Introduction to Inverse Problems

1. *Simple regression* – This exercise is available as a Jupyter notebook.

2. *Nature of inverse problems* – Below are descriptions of inverse problems. Use what you have learned to decide for each case, i) what are the data and what are the unknowns, ii) whether each is a continuous or discrete inverse problem, and iii) whether each is a linear or nonlinear inverse problem. For the latter you could try and prove your answer using the tests of superposition and scaling:

   (i) The travel time t_i of a sound wave following path i through a medium can be described by

   $$t_i = \int_{path} \frac{1}{v(\mathbf{r})} ds$$

   where $v(\mathbf{r})$ is the sound velocity of the medium at location $\mathbf{r}$ and s is the distance along the wave path. We want to determine $v(\mathbf{r})$ of the medium from measuring t for many ray paths.

   (ii) We want to find a straight line that predicts temperature increase with depth at the site of a planned tunnel. For this purpose we measure temperature T_i at n depths z_i in a borehole, where $1 < i < n$. We use the following relation:

   $$T(z) = m_1 + m_2 z$$

   (iii) We want to predict rock density ρ in the Earth at a given radius r from its center from the known mass M and moment of inertia I of the Earth. We use the following relation:

   $$g(r) = \int_0^a g_i(r)\rho(r)dr$$

   where $d_1 = M$ and $d_2 = I$ and $g_i(r)$ are the corresponding Frechet kernels: $g_1(r) = 4\pi r^2$ and $g_2(r) = \frac{8}{3}\pi r^4$.

   (iv) We want to predict how much of a given rock mass, fraction X, in the Earth melts at a given value for dimensionless temperature T', where T' depends on the temperature T as well as on the solidus and liquidus of the rock. The solidus and liquidus of a rock depend on the pressure. For this purpose we measure melt fraction at various combinations of temperature and pressure. We use the following relation:

   $$X(T') = \frac{1}{2} + T' + \left(T'^2 - \frac{1}{4}\right)(m_o + m_1 T' + m_2 T'^2)$$

3. *Discretising continuous problems* – As we saw in Exercise 1, the central assumption in a regression problem is that we can express an unknown function, $f(x)$, in terms of a set of N basis functions, $\phi_i(x)$

   $$f(x) = \sum_{i=1}^{N} m_i \phi_i(x)$$

   (a) By multiplying both sides of this expression by an arbitrary basis function, and integrating, obtain an expression for the model vector $\mathbf{m}$.

   (b) Suppose we have only two basis functions, $\phi_0(x) = 1$ and $\phi_1(x) = x$, both defined to exist only for $-1 \leq x \leq 1$. Demonstrate that your expression allows the model parameters to be recovered exactly in the case that $f(x) = mx + c$.

   (c) Using the same basis, compute the model parameters that would be recovered if $f(x) = ax^2 + bx + c$. Does the answer make sense? Why/why not?

   (d) In a regression problem, we do not know $f(x)$ and so cannot compute the integrals directly. Approximate these integrals by sums. Can you see how you might estimate the solution to a regression problem? What will control the accuracy of your solution?

Linear, over-determined problems

4. *Regression, revisited* – This exercise is available as a Jupyter notebook.

5. *Lines of best fit* – You wish to fit a straight line, $y = mx + c$, to a dataset $\{(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)\}$. Show that the best-fitting line has

$$m = \frac{\langle xy\rangle - \langle x\rangle\langle y\rangle}{\langle x^2\rangle - \langle x\rangle^2} \qquad \text{and} \qquad c = \frac{\langle x^2\rangle\langle y\rangle - \langle x\rangle\langle xy\rangle}{\langle x^2\rangle - \langle x\rangle^2}$$

where we have introduced the notation $\langle z\rangle = \frac{1}{N}\sum_i z_i$ to denote the mean value of some quantity z, as estimated using the dataset.

