data science for (physical) scientists 5

the ABC of regression

dr.federica bianco | *fbb.space* | *fedhere* | *fedhere*

this slide deck

https://slides.com/federicabianco/dsps_5

1 what is a model

2 the principle of parsimony

epistemology

3 fitting a model to data

line fit

standard linear fit

higher order equation

uncertainties in the fit parameters

generative models

4 cross-validation

what is a model?

it's always a simplification

what is a model?

a *mathematical* representastion of reality

In applying mathematics to subjects such as physics or statistics we make tentative assumptions about the real world which we know are false but which we believe may be useful nonetheless.

George Box, 1976

- no model is right
- some models are useful

why do we model?

1

https://www.youtube.com/embed/Tk2v1UaTgmk?enablejsapi=1

1. to undestand

$$h(t+t_0) = v_0 t - \frac{1}{2} g t^2$$

what is a model?

why do we model?

2. to predict

$$r(\nu) = \frac{a(1-e^2)}{1+e\cos(\nu)}$$

why do we model?

1

to understand + predict

what is a model?

the best way to think about it

in the ML context:

a model is a low dimensional representation of a higher dimensionality datase

1 what is machine learning

what is machine learning?

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959

what is machine learning?

Machine Learning models are parametrized representation of "reality" where the parameters are learned from finite sets of realizations of that reality

Machine Learning is the disciplines that conceptualizes, studies, and applies those models.

Key Concept

what is machine learning?

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959

Model:

a mathematical formula
with parameters

model $y = ax + b$

parameters: slope (*a*), intercept (*b*)

what is machine learning?

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959

Model:

a mathematical formula
with parameters

model

parameters: slope (*a*), intercept (*b*)

$$y = ax + b$$

model variable: x - for example time, location, energy

what is machine learning?

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959

Model:

a mathematical formula
with parameters

model

$$y = a_1 x_1 + a_2 x_2 + b$$

model variable: x - for example time, location, energy

what is machine learning?

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959

Model:

a mathematical formula with parameters

Data:

a set of observations

what is machine learning?

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959

Model:

a mathematical formula with parameters

Data:

a set of observations

for every parameter there are an infinity of models

what is machine learning?

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959

Model:

a mathematical formula with parameters

Data:

a set of observations

Use the data to *learn* the parameters of the model

the best way to think about it *in the ML context:*

a model is a low dimensional representation of a higher dimensionality dataset

Use the data to *learn* the parameters of the model

1.2 what is machine learning

what is machine learning?

supervised learning

extract features and create models that allow prediction where the correct answer is known for a subset of the data

unsupervised learning

identify features and create models that allow to understand structure in the data

what is machine learning?

- k-Nearest Neighbors
- Regression
- Support Vector Machines
- Neural networks
- Classification/Regression Trees

supervised learning

extract features and create models that allow prediction where the correct answer is known for a subset of the data

- clustering
- Principle Component Analsysis
- Apriori (association rule)

unsupervised learning

identify features and create models that allow to understand structure in the data

unsupervised vs supervised learning

Unsupervised learning

- understanding structure
- anomaly detection
- dimensionality reduction

Clustering

partitioning the feature space so that the existing data is grouped (according to some target function!)

All features are observed for all datapoints

unsupervised vs supervised learning

understand the structure of a feature space

Clustering

partitioning the feature space so that the existing data is grouped (according to some target function!)

unsupervised vs supervised learning

understand the structure of a feature space

Unsupervised learning

- understanding structure
- anomaly detection
- dimensionality reduction

Clustering

partitioning the feature space so that the existing data is grouped (according to some target function!)

All features are observed for all datapoints

unsupervised vs supervised learning

prediction and classification based on examples

Unsupervised learning

- understanding structure
- anomaly detection
- dimensionality reduction

Clustering

partitioning the feature space so that the existing data is grouped (according to some target function!)

All features are observed for all datapoints

Classifying & regression

finding functions of the variables that allow to predict unobserved properties of new observations

unsupervised vs supervised learning

prediction and classification based on examples

Unsupervised learning

- understanding structure
- anomaly detection
- dimensionality reduction

Clustering

partitioning the feature space so that the existing data is grouped (according to some target function!)

Classifying & regression

finding functions of the variables that allow to predict unobserved properties of new observations

All features are observed for all datapoints

unsupervised vs supervised learning

prediction and classification based on examples

Unsupervised learning

- understanding structure
- anomaly detection
- dimensionality reduction

Clustering

partitioning the feature space so that the existing data is grouped (according to some target function!)

All features are observed for all datapoints

Supervised learning

- classification
- prediction
- feature selection

Classifying & regression

finding functions of the variables that allow to predict unobserved properties of new observations

Some features are not observed for some data points we want to predict them.

unsupervised vs supervised learning

Unsupervised learning

All features are observed for all datapoints

and we are looking for structure in the feature space

Supervised learning

Some features are not observed for some data points we want to predict them.

The datapoints for which the target feature is observed are said to be "*labeled*"

also...

Semi-supervised learning

A small amount of labeled data is available. Data is cluster and clusters inherit labels

Active learning

The code can interact with the user to update labels and update model.

what is machine learning?

- **k-Nearest Neighbors**
- **Regression**
- Support Vector Machines
- **Classification/Regression Trees**
- **Neural networks**

- **clustering**
- Principle Component Analsysis
- Apriori (association rule)

supervised learning

extract features and create models that allow prediction where the correct answer is known for a subset of the data

unsupervised learning

identify features and create models that allow to understand structure in the data

unsupervised vs supervised learning

used to:

- classify based on examples,
- understand structure of feature space
- regression (classification with infinitely small classes)
- understand which features are important in prediction (to get close to causality)

13

train, test, and validate

In ML models need to be "validated":

1. split the data into a training and a test set (typical split 70/30).
2. learn the model parameters by "training" the model on the training set
3. "test" the model on the test set: measure the accuracy of the prediction (e.g. as the distance between the prediction and the test data).

The performance on the model is the performance achieved on the test set.

a significance performance degradation on the test compared to training set indicates that the model is "overtrained" and does not generalize well.

An upgrade on this workflow is to create a training, a test, and a validation test. Iterate between training and test to achieve optimal performance, then measure accuracy on the validation set.This is because you can use the test set performance to tune the model hyperparameters (model selection) but then you would report a performance that is tuned on the test set.

ML standard

training – testing – validating

validating a model

How do we measure if a model is good?

Accuracy

Precision

Recall

ROC

AOC

We will talk more about this later...

but for now focus on

regression performance metrics

validating a model

How do we measure if a model is good? $\epsilon_i = y_i - f(t_i)$

Accuracy

Precision

Recall

ROC

AOC

We will talk more about this later...

but for now focus on

regression performance metrics

$$AE = \sum_i |\epsilon_i|$$ **Absolute error**

$$SE = \sum_i \epsilon_i^2$$ **Squared error**

$$MSE = \frac{1}{N} SE$$ **Mean squared error**

$$RMSE = \sqrt{MSE}$$ **Root mean squared error**

$$rMSE = \frac{MSE}{\sigma^2}$$ **Relative mean squared error**

$$R^2 = 1 - rMSE$$ **R squared**

validating a model

How do we measure if a model is good? $\epsilon_i = y_i - f(t_i)$

Accuracy

Precision

Recall

ROC

AOC

We will talk more about this later...

but for now focus on

regression performance metrics

$$AE = \sum_i |\epsilon_i|$$ **Absolute error**

$$SE = \sum_i \epsilon_i^2$$ **Squared error**

$$MSE = \frac{1}{N} SE$$ **Mean squared error**

$$RMSE = \sqrt{MSE}$$ **Root mean squared error**

$$rMSE = \frac{MSE}{\sigma^2}$$ **Relative mean squared error**

$$R^2 = 1 - rMSE$$ **R squared**

validating a model

How do we measure if a model is good? $\epsilon_i = y_i - f(t_i)$

Accuracy

Precision

Recall

ROC

AOC

We will talk more about this later...

