A is true
3. $P(A) = 0$, certain A is false

Conditional

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

Product Rule

2. $P(A) = 1$, [illegible]
3. $P(A) = 0$, certain A is false

Joint $P(A \cap B) = P(A, B) = P(B, A) = P(B \cap A)$

Product Rule

$P(A, B) = P(A|B)P(B)$

Sum Rule

$P(A) + P(\bar{A}) = 1$

$\sum_i P(A_i) = 1$

$\sum_i P(A_i | C) = 1$

Marginalization

$P(B) = \sum_i P(A_i, B)$

if A_i, are mutually exclusive statements

More fun rules!

$P(A) + P(\bar{A}) = 1$

$P(A|B) + P(\bar{A}|B) = 1$

$P(A|B)P(B) + P(\bar{A}|B)P(B) = P(B)$

$P(A, B) + P(\bar{A}, B) = P(B)$

if $\perp$ then

$P(A, B) = P(A)P(B)$.

$P(A|B) = P(A)$

TLDR

- $P(A) \in [0, 1]$
  - $P(A, B) = P(A|B)P(B)$
  - $P(A) + P(\bar{A}) = 1$
  - iif $A \perp B$, $P(A, B) = P(A)P(B)$
  - Marginalizing $P(B) = \sum_i P(A_i, B)$

Binomial / Multinomial Distribution

Binomial

$$P(K = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, \ldots, n$$

where

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Multinomial

$$P(X_1 = x_1, X_2 = x_2, X_3 = x_3 \cdots X_k = x_k)$$

$$= \frac{n!}{\prod_{i=1}^{k} x_i!} \prod_{i=1}^{k} p_i^{x_i}$$

$$= n! \prod_{i=1}^{k} \left(\frac{p_i^{x_i}}{x_i!} \right)$$

Expectation

$$E[X] = \sum_i P(x_i) x_i$$

$$E[f(X)] = \sum P(x_i) f(x_i)$$