6. *Deriving the least-squares algorithm* – We define the misfit, $\phi(\mathbf{m}) = \|\mathbf{d} - \mathbf{Gm}\|_2^2$. By differentiating this expression with respect to an arbitrary model component, m_λ, show that $\mathbf{m} = (\mathbf{A^T A})^{-1}\mathbf{A^T d}$.

   Compare this with your answer to Exercise 3d.

   Hint: You will need to begin by expressing ϕ in index notation—that is, writing it in terms of sums over the individual components of $\mathbf{m}$, $\mathbf{d}$ and $\mathbf{G}$.

Goodness of fit, probability and statistics

7. *Performing goodness of fit* – This exercise is available as a Jupyter notebook.

8. *The multidimensional Gaussian* – An N-dimensional normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ has probability density function

$$\mathbb{P}(\mathbf{x}) = \mathcal{N}_N(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \left[(2\pi)^N |\boldsymbol{\Sigma}|\right]^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\mathbf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right]$$

   (a) By noting that $\mathbf{MM^{-1}} = \mathbf{I}$, show that

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{P} & \mathbf{Q} \\ \mathbf{R} & \mathbf{S} \end{pmatrix} \qquad \text{where} \qquad \begin{cases} \mathbf{P} & = \mathbf{A^{-1}} + \mathbf{A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1}} \\ \mathbf{Q} & = -\mathbf{A^{-1}B(D - CA^{-1}B)^{-1}} \\ \mathbf{R} & = -\mathbf{(D - CA^{-1}B)^{-1}CA^{-1}} \\ \mathbf{S} & = \mathbf{(D - CA^{-1}B)^{-1}} \end{cases}$$

   (b) By performing the same computations in a different order, derive the Woodbury matrix identity,

$$\mathbf{(A + BDC)^{-1}} = \mathbf{A^{-1}} - \mathbf{A^{-1}B}\left(\mathbf{D^{-1}} + \mathbf{CA^{-1}B}\right)^{-1}\mathbf{CA^{-1}},$$

   and the relationship

$$\mathbf{DC(A + BDC)^{-1}} = \mathbf{(D^{-1} + CA^{-1}B)}^{-1}\mathbf{CA^{-1}}.$$

   (c) If the vector $\mathbf{x}$ is partitioned into two arbitrary pieces, so that $\mathbf{x} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}$, with corresponding partitioning of $\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_\mathbf{u} \\ \boldsymbol{\mu}_\mathbf{v} \end{pmatrix}$ and $\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_\mathbf{u} & \boldsymbol{\Sigma}_\mathbf{uv} \\ \boldsymbol{\Sigma}_\mathbf{uv}^\mathbf{T} & \boldsymbol{\Sigma}_\mathbf{v} \end{pmatrix}$, write down an expression for $\mathbb{P}(\mathbf{x})$. Show that this can be factorised into the form

$$\mathbb{P}(\mathbf{x}) = \mathcal{N}_{N_u}(\mathbf{u}, \boldsymbol{\mu}_\mathbf{u}, \boldsymbol{\Sigma}_\mathbf{u}) \cdot \mathcal{N}_{N_v}\left(\mathbf{v}, \boldsymbol{\mu}_\mathbf{v} + \boldsymbol{\Sigma}_\mathbf{uv}^\mathbf{T}\boldsymbol{\Sigma}_\mathbf{u}^{-1}(\mathbf{u} - \boldsymbol{\mu}_\mathbf{u}), \boldsymbol{\Sigma}_\mathbf{v} - \boldsymbol{\Sigma}_\mathbf{uv}^\mathbf{T}\boldsymbol{\Sigma}_\mathbf{u}^{-1}\boldsymbol{\Sigma}_\mathbf{uv}\right).$$

   Hint: You will need the following property of determinants:

$$\left|\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}\right| = |\mathbf{A}|\left|\mathbf{D - CA^{-1}B}\right| = |\mathbf{D}|\left|\mathbf{A - CD^{-1}B}\right|.$$