but for now focus on

regression performance metrics

$$AE = \sum_i |\epsilon_i| \equiv L_1$$ **Absolute error**

$$SE = \sum_i \epsilon_i^2 \equiv L_2$$ **Squared error**

$$MSE = \frac{1}{N} SE$$ **Mean squared error**

$$RMSE = \sqrt{MSE}$$ **Root mean squared error**

$$rMSE = \frac{MSE}{\sigma^2}$$ **Relative mean squared error**

$$R^2 = 1 - rMSE$$ **R squared**

validating a model

How do we measure if a model is good? $\epsilon_i = y_i - f(t_i)$

Accuracy

Precision

Recall

ROC

AOC

We will talk more about this later...

but for now focus on

regression performance metrics

$$AE = \sum_i |\epsilon_i| \equiv L_1$$ **Absolute error**

$$SE = \sum_i \epsilon_i^2 \equiv L_2$$ **Squared error**

$$MSE = \frac{1}{N} SE$$ **Mean squared error**

$$RMSE = \sqrt{MSE}$$ **Root mean squared error**

$$rMSE = \frac{MSE}{\sigma^2}$$ **Relative mean squared error**

$$R^2 = 1 - rMSE$$ **R squared**

validating a model

How do we measure if a model is good? $\epsilon_i = y_i - f(t_i)$

Accuracy

Precision

Recall

ROC

AOC

Split the sample in test and training sets

Train on the training set

Test (measure accuracy) on the test set

We will talk more about this later...

but for now focus on

regression performance metrics $R^2 = 1 - rMSE$

validating a model

```python
from sklearn.model_selection import train_test_split

def line(x, intercept, slope):
    return slope * x + intercept

def chi2(args, x, y, s):
    a, b = args
    return sum((y - line(x, a, b))**2 / s)

x_train, x_test, y_train, y_test, s_train, s_test = train_test_split(
     x, y, s, test_size=0.25, random_state=42)

initialGuess = (10, 1)

chi2Solution_goodsplit = minimize(chi2, initialGuess,
        args=(x_train, y_train, s_train))

print("best fit parameters from the minimization of the chi squared: " +
       "slope {:.2f}, intercept {:.2f}".format(*chi2Solution_goodsplit.x))

print("R square on training set: ", Rsquare(chi2Solution_goodsplit.x, x_train, y_train))
print("R square on test set: ", Rsquare(chi2Solution_goodsplit.x, x_test, y_test))
```

In ML models need to be "validated":

1. split the data into a training and a test set (typical split 70/30).
2. learn the model parameters by "training" the model on the training set
3. "test" the model on the test set: measure the accuracy of the prediction (e.g. as the distance between the prediction and the test data).

The performance on the model is the performance achieved on the test set.

a significance performance degradation on the test compared to training set indicates that the model is "overtrained" and does not generalize well.

An upgrade on this workflow is to create a training, a test, and a validation test. Iterate between training and test to achieve optimal performance, then measure accuracy on the validation set.This is because you can use the test set performance to tune the model hyperparameters (model selection) but then you would report a performance that is tuned on the test set.

ML standard

2.1 Linear Regression

analytical solution

Linear Regression Normal Equation

It can be shown that the optimal parameters for a line fit to data without uncertainties is:

$$(X^T \cdot X)^{-1} \cdot X^T \cdot \vec{y} = \binom{a}{b}$$

```
x = np.sort(10 * np.random.rand(N))
y = x * m_true + b_true
yerr = 0.1 + 0.5 * np.random.rand(N)
y += np.abs(f_true * y) * np.random.randn(N) + yerr *
np.random.randn(N)

X = np.c_[np.ones(len(x)), x]
theta_best = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)
```

Linear Regression Normal Equation

It can be shown that the optimal parameters for a line fit to data without uncertainties is:

$$(X^T \cdot X)^{-1} \cdot X^T \cdot \vec{y} = \binom{a}{b}$$

```
x = np.sort(10 * np.random.rand(N))
y = x * m_true + b_true
yerr = 0.1 + 0.5 * np.random.rand(N)
y += np.abs(f_true * y) * np.random.randn(N) + yerr *
np.random.randn(N)

X = np.c_[np.ones(len(x)), x]
theta_best = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)
```

Linear Regression Normal Equation

It can be shown that the optimal parameters for a line fit to data without uncertainties is:

$$(X^T \cdot X)^{-1} \cdot X^T \cdot \vec{y} = \binom{a}{b}$$

2xN Nx2 2xN Nx1

$$x = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{pmatrix}$$

```
theta_best = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)
```

Linear Regression Normal Equation

It can be shown that the optimal parameters for a line fit to data without uncertainties is:

$$(X^T \cdot X)^{-1} \cdot X^T \cdot \vec{y} = \binom{a}{b}$$

2xN Nx2 2xN Nx1

$$x = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{pmatrix}$$

```
theta_best = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)
```

Linear Regression Normal Equation

It can be shown that the optimal parameters for a line fit to data without uncertainties is:

$$(X^T \cdot X)^{-1} \cdot X^T \cdot \vec{y} = \binom{a}{b}$$

```
x = np.sort(10 * np.random.rand(N))
y = x * m_true + b_true
yerr = 0.1 + 0.5 * np.random.rand(N)
y += np.abs(f_true * y) * np.random.randn(N) + yerr *
np.random.randn(N)

X = np.c_[np.ones(len(x)), x]
theta_best = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)
```

We can let sklearn solve the equation for us:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

X = np.c_[np.ones(len(x)), x]
lr.fit(X, y)
lr.coef_, lr.intercept_
```

Linear Regression Normal Equation

It can be shown that the optimal parameters for a line fit to data without uncertainties is:

$$(X^T \cdot X)^{-1} \cdot X^T \cdot \vec{y} = \binom{a}{b}$$

```
x = np.sort(10 * np.random.rand(N))
y = x * m_true + b_true
yerr = 0.1 + 0.5 * np.random.rand(N)
y += np.abs(f_true * y) * np.random.randn(N) + yerr *
np.random.randn(N)

X = np.c_[np.ones(len(x)), x]
theta_best = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)
```

We can let sklearn solve the equation for us:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

X = np.c_[np.ones(len(x)), x]
lr.fit(X, y)
lr.coef_, lr.intercept_
```

2.2 (Linear) Regression

numerical solutions

Optimization

Gradient descent is an iterative algorithm, that starts from a random point on a function and travels down its slope in steps until it reaches the lowest point of that function.

(*Toward data science*)

Univariate Linear regression $y_i = (a\, x_i + b)$

Target Funcion $\chi^2 = \sum_{i=1}^{N} \left(\frac{(y_i - (a\, x_i + b))^2}{\sigma_i^2} \right)$

Optimization

Univariate Linear regression $y_i = (a\ x_i)$

Target Funcion $\chi^2 = \sum_{i=1}^{N} \left(\frac{(\hat{y}_i - (a\ x_i))^2}{\sigma_i^2} \right)$

Optimizing the Objective Function

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

Minimum (optimal) χ^2

➡ a = 4

Many packages do it in python!

but what are they doing??

- numpy (e.g. np.polyfit)
- scipy (e.g. sp.optimize.curve_fit)
- skitlearn-optimize

objective function:

what you want to optimize for

In principle, there are many choices for objective function. But the only procedure that is truly justified—in the sense that it leads to interpretable probabilistic inference, is to make a generative model for the data.

Hogg&Bovey&Lang Section 2

https://arxiv.org/pdf/1008.4686.pdf

Objective Function

If there is no analytical solution

to select the "best" set of parameters we need a plan: we need to choose a function of the parameters to minimize or maximize

Objective Function

If there is no analytical solution

to select the best fit parameters we define a function of the parameters to minimize or maximize

Objective Function

If there is no analytical solution

to select the best fit parameters we define a function of the parameters to minimize or maximize

Objective Function

If there is no analytical solution

to select the best fit parameters we define a function of the parameters to minimize or maximize

which is the "best fit" line?

A , B, C, D?

Objective Function

If there is no analytical solution

to select the best fit parameters we define a function of the parameters to minimize or maximize

which is the "best fit" line?

A , B, C, D?

Objective Function

If there is no analytical solution

to select the best fit parameters we define a function of the parameters to minimize or maximize

$$L_1 = \sum_{i=1}^{N} |f(x) - y|$$

Objective Function

If there is no analytical solution

to select the best fit parameters we define a function of the parameters to minimize or maximize

$$L_1 = \sum_{i=1}^{N} |f(x) - y|$$

$$L_2 = \sum_{i=1}^{N} (f(x) - y)^2$$

Objective Function

If there is no analytical solution

to select the best fit parameters we define a function of the parameters to minimize or maximize

$$L_1 = \sum_{i=1}^{N} |f(x) - y|$$

$$L_2 = \sum_{i=1}^{N} (f(x) - y)^2$$

Objective Function

If there is no analytical solution

to select the best fit parameters we define a function of the parameters to minimize or maximize

$$L_1 = \sum_{i=1}^{N} |f(x) - y|$$

$$L_2 = \sum_{i=1}^{N} (f(x) - y)^2$$

$$\chi^2 = \sum_{i=1}^{N} \frac{(f(x)-y)^2}{\sigma^2}$$

chi square: relates to the likelihood if the distribution is Gaussian

~~Stochastic~~ Gradient Descent

If there is no analytical solution

to select the "best" set of parameters we need a plan:
we need to choose a function of the parameters to minimize or maximize

$$L_1 = \sum_{i=1}^{N} |f(x) - y|$$

$$L_2 = \sum_{i=1}^{N} (f(x) - y)^2$$

$$\chi^2 = \sum_{i=1}^{N} \frac{(f(x)-y)^2}{\sigma^2}$$

```
from scipy.optimize import minimize
def fitfunc(args, x, y):
  m, b = args
  return np.sum((y-line(x, m, b))**2)

initialGuess = (10, 1)
solution  = minimize(fitfunc, initialGuess, args=(x, y))
```

~~Stochastic~~ Gradient Descent

If there is no analytical solution

to select the "best" set of parameters we need a plan:
we need to choose a function of the parameters to minimize or maximize

$$L_1 = \sum_{i=1}^{N} |f(x) - y|$$

$$L_2 = \sum_{i=1}^{N} (f(x) - y)^2$$

$$\chi^2 = \sum_{i=1}^{N} \frac{(f(x)-y)^2}{\sigma^2}$$

```
def chi2(args, x, y, yerr):
  m, b = args
  return np.sum((y-line(x, m, b))**2 / yerr**2)
chi2((m_true, b_true), x, y, yerr)

initialGuess = (10, 1)
solutionChi2  = minimize(chi2, initialGuess, args=(x, y, yerr
solutionChi2.x
```

What is Machine Learning? Machine Learning models are parametrized representations of "reality" where the parameters are learned from finite sets of realizations of that reality. Machine Learning is the discipline that conceptualizes, studies, and applies those models.

Model selection: Choosing a model i.e. a mathematical formula which we expect to be a simplified representation of our observations.

Model fitting: Determining the best set of parameters to fit the observations within a chosen model.

Objective Functions and optimization: To find the best model parameters we define a function of the data and parameters *f(data, parameters)* to be minimized or maximized.

Key Concepts

Data analysis recipes: Fitting a model to data

Intro and Chapter 1; pages 1-8

D. Hogg et al. https://arxiv.org/abs/1008.4686

Lots of details about how to properly treat outliers, uncertainties, assumptions in fitting a line to data. Witty comments make it entertaining. Exercise it make it very helpful

Reading

Data analysis recipes: Fitting a model to data*

David W. Hogg
Center for Cosmology and Particle Physics, Department of Physics, New York University
Max-Planck-Institut für Astronomie, Heidelberg

Jo Bovy
Center for Cosmology and Particle Physics, Department of Physics, New York University

Dustin Lang
Department of Computer Science, University of Toronto
Princeton University Observatory

Abstract

We go through the many considerations involved in fitting a model to data, using as an example the fit of a straight line to a set of points in a two-dimensional plane. Standard weighted least-squares fitting is only appropriate when there is a dimension along which the data points have negligible uncertainties, and another along which all the uncertainties can be described by Gaussians of known variance; these conditions are rarely met in practice. We consider cases of general, heterogeneous, and arbitrarily covariant two-dimensional uncertainties, and situations in which there are bad data (large outliers), unknown uncertainties, and unknown but expected intrinsic scatter in the linear relationship being fit. Above all we emphasize the importance of having a "generative model" for the data, even an approximate one. Once there is a generative model, the subsequent fitting is non-arbitrary because the model permits direct computation of the likelihood of the parameters or the posterior probability distribution. Construction of a posterior probability distribution is indispensible if there are "nuisance parameters" to marginalize away.

References

8 ways to do linear regression and measure their speed

Data analysis recipes: Fitting a model to data

D. Hogg et al. https://arxiv.org/abs/1008.4686 - lots of details about how to properly treat outliers, uncertainties, assumptions in fitting a line to data. Witty comments make it entertaining. Exercise it make it very helpful

AstroML Chapter 10 - Intro

HOMLwSKLKerasTF Chapter 4 pages 111-117

Elements of Statistical Learning Chapter 3 Section 1 and 2

Data Analysis: A Bayesian Tutorial

https://www.amazon.com/Data-Analysis-Bayesian-Devinderjit-Sivia-ebook/dp/B01BHLXKEI

Elements of Statistical Learning Chapter 7

https://web.stanford.edu/~hastie/Papers/ESLII.pdf

Sorry ARIMA, but I'm Going Bayesian

https://towardsdatascience.com/implementing-facebook-prophet-efficiently-c241305405a3

emcee: The MCMC Hammer

https://arxiv.org/abs/1202.3665

Forecasting at scale

(the facebook Prophet paper)

https://peerj.com/preprints/3190

references

Bayes theorem

$$P(\text{model}|\text{data})P(\text{data}) = P(\text{data}|\text{model})P(\text{model})$$

Likelihood Prior

$$P(\text{model}|\text{data}) = \frac{P(\text{data}|\text{model})P(\text{model})}{P(\text{data})}$$

Posterior Evidence

Likelihood

Probability of data given a model, e.g. Gaussian

unknown variables

$$P(\vec{x}|\mu,\sigma) \sim \frac{1}{\sqrt{2\pi}\sigma} exp \frac{-(\vec{x}-\mu)^2}{2\sigma^2}$$

Posterior: the probability of the model given the data. Same Gaussian assumptions

$$Post(\mu,\sigma|\vec{x}) \sim \frac{1}{\sqrt{2\pi}\sigma} exp \frac{-(\vec{x}-\mu)^2}{2\sigma^2}$$

unknown variables

Maximizing the likelihood we seek the parameters that maximize the probability of the *observed* data under the chosen model

The problem of fitting models to data reduces to finding the

maximum *likelihood*

of the data given the model

This is effectively done by finding the **minimum** of the

-log(*likelihood*)

Data analysis recipes:
Fitting a model to data*

David W. Hogg
Center for Cosmology and Particle Physics, Department of Physics, New York University
Max-Planck-Institut für Astronomie, Heidelberg

Jo Bovy
Center for Cosmology and Particle Physics, Department of Physics, New York University

Dustin Lang
Department of Computer Science, University of Toronto
Princeton University Observatory

https://arxiv.org/pdf/1008.4686.pdf

cedure that could reasonably have generated the data set you have. In the case of the straight line fit in the presence of known, Gaussian uncertainties in one dimension, one can create this generative model as follows: Imagine that the data *really do* come from a line of the form $y = f(x) = m\,x + b$, and that the only reason that any data point deviates from this perfect, narrow, straight line is that to each of the true y values a small y-direction offset has been added, where that offset was drawn from a Gaussian distribution of zero mean and known variance σ_y^2. In this model, given an independent position x_i, an uncertainty σ_{yi}, a slope m, and an intercept b, the frequency distribution $p(y_i|x_i, \sigma_{yi}, m, b)$ for y_i is

$$p(y_i|x_i, \sigma_{yi}, m, b) = \frac{1}{\sqrt{2\,\pi\,\sigma_{yi}^2}}\,\exp\left(-\frac{[y_i - m\,x_i - b]^2}{2\,\sigma_{yi}^2}\right) \quad , \tag{9}$$

where this gives the expected frequency (in a hypothetical set of repeated experiments[13]) of getting a value in the infinitesimal range $[y_i, y_i + \mathrm{d}y]$ per unit $\mathrm{d}y$.