(d) Show that

$$\mathbb{P}(\mathbf{u}) = \int \mathbb{P}(\mathbf{x})\,\mathrm{d}\mathbf{v} = \mathcal{N}_{N_u}(\mathbf{u}, \boldsymbol{\mu}_\mathbf{u}, \boldsymbol{\Sigma}_\mathbf{u}) \quad \text{and hence argue} \quad \mathbb{P}(\mathbf{v}) = \int \mathbb{P}(\mathbf{x})\,\mathrm{d}\mathbf{u} = \mathcal{N}_{N_v}(\mathbf{v}, \boldsymbol{\mu}_\mathbf{v}, \boldsymbol{\Sigma}_\mathbf{v})\,.$$

These are described as 'marginal' densities, as they are obtained by 'marginalising over' some of the parameters.

Hint: You may find it easiest to start by considering the 2-dimensional case, where u and v are scalar variables.

(e) Starting from the definition of conditional probability, show that the distribution of $\mathbf{u}$ given an observation $\mathbf{v_0}$ has density

$$\mathbb{P}(\mathbf{u}\,|\,\mathbf{v_0}) = \mathcal{N}_{N_u}\left(\boldsymbol{\mu}_\mathbf{u} + \boldsymbol{\Sigma}_\mathbf{uv}\boldsymbol{\Sigma}_\mathbf{v}^{-1}(\mathbf{v_0} - \boldsymbol{\mu}_\mathbf{v}), \boldsymbol{\Sigma}_\mathbf{u} - \boldsymbol{\Sigma}_\mathbf{uv}\boldsymbol{\Sigma}_\mathbf{v}^{-1}\boldsymbol{\Sigma}_\mathbf{uv}^\mathbf{T}\right)\,.$$

Linear, under-determined problems

9. *X-Ray Tomography* – This exercise is available as a Jupyter notebook.

10. *Regression, regularised* – Generally, regularisation isn't sensible in regression problems: if the data is insufficient to constrain a complex model, we usually adopt the principle of 'Occam's razor' and fit a simpler model instead. However, it can be instructive to visualise how regularisation affects the solution to a regression problem. Adapt the examples from Exercises 1 and 4 to incorporate regularisation, and explore how this changes performance. The function `curveFitting.curveFittingInv` accepts an argument `regularisation=eps2`, where `eps2` is the ϵ^2 parameter of Tikhonov regularisation. In addition, you can (optionally) specifiy `priorModel=m`.

11. *Deriving Tikhonov regularisation* – This exercise follows on from Exercise 6. Define a new misfit function,

$$\phi(\mathbf{m}) = \|\mathbf{d} - \mathbf{Gm}\|_2^2 + \epsilon^2\|\mathbf{m}\|_2^2$$

Differentiate this expression with respect to an arbitrary model parameter, and hence obtain an expression for the regularised solution to an inverse problem. How does this expression change if the penalty term is, instead, $\epsilon^2\|\mathbf{Wm}\|_2^2$ for a general matrix $\mathbf{W}$? Compare your solution to the one for Exercise 18. How are the two related?

12. *Inversion in the data-space* – Using the results from Exercise 8b, rewrite the posterior distribution from Exercise 18 in an alternate form. What are the computational pros and cons of employing this in practice?

Hint: Try employing this for one of the inversion problems!

13. *Regularisation and the SVD* – The Singular Value Decomposition (SVD) is a tool that allows any $N \times M$ matrix $\mathbf{M}$ to be expressed in the form $\mathbf{M} = \mathbf{U\Sigma V^T}$, where $\mathbf{U}$ has dimension $N \times M$, $\boldsymbol{\Sigma}$ is $M \times M$ with entries on the diagonal only (known as 'the singular values'), and where $\mathbf{V}$ is $M \times M$. Furthermore, $\mathbf{U^TU} = \mathbf{UU^T} = \mathbf{I}$ and $\mathbf{V^TV} = \mathbf{VV^T} = \mathbf{I}$. In Python, you can perform an SVD by calling

```
> u,sigma,vt = np.linalg.svd(M)
```

For square matrices $\mathbf{M}$, the SVD is closely-related to the concept of eigenvector/eigenvalue decompositions.