The generative model provides us with a natural, justified, scalar objective: We seek the line (parameters m and b) that maximize the probability of the observed data given the model or (in standard parlance) the *likelihood of the parameters.*[14] In our generative model the data points are independently drawn (implicitly), so the likelihood $\mathscr{L}$ is the product of conditional probabilities

$$\mathscr{L} = \prod_{i=1}^{N} p(y_i|x_i, \sigma_{yi}, m, b) \quad . \tag{10}$$

Data analysis recipes:
Fitting a model to data*

David W. Hogg
Center for Cosmology and Particle Physics, Department of Physics, New York University
Max-Planck-Institut für Astronomie, Heidelberg

Jo Bovy
Center for Cosmology and Particle Physics, Department of Physics, New York University

Dustin Lang
Department of Computer Science, University of Toronto
Princeton University Observatory

Gaussian

$$N \sim \frac{1}{\sqrt{2\pi}\sigma} exp \frac{-(x-\mu)^2}{2\sigma^2}$$

https://arxiv.org/pdf/1008.4686.pdf

In the case of the straight line fit in the presence of known, Gaussian uncertainties in one dimension, one can create this generative model as follows: Imagine that the data *really do* come from a line of the form $y = f(x) = m\,x + b$, and that the only reason that any data point deviates from this perfect, narrow, straight line is that to each of the true y values a small y-direction offset has been added, where that offset was drawn from a Gaussian distribution of zero mean and known variance σ_y^2. In this model, given an independent position x_i, an uncertainty σ_{yi}, a slope m, and an intercept b, the frequency distribution $p(y_i|x_i, \sigma_{yi}, m, b)$ for y_i is

$$p(y_i|x_i, \sigma_{yi}, m, b) = \frac{1}{\sqrt{2\,\pi\,\sigma_{yi}^2}} \exp\left(-\frac{[y_i - m\,x_i - b]^2}{2\,\sigma_{yi}^2}\right) \quad , \tag{9}$$

where this gives the expected frequency (in a hypothetical set of repeated experiments[13]) of getting a value in the infinitesimal range $[y_i, y_i + \mathrm{d}y]$ per unit $\mathrm{d}y$.

The generative model provides us with a natural, justified, scalar objective: We seek the line (parameters m and b) that maximize the probability of the observed data given the model or (in standard parlance) the *likelihood of the parameters.*[14] In our generative model the data points are independently drawn (implicitly), so the likelihood $\mathscr{L}$ is the product of conditional probabilities

$$\mathscr{L} = \prod_{i=1}^{N} p(y_i|x_i, \sigma_{yi}, m, b) \quad . \tag{10}$$

likelihood, probability, and objective functions

$$L(m, b|\vec{y}) = \prod_i^N p_i(y_i|x_i, m, b)$$

$$L(m, b|\vec{y}) = \prod_i^N p_i(y_i|x_i, \sigma_i, m, b)$$

$$p(y_i) = \frac{1}{\sigma_i\sqrt{2\pi}} \exp - \frac{(y_i - (mx_i + b))^2}{2\sigma_i^2}$$

$$\log a \cdot b = \log a + \log b$$

$$\ln L(m, b|\vec{y}) = \ln \prod_i^N \frac{1}{\sigma_i\sqrt{2\pi}} \exp - \frac{y_i - (mx_i + b)}{2\sigma_i^2}$$

likelihood, probability, and objective functions

$$L(m, b|\vec{y}) = \prod_i^N p_i(y_i|x_i, m, b)$$

$$L(m, b|\vec{y}) = \prod_i^N p_i(y_i|x_i, \sigma_i, m, b)$$

$$p(y_i) = \frac{1}{\sigma_i\sqrt{2\pi}} \exp - \frac{(y_i - (mx_i + b))^2}{2\sigma_i^2}$$

$$x^a \cdot x^b = x^{(a+b)}$$

$$\ln L(m, b|\vec{y}) = \ln \prod \frac{1}{\sigma_i\sqrt{2\pi}} + \ln\left(\prod_i^N e^{-\frac{y_i - (mx_i + b)}{2\sigma_i^2}}\right)$$

likelihood, probability, and objective functions

$$L(m, b|\vec{y}) = \prod_i^N p_i(y_i|x_i, m, b)$$

$$L(m, b|\vec{y}) = \prod_i^N p_i(y_i|x_i, \sigma_i, m, b)$$

$$p(y_i) = \frac{1}{\sigma_i\sqrt{2\pi}} \exp - \frac{(y_i - (mx_i + b))^2}{2\sigma_i^2}$$

$$\ln L(m, b|\vec{y}) = \ln \prod \frac{1}{\sigma_i\sqrt{2\pi}} + \ln \left(e^{-\sum_i^N \frac{y_i - (mx_i + b)}{2\sigma_i^2}} \right)$$

likelihood, probability, and objective functions

$$L(m, b|\vec{y}) = \prod_i^N p_i(y_i|x_i, m, b)$$

$$L(m, b|\vec{y}) = \prod_i^N p_i(y_i|x_i, \sigma_i, m, b)$$

$$p(y_i) = \frac{1}{\sigma_i\sqrt{2\pi}} \exp - \frac{(y_i - (mx_i + b))^2}{2\sigma_i^2}$$

σ_i not part of the model

$$\ln L(m, b|\vec{y}) = \ln \prod \frac{1}{\sigma_i\sqrt{2\pi}} + \ln\left(e^{-\sum_i^N \frac{y_i - (mx_i + b)}{2\sigma_i^2}}\right)$$

likelihood, probability, and objective functions

$$L(m, b|\vec{y}) = \prod_i^N p_i(y_i|x_i, m, b)$$

$$L(m, b|\vec{y}) = \prod_i^N p_i(y_i|x_i, \sigma_i, m, b)$$

$$p(y_i) = \frac{1}{\sigma_i\sqrt{2\pi}} \exp - \frac{(y_i - (mx_i + b))^2}{2\sigma_i^2}$$

$$\ln L(m, b|\vec{y}) = K - \sum \frac{(y_i - (mx_i + b))^2}{2\sigma_i^2}$$

likelihood, probability, and objective functions

$$L(m, b|\vec{y}) = \prod_i^N p_i(y_i|x_i, m, b)$$

$$L(m, b|\vec{y}) = \prod_i^N p_i(y_i|x_i, \sigma_i, m, b)$$

$$p(y_i) = \frac{1}{\sigma_i\sqrt{2\pi}} \exp - \frac{(y_i - (mx_i + b))^2}{2\sigma_i^2}$$

$$\ln L(m, b|\vec{y}) = K - \sum \frac{(y_i - (mx_i + b))^2}{2\sigma_i^2} = K - \frac{1}{2}\chi^2$$

All problems through Section 2 of Hogg&Bovey&Lang

https://arxiv.org/pdf/1008.4686.pdf

hewat

https://arxiv.org/pdf/1008.4686.pdf

read section 1 & 2

Data analysis recipes: Fitting a model to data*

David W. Hogg
Center for Cosmology and Particle Physics, Department of Physics, New York University
Max-Planck-Institut für Astronomie, Heidelberg

Jo Bovy
Center for Cosmology and Particle Physics, Department of Physics, New York University

Dustin Lang
Department of Computer Science, University of Toronto
Princeton University Observatory

reading

http://www-sop.inria.fr/members/Ian.Jermyn/philosophy/writings/Boxonmaths.pdf

optional

its long,

but you have 2 weeks!

Science and Statistics

GEORGE E. P. BOX*

Aspects of scientific method are discussed: In particular, its representation as a motivated iteration in which, in succession, practice confronts theory, and theory, practice. Rapid progress requires sufficient flexibility to profit from such confrontations, and the ability to devise parsimonious but effective models, to worry selectively about model inadequacies and to employ mathematics skillfully but appropriately. The development of statistical methods at Rothamsted Experimental Station by Sir Ronald Fisher is used to illustrate these themes.

1. INTRODUCTION

In 1952, when presenting R.A. Fisher for the Honorary degree of Doctor of Science at the University of Chicago, W. Allen Wallis described him in these words.

> He has made contributions to many areas of science; among them are agronomy, anthropology, astronomy, bacteriology, botany, economics, forestry, meteorology, psychology, public health, and—above all—genetics, in which he is recognized as one of the leaders. Out of this varied scientific research and his skill in mathematics, he has evolved systematic principles for

on the one hand, nor by the undirected accumulation of practical facts on the other, but rather by a motivated *iteration* between theory and practice such as is illustrated in Figure A(1).

A. The Advancement of Learning

A(1) An Iteration Between Theory and Practice

A(2) A Feedback Loop

https://www.slac.stanford.edu/econf/C030908/papers/TUAT004.pdf

PHYSTAT2003, SLAC, Stanford, California, September 8-11, 2003

Definition and Treatment of Systematic Uncertainties in High Energy Physics and Astrophysics

Pekka K. Sinervo
Department of Physics, University of Toronto, Toronto, ON M5S 1A7, CANADA

Systematic uncertainties in high energy physics and astrophysics are often significant contributions to the overall uncertainty in a measurement, in many cases being comparable to the statistical uncertainties. However, consistent definition and practice is elusive, as there are few formal definitions and there exists significant ambiguity in what is defined as a systematic and statistical uncertainty in a given analysis. I will describe current practice, and recommend a definition and classification of systematic uncertainties that allows one to treat these sources of uncertainty in a consistent and robust fashion. Classical and Bayesian approaches will be contrasted.

references

1. INTRODUCTION TO SYSTEMATIC UNCERTAINTIES

Most measurements of physical quantities in high energy physics and astrophysics involve both a statistical uncertainty and an additional "systematic" uncertainty. Systematic uncertainties play a key role in

include uncertainties that arise from the calibration of the measurement device, the probability of detection of a given type of interaction (often called the "acceptance" of the detector), and parameters of the model used to make inferences that themselves are not precisely known. The definition of such uncertainties is often ad hoc in a given measurement, and there are

2.3 Optimization schemes

Optimization

Gradient descent is an iterative algorithm, that starts from a random point on a function and travels down its slope in steps until it reaches the lowest point of that function.

(*Toward data science*)

Univariate Linear regression $y_i = (a\, x_i + b)$

Target Funcion $\chi^2 = \sum_{i=1}^{N} \left(\frac{(y_i - (a\, x_i + b))^2}{\sigma_i^2} \right)$

~~Stochastic~~ Gradient Descent Optimization

1. choose initial value for *a* $a = 1$

~~Stochastic~~ Gradient Descent Optimization

1. choose initial value for *a*
2. calculate the X2

$$\chi^2 = 71$$

~~Stochastic~~ Gradient Descent Optimization

1. choose initial value for *a*
2. calculate the X2
3. calculate best direction to go to decrease the X2

~~Stochastic~~ Gradient Descent Optimization

1. choose initial value for *a*
2. calculate the X2
3. calculate best direction to go to decrease the X2

$$\frac{d\chi^2}{da} > 0$$

~~Stochastic~~ Gradient Descent Optimization

1. choose initial value for *a*
2. calculate the X2
3. calculate best direction to go to decrease the X2

$$\frac{d\chi^2}{da} < 0$$

~~Stochastic~~ Gradient Descent Optimization

1. choose initial value for *a*
2. calculate the X2
3. calculate best direction to go to decrease the X2
4. step in that direction

$$\frac{d\chi^2}{da} < 0$$

~~Stochastic~~ Gradient Descent Optimization

1. choose initial value for *a*
2. calculate the X2
3. calculate best direction to go to decrease the X2
4. step in that direction
5. repeat 2-4 until both direction take you to larger gradient

$$\frac{d\chi^2}{da} < 0$$

~~Stochastic~~ Gradient Descent Optimization

1. choose initial value for *a*
2. calculate the X2
3. calculate best direction to go to decrease the X2
4. step in that direction
5. repeat 2-4 until both direction take you to larger gradient

$$\frac{d\chi^2}{da} < 0$$

~~Stochastic~~ Gradient Descent Optimization

1. choose initial value for *a*
2. calculate the X2
3. calculate best direction to go to decrease the X2
4. step in that direction
5. repeat 2-4 until both direction take you to larger gradient

$$\frac{d\chi^2}{da} < 0$$

~~Stochastic~~ Gradient Descent Optimization

1. choose initial value for *a*
2. calculate the X2
3. calculate best direction to go to decrease the X2
4. step in that direction
5. repeat 2-4 until both direction take you to larger gradient

$$\frac{d\chi^2}{da} < 0$$

~~Stochastic~~ Gradient Descent Optimization

1. choose initial value for *a*
2. calculate the X2
3. calculate best direction to go to decrease the X2
4. step in that direction
5. repeat 2-4 until both direction take you to larger gradient

$$\frac{d\chi^2}{da} < 0$$

~~Stochastic~~ Gradient Descent Optimization

1. choose initial value for *a*
2. calculate the X2
3. calculate best direction to go to decrease the X2
4. step in that direction
5. repeat 2-4 until both direction take you to larger gradient

$$\frac{d\chi^2}{da} < 0$$

~~Stochastic~~ Gradient Descent Optimization

Gradient Descent

$$p_{\text{new}} := p_{\text{old}} - \eta \nabla Q(p)$$

Univariate Linear regression $y_i = (a\, x_i)$

Target Funcion $\chi^2 = \sum_{i=1}^{N} \left(\frac{(y_i - (a\, x_i + b))^2}{\sigma_i^2} \right)$

~~Stochastic~~ Gradient Descent Optimization

Q : target function

p : parameters

η : learning rate

ϵ : tolerance

$$p_{\text{new}} := p_{\text{old}} - \eta \nabla Q(p)$$

"convergence" is reached when the gradient is

~0: with ε tollrance

$$\Delta Q(p_{\text{final}}) \in [-\epsilon, \epsilon]$$

gradient descent algorithm

1. Choose a target function ***Q(p)*** of the parameters ***p***

2. Choose a (random) initial value for the parameters: (e.g. $p_0 = (a_0, b_0)$)

3. Choose a learning rate ***η*** (this could be a multidimensional vector η_i setting a different learning rate for different features)

   Repeat steps 4, 5, 6 until "convergence":

4. Calculate the gradient ***Q'*** of the target function for the current parameter values ***calculating it over all observations in the training set***

5. Calculate the next step sizes for each feature :
   stepsize = ***Q'(p_now) * η***

6. Calculate the new parameters ***p_new*** as :
   p_new = p_now - stepsize

~~Stochastic~~ Gradient Descent

Optimization

Gradient Descent

$$p_{\text{new}} = p_{\text{old}} - \sum_j \eta_j \sum_{i=1}^{N} \frac{df'(x_{i,j})}{dx_{i,j}}$$

Univariate Linear regression $y_i = (a\, x_i + b)$

Target Funcion $\chi^2 = \sum_{i=1}^{N} \left(\frac{(y_i - (a\, x_i + b))^2}{\sigma_i^2} \right)$

~~Stochastic~~ Gradient Descent Optimization

Target Funcion $\chi^2 = \sum_{i=1}^{N} \left(\frac{(y_i - (a\, x_i + b))^2}{\sigma_i^2} \right)$

Stochastic Gradient Descent Optimization

Add stochasticity to avoid getting stuck in a local minimum

Target Funcion $\chi^2 = \sum_{i=1}^{N} \left(\frac{(y_i - (a\, x_i + b))^2}{\sigma_i^2} \right)$

Stochastic Gradient Descent

<u>Stochastic</u> Gradient Descent (SGD): use a different (random) sub-sample of the data at each iteration

https://datascience.stackexchange.com/questions/52884/possible-for-batch-size-of-neural-network-to-be-too-small

$$Q: \text{ target function}$$
$$p: \text{ parameters}$$
$$\eta: \text{ learning rate}$$
$$\epsilon: \text{ tolerance}$$

$$p_{\text{new}} := p_{\text{old}} - \eta \nabla Q(p)$$

"convergence" is reached when the gradient is ~0: with ε tollrance

$$\Delta Q(p_{\text{final}}) \in [-\epsilon, \epsilon]$$

Stochastic Gradient Descent Optimization

Q : target function

p : parameters

η : learning rate

ϵ : tolerance

$$p_{\text{new}} := p_{\text{old}} - \eta \nabla Q(p)$$

"convergence" is reached when the gradient is

~0: with ε tollrance

$$\Delta Q(p_{\text{final}}) \in [-\epsilon, \epsilon]$$

stochastic gradient descent algorithm

1. Choose a target function ***Q(p)*** of the parameters ***p***

2. Choose a (random) initial value for the parameters: (e.g. $p_0 = (a_0, b_0)$)

3. Choose a learning rate ***η*** (this could be a multidimensional vector η_i setting a different learning rate for different features)

   Repeat steps 4, 5, 6 until "convergence":

4. Calculate the gradient ***Q'*** of the target function for the current parameter values ***on a subset of the observations (extreme: size=1)***

5. Calculate the next step sizes for each feature :
   stepsize = ***Q'(p_now) * η***

6. Calculate the new parameters ***p_new*** as :
   p_new = p_now - stepsize

Stochastic Gradient Descent Optimization

gradient descent algorithm

1. Choose a target function ***Q(p)*** of the parameters ***p***

2. Choose a (random) initial value for the parameters: (e.g. $p_0 = (a_0, b_0)$)

3. Choose a learning rate ***η*** (this could be a multidimensional vector η_i setting a different learning rate for different features)

   Repeat steps 4, 5, 6 until "convergence":

4. Calculate the gradient ***Q'*** of the target function for the current parameter values ***calculating it over all observations in the training set***

5. Calculate the next step sizes for each feature :
   stepsize = ***Q'(p_now) * η***

6. Calculate the new parameters ***p_new*** as :
   p_new = p_now - stepsize

stochastic gradient descent algorithm

1. Choose a target function ***Q(p)*** of the parameters ***p***

2. Choose a (random) initial value for the parameters: (e.g. $p_0 = (a_0, b_0)$)

3. Choose a learning rate ***η*** (this could be a multidimensional vector η_i setting a different learning rate for different features)

   Repeat steps 4, 5, 6 until "convergence":

4. Calculate the gradient ***Q'*** of the target function for the current parameter values ***on a subset of the observations (extreme: size=1)***

5. Calculate the next step sizes for each feature :
   stepsize = ***Q'(p_now) * η***

6. Calculate the new parameters ***p_new*** as :
   p_new = p_now - stepsize

Stochastic Gradient Descent Optimization

Stochastic Gradient Descent

$$p_{\text{new}} = p_{\text{old}} - \sum_j \eta_j \frac{df'(x_{i,j})}{dx_{i,j}} \quad i \in N$$

where *i* element of the full N-dimensional observation set (a subset)

Univariate Linear regression $y_i = (a\, x_i)$

Target Funcion $\chi^2 = \sum_{i=1}^{N} \left(\frac{(y_i - (a\, x_i + b))^2}{\sigma_i^2} \right)$

Stochastic Gradient Descent Optimization

idea 2. start a bunch of parallel optimizations

Univariate Linear regression $y_i = (a\ x_i)$

Target Funcion $\chi^2 = \sum_{i=1}^{N} \left(\frac{(y_i - (a\ x_i + b))^2}{\sigma_i^2} \right)$

3 Model Selection principles

what model should I choose?

No matter what anyone tells you an answer to this question cannot be given in the abstract case: it is a domain specific question!

except:

the principle of parsimony

the principle of parsimony

or Ockham's razor

Pluralitas non est ponenda sine neccesitate

William of Ockham (logician and Franciscan friar) 1300ca
but probably to be attributed to John Duns Scotus (1265–1308)

“Complexity needs not to be postulated without a need for it”

the principle of parsimony

the earth is round,

and it orbits around the sun

Geocentric models are intuitive:

from our perspective we see the Sun moving, while we stay still

Peter Apian, *Cosmographia*, Antwerp, 1524 from Edward Grant, "Celestial Orbs in the Latin Middle Ages", *Isis*, Vol. 78, No. 2. (Jun., 1987).

the principle of parsimony

Encyclopaedia Brittanica 1st Edition

Dr Long's copy of Cassini, 1777

the earth is round,

and it orbits around the sun

As observations improve

this model can no longer fit the data!

not easily anyways...

the principle of parsimony

the earth is round,

~~and it orbits around the sun~~

A new model that is much simpler fit the data just as well

(perhaps though only until better data comes...)

Heliocentric model from Nicolaus Copernicus' *De revolutionibus orbium coelestium*.

the principle of parsimony

or Ockham's razor

Pluralitas non est ponenda sine neccesitate

William of Ockham (logician and Franciscan friar) 1300ca
but probably to be attributed to John Duns Scotus (1265–1308)

“Complexity needs not to be postulated without a need for it”

the principle of parsimony

or Ockham's razor

Pluralitas non est ponenda sine neccesitate

William of Ockham (logician and Franciscan friar) 1300ca
but probably to be attributed to John Duns Scotus (1265–1308)

“Complexity needs not to be postulated without a need for it”
“Between 2 theories that perform similarly choose the ***simpler one***”

the principle of parsimony

or Ockham's razor

Pluralitas non est ponenda sine neccesitate

William of Ockham (logician and Franciscan friar) 1300ca
but probably to be attributed to John Duns Scotus (1265–1308)

“Complexity needs not to be postulated without a need for it”
“Between 2 theories that perform similarly choose the ***one with fewer parameters***”

the principle of parsimony

Science and Statistics George E. P. Box (1976)

Journal of the American Statistical Association, Vol. 71, No. 356, pp. 791-799.

Since all models are wrong the scientist cannot obtain a "correct" one by excessive elaboration. On the contrary following William of Occam he should seek an economical description of natural phenomena

Since all models are wrong the scientist must be alert to what is importantly wrong.

32

Model Selection methods

HOW DO I CHOOSE A MODEL?

Given two models which is preferable?

A *rigorous* answer (in terms of NHST) can be obtained for ***2 nested* models**

This directly answers the question:

"is my more complex model *overfitting* the data?"

The LR statistics is expected to follow a χ^2 distrbution under the *Null Hypothesis* that the ***simpler model is preferable***

NESTED MODELS : one model contains the other one, e.g.

y= mx + l

is contained in

*y=ax**2+ mx + l*

Likelihood-ratio tests

likelihood ratio statistics LR

$$LR = -2log_e \frac{L(\text{complex model})}{L(\text{simple model})}$$

statsmodels.model.compare_lr_ratio()

HOW DO I CHOOSE A MODEL?

Given two models which is preferable?

Likelihood-ratio tests

likelihood ratio statistics LR

$$LR = -2log_e \frac{L(\text{complex model})}{L(\text{simple model})}$$

statsmodels.model.compare_lr_ratio()

The LR statistics is expected to follow a χ^2 distrbution under the *Null Hypothesis* that the ***simpler model is preferable***

The table below gives the value x_0^2 for which $P[x^2 < x_0^2] = P$ for a given number of degrees of freedom and a given value of P.

Degrees of Freedom	Values of P									
	0.005	0.010	0.025	0.050	0.100	0.900	0.950	0.975	0.990	0.995
1	---	---	0.001	0.004	0.016	2.706	3.841	5.024	6.635	7.879
2	0.01	0.020	0.051	0.103	0.211	4.605	5.991	7.378	9.210	10.597
3	0.072	0.115	0.216	0.352	0.584	6.251	7.815	9.348	11.345	12.838
4	0.207	0.297	0.484	0.711	1.064	7.779	9.488	11.143	13.277	14.860
5	0.412	0.554	0.831	1.145	1.610	9.236	11.070	12.833	15.086	16.750
6	0.676	0.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812	18.548
7	0.989	1.239	1.690	2.167	2.833	12.017	14.067	16.013	18.475	20.278
8	1.344	1.646	2.180	2.733	3.490	13.362	15.507	17.535	20.090	21.955
9	1.735	2.088	2.700	3.325	4.168	14.684	16.919	19.023	21.666	23.589
10	2.156	2.558	3.247	3.940	4.865	15.987	18.307	20.483	23.209	25.188
11	2.603	3.053	3.816	4.575	5.578	17.275	19.675	21.920	24.725	26.757
12	3.074	3.571	4.404	5.226	6.304	18.549	21.026	23.337	26.217	28.300
13	3.565	4.107	5.009	5.892	7.042	19.812	22.362	24.736	27.688	29.819
14	4.075	4.660	5.629	6.571	7.790	21.064	23.685	26.119	29.141	31.319
15	4.601	5.229	6.262	7.261	8.547	22.307	24.996	27.488	30.578	32.801
16	5.142	5.812	6.908	7.962	9.312	23.542	26.296	28.845	32.000	34.267
17	5.697	6.408	7.564	8.672	10.085	24.769	27.587	30.191	33.409	35.718
18	6.265	7.015	8.231	9.390	10.865	25.989	28.869	31.526	34.805	37.156
19	6.844	7.633	8.907	10.117	11.651	27.204	30.144	32.852	36.191	38.582
20	7.434	8.260	9.591	10.851	12.443	28.412	31.410	34.170	37.566	39.997

MLTSA: model selection

model selection is also based on the minimization of a quantity. Several quantities are suitable:

AIC Optimism and likelihood maximization on the training set

BIC Bayese theorem

MLD Shannon 1948: A Mathematical Theory of Communication

a theory to find fundamental limits on signal processing and communication operations such as data compression

MLTSA:

AIC, BIC, & MDL

Likelihood: **Model Performance**.

$$\mathrm{AIC} = -\frac{2}{N}\log(L) + \frac{2}{N}k$$

number of parameters:

Model Complexity

Akaike information criterion (AIC) .

Based on $\lim_{N\to\infty}(-2E(\log Pr_{\hat{\theta}}(Y))) = -\frac{2}{N}E\ \log(L) + d\frac{2}{N}$

where $Pr_{\hat{\theta}}(Y)$ is a family of function (=densities) containing the correct (=true) function and $\hat{\theta}$ is the set of parameters that maximized the likelihood *L*

***L* is the likelihood of the data, *k* is the number of parameters,**

N the number of variables.

MLTSA:

AIC, BIC, & MDL

Likelihood: **Model Performance**.

$$\mathrm{AIC} = -\frac{2}{N}\log(L) + \frac{2}{N}k$$

number of parameters:

Model Complexity

Akaike information criterion (AIC) .

Based on $\lim_{N\to\infty}(-2E(\log Pr_{\hat{\theta}}(Y))) = -\frac{2}{N}E\ \log(L) + d\frac{2}{N}$

where $Pr_{\hat{\theta}}(Y)$ is a family of function (=densities) containing the correct (=true) function and $\hat{\theta}$ is the set of parameters that maximized the likelihood *L*

***L* is the likelihood of the data, *k* is the number of parameters,**

N the number of variables.

"-" sign in front of the log-likelihood: AIC shrinks for better models,

AIC ~ k => is linearly proportional to the number of parameters

MLTSA:

AIC, BIC, & MDL

Likelihood: **Model Performance**.

$$\mathrm{BIC} = -2\log(L) + \log(N)k$$

number of parameters:

Model Complexity

Bayesian information criterion (BIC) .

***L* is the likelihood of the data, *k* is the number of parameters,**

***N the number of variables*.**

stronger penalization of complexity (as long as N>e^2)

Bayes Factor

The derivation is very different:

$$\frac{P(M_m|D)}{P(M_l|D)} = \frac{P(M_m)}{P(M_l)} \cdot \frac{P(D|M_m)}{P(D|M_l)}$$

MLTSA:
AIC, BIC, & MDL

$$\mathrm{MDL} = -\log(L(\theta)) - \log(L(y|X,\theta))$$

Minimum Description Length (MDL) .

negative log-likelihood of the model parameters (*θ*) and the negative log-likelihood of the target values (*y*) given the input values (*X*) and the model parameters (*θ*).

also: log(L(θ)): number of bits required to represent the model,

log(L(y| X,θ)): number of bits required to represent the predictions on observations

minimize the encoding of the model and its predictions

derived from Shannon's theorem of information

MLTSA:

AIC, BIC, & MDL

$$\mathrm{AIC} = -\frac{2}{N}\log(L) + \frac{2}{N}k$$

$$\mathrm{BIC} = -2\log(L) + \log(N)k$$

$$\mathrm{MDL} = -\log(L(\theta)) - \log(L(y|X,\theta))$$

Mathematically similar, though derived from different approaches. All used the same way: the preferred model is the model that minimized the estimator

implementation https://machinelearningmastery.com/probabilistic-model-selection-measures/

MLTSA: AIC, BIC, & MDL

$$\mathrm{AIC} = -\frac{2}{N}\log(L) + \frac{2}{N}k$$

$$\mathrm{BIC} = -2\log(L) + \log(N)k$$

$$\mathrm{MDL} = -\log(L(\theta)) - \log(L(y|X,\theta))$$

4 Model Optimization with MCMC

hospitalized DE
hospitalized
600
500
400
300
200
100
0
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01
2021-03
2021-05
2021-07

400
200
0
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01
2021-03
2021-05
2021-07

fracsymp_hospitalized
sociald1
sociald2
sociald3
sociald4
sociald5
sociald6
sociald7
sociald8
cut

Monte Carlo Markov Chain

stochastic

"markovian"

sequence

MCMC

$$P(\theta|D,f) \propto P(D|\theta,f)P(\theta,f)$$

Goal: sample the posterior distribution

posterior: joint probability distributin of a parameter set (*m, b*) conditioned upon some data *D* and a model hypothesys *f*

MCMC

Goal: sample the posterior distribution

```
choose a starting point in the parameter space: current = θ0 = (m0,b0)
WHILE convergence criterion is met:
    calculate the current posterior pcurr = P(D|θ0,f)
    //proposal
    choose a new set of parameters new = θnew = (mnew,bnew)
    calculate the proposed posterior pnew = P(D|θnew,f)
    IF pnew/pcurr > 1:
        current = new
    ELSE:
        //probabilistic step: accept with probability pnew/pcurr
        draw a random number r ϵ U[0,1]
        IF pnew/pcurr > r:
            current = new
        ELSE:
            pass // do nothing
```

stochasticity allows us to explore the whole surface but spend more time in interesting spots

MCMC

Goal: sample the posterior distribution

choose a starting point in the parameter space: ***current*** $= \boldsymbol{\theta_0} = (m_0, b_0)$

WHILE convergence criterion is met:

calculate the current posterior $p_{curr} = P(D|\theta_0, f)$

//proposal

choose a new set of parameters ***new*** $= \boldsymbol{\theta_{new}} = (m_{new}, b_{new})$

calculate the proposed posterior $p_{new} = P(D|\theta_{new}, f)$

IF $p_{new}/p_{curr} > 1$:

current = new

ELSE:

MCMC

```
choose a starting point in the parameter space: current = θ0 = (m0,b0)
WHILE convergence criterion is met:
    calculate the current posterior pcurr = P(D|θ0,f)
    //proposal
    choose a new set of parameters new = θnew = (mnew,bnew)
    calculate the proposed posterior pnew = P(D|θnew,f)
    IF pnew/pcurr > 1:
        current = new
    ELSE:
        //probabilistic step: accept with probability pnew/pcurr
        draw a random number r ϵ U[0,1]
        IF pnew/pcurr > r:
            current = new
        ELSE:
            pass // do nothing
```

Questions:

1. how do I choose the next point?

Any *Markovian* e*rgodic* process

A Markovian Process

A process is Markovian if the next state of the system is determined stochastically as a perturbation of the current state of the system, and *only* the current state of the system, i.e. the system has no memory of earlier states (a *memory-less* process).

Definition

Ergodic Process

(given enough time) the entire parameter space would be sampled.

Detailed Balance is a sufficient condition
for ergodicity
Metropolis Rosenbluth Rosenbluth Teller 1953 - Hastings 1970

At equilibrium, each elementary process should be equilibrated by its reverse process.

reversible Markov process

$$\pi(x_1)P(x_2|x_1) = \pi(x_2)P(x_1|x_2)$$

Definition

it can be shown that

If the chains are a Markovian Ergodic process

the algorithm is guaranteed to explore the entire likelihood surface given infinite time

This is in contrast to gradient descent, which can get stuck in local minima or in local saddle points.

MCMC

The chains: *the algorithm creates a "chain" (a random walk) that "explores" the likelihood surface.*

More efficient is to run many parallel chains - each exploring the surface, an "ensemble"

The path of the chains can be shown along each feature

choose a starting point in the parameter space: ***current*** $= \boldsymbol{\theta_0} = (m_0, b_0)$

WHILE convergence criterion is met:

calculate the current posterior $p_{curr} = P(D|\theta_0, f)$

//proposal

choose a new set of parameters ***new*** $= \boldsymbol{\theta_{new}} = (m_{new}, b_{new})$

calculate the proposed posterior $p_{new} = P(D|\theta_{new}, f)$

IF $p_{new}/p_{curr} > 1$:

current = new

ELSE:

//probabilistic step: accept with probability p_{new}/p_{curr}

draw a random number $r \in U[0,1]$

IF $p_{new}/p_{curr} > r$:

current = new

ELSE:

pass // do nothing

MCMC

how to choose the next point

how you make this decision names the algorithm

simulated annealing (good for multimodal)

parallel tempering (good for multimodal)

differential evolution (good for covariant spaces)

Gibbs sampling (move in along one variable at a time)

choose a starting point in the parameter space: ***current*** $= \boldsymbol{\theta_0} = (m_0, b_0)$

WHILE convergence criterion is met:

calculate the current posterior $p_{curr} = P(D|\theta_0, f)$

//proposal

choose a new set of parameters ***new*** $= \boldsymbol{\theta_{new}} = (m_{new}, b_{new})$

calculate the proposed posterior $p_{new} = P(D|\theta_{new}, f)$

IF $p_{new}/p_{curr} > 1$:

current = new

ELSE:

//probabilistic step: accept with probability p_{new}/p_{curr}

draw a random number r $\in U[0,1]$

IF $p_{new}/p_{curr} > r$:

current = new

ELSE:

pass // do nothing

MCMC

how to choose the next point

how you make this decision names the algorithm

simulated annealing (good for multimodal)

parallel tempering (good for multimodal)

differential evolution (good for covariant spaces)

Gibbs sampling (move in along one variable at a time)

Annealing Markov Chain Monte Carlo With Applications to Ancestral Inference

Charles J. Geyer and Elizabeth A. Thompson*

https://www.jstor.org/stable/2291325?seq=1

Markov chain Monte Carlo (MCMC) in the form of the Metropolis–Hastings algorithm (Hastings 1970; Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller 1953) and its special case the Gibbs sampler (Geman and Geman 1984) has been used in recent years to attack a wide variety of intractable statistical problems. (See, for example, Besag and Green 1993, Geyer 1992, Geyer and Thompson 1992, Smith and Roberts 1993, Tierney 1994, and the accompanying discussions and references.) MCMC simulates realizations from probability distributions whose densities are known up to a normalizing factor. If $h(x)$ is a nonnegative integrable function on the sample space, then the Metropolis–Hastings algorithm simulates a Markov chain whose equilibrium distribution is proportional to $h(x)$ using only evaluations of $h(x)$.

If the chain is irreducible, then time averages over the chain converge to expectations with respect to the stationary distribution as the Monte Carlo sample size goes to infinity; but if the chain is slowly mixing, then it may take astronomically large sample sizes to get accurate estimates. Slow mixing typically occurs in problems where the sample space has high dimension. For samplers that update one variable at a time, like the Gibbs sampler, the mixing time can be exponential in the number of variables. Thus, to do MCMC on high-

MCMC

MCMC animations
https://www.youtube.com/watch?v=J6FrNf5__G0&list=PLgArfv_fOU5dwjeP_57NO_jnRJ7cWCe6J

Examples of how to choose the next point

affine invariant : EMCEE package

choose a starting point in the parameter space: ***current*** $= \boldsymbol{\theta_0} = (m_0, b_0)$

WHILE convergence criterion is met:

calculate the current posterior $p_{curr} = P(D|\theta_0, f)$

//proposal

choose a new set of parameters ***new*** $= \boldsymbol{\theta_{new}} = (m_{new}, b_{new})$

calculate the proposed posterior $p_{new} = P(D|\theta_{new}, f)$

IF $p_{new}/p_{curr} > 1$:

current = new

ELSE:

//probabilistic step: accept with probability p_{new}/p_{curr}

draw a random number $r \in U[0,1]$

IF $p_{new}/p_{curr} > r$:

current = new

ELSE:

pass // do nothing

MCMC convergence

Goal: sample the posterior distribution

Questions:

1. how do I choose the next point?

2. when have I sampled the posterior adequatly?
   has your MCMC *converged* ?

MCMC convergence

Goal: sample the posterior distribution

Questions:

1. how do I choose the next point?
2. when have I sampled the posterior adequatly?
   has your MCMC *converged*?

1. **check autocorrelation within a chain (*Raftery*)**
2. check that all chains coverged to same region (a stationary distribution *GelmanRubin*)
3. mean at beginning = mean at end (*Geweke*)
4. check that entire chain reached stationary distribution (or a final fraction of the chain, *Heidelberg-Welch* using Cramer-von-Mises statistic)

MCMC convergence

Goal: sample the posterior distribution

Questions:

1. how do I choose the next point?
2. when have I sampled the posterior adequatly?
   has your MCMC *converged*?

1. check autocorrelation within a chain (*Raftery*)
2. **check that all chains coverged to same region (a stationary distribution *GelmanRubin*)**
3. mean at beginning = mean at end (*Geweke*)
4. check that entire chain reached stationary distribution (or a final fraction of the chain, *Heidelberg-Welch* using Cramer-von-Mises statistic)

MCMC convergence

Goal: sample the posterior distribution

Questions:

1. how do I choose the next point?
2. when have I sampled the posterior adequatly?
   has your MCMC *converged*?
3. how can it be-the samples are *not independent!*

good point!…

1. check autocorrelation within a chain (*Raftery*)
2. check that all chains coverged to same region (a stationary distribution *GelmanRubin*)
3. mean at beginning = mean at end (*Geweke*)
4. check that entire chain reached stationary distribution (or a final fraction of the chain, *Heidelberg-Welch* using Cramer-von-Mises statistic)

MCMC convergence

Goal: sample the posterior distribution

Questions:

1. how do I choose the next point?
2. when have I sampled the posterior adequatly?
   has your MCMC *converged*?

1. check autocorrelation within a chain (*Raftery*)
2. check that all chains coverged to same region (a stationary distribution *GelmanRubin*)
3. **mean at beginning = mean at end (*Geweke*)**
4. **check that entire chain reached stationary distribution (or a final fraction of the chain, *Heidelberg-Welch* using Cramer-von-Mises statistic)**