(a) Express the *unregularised* least-squares algorithm in terms of the Singular Value Decomposition for $\mathbf{G}$. What happens if some of the singular values are very small or even zero?

(b) How does the addition of (Tikhonov) regularisation change your expression? What does this mean for small singular values?

Implement an SVD-based inversion algorithm for the X-ray tomography problem.

14. *Regularisation via an L_1 norm* – This exercise builds on Exercise 11, where we modified the misfit function to incorporate an L_2 penalty on the model length. An alternative might be to penalise the L_1 norm,

$$\|\mathbf{m}\|_1 = \sum_i |m_i|$$

by using a misfit function

$$\phi(\mathbf{m}) = \|\mathbf{d} - \mathbf{Gm}\|_2^2 + \epsilon^2 \|\mathbf{m}\|_1 \,.$$

However, the $|m_i|$ terms in this expression cannot be handled within the 'standard' least-squares framework. Instead, it is necessary to employ methods of *convex optimisation.* In Python, the appropriate routine is `cvxopt.solvers.qp` which can be called as follows:

```
sol = cvxopt.solvers.qp(P,q,G,h,A,b)
```

This solves problems with the following form:

$$\text{minimize } \frac{1}{2}\mathbf{x}^\mathbf{T}\mathbf{P}\mathbf{x} + \mathbf{q}^\mathbf{T}\mathbf{x} \text{ subject to } \begin{cases} \mathbf{Gx} \leq \mathbf{h} \\ \mathbf{Ax} = \mathbf{b} \end{cases}$$

where vector inequalities are applied element-by-element; the solution is then accessible as `sol['x']`. By introducing new 'slack variables' **u** and **v**, defined such that

$$u_i = \begin{cases} m_i & m_i \geq 0 \\ 0 & m_i < 0 \end{cases} \qquad\qquad v_i = \begin{cases} 0 & m_i \geq 0 \\ -m_i & m_i < 0 \end{cases}$$

rewrite our optimisation problem in a form appropriate for `cvxopt.solvers.qp` and implement this in Python for the X-ray tomography problem. How do results compare to L_2 optimisation?

15. *Inversion using global basis functions* – The X-ray tomography example employed local basis functions: we discretised the model region into individual grid cells. However, one could use any other set of basis functions. Write a forward solver that works with (for example) a model expressed as a two-dimensional Fourier series, and experiment with solving inverse problems in this basis. What are the differences in performance between this and the discretised case?

Hint: You can use `scipy.integrate.quad` to perform integrals along a ray path.

Bayesian inference

16. *Bayesian and Frequentist Inference of a biased coin* – This exercise is available as a Jupyter notebook.

17. *Use Bayesian inference to infer the number of tickets sold in a Lottery* – The unknown in the problem is the total number of entries sold, n, (i.e. sets of 6 numbers) which must be inferred from knowledge of the number of winners of each division $d_i, (i = 1, \ldots, N_{div})$, where N_{div} is 6 for the example below. The values of $d_i, (i = 1, \ldots, N_{div})$ are our data and are assumed to be a known set of integers without error. In reality the real value of n is never made public.

    Your task is to get a solution from a Frequentist viewpoint, using the data to make a single estimate of n, and also a Bayesian inference viewpoint using the data to construct a probability distribution for n.

    The probability of winning each division, is independent of the total number of entries n, so these may be treated as a set of known constants, $p_i, (i = 1, \ldots, N_{div})$, the value of which depends on the details of the game.

    (a) Decide how to make a Frequentist solution to this problem. Do this for the data above to get an estimate for n. By how much do these estimates vary ?

i	1	2	3	4	5	6
d_i	14	169	3059	149721	369543	802016
p_i^{-1}	8145060	678756	36696	732	300	144

Table 1: Tattslotto dividend results for draw number 3253 on 29/09/2012. Total prize pool of \$49.92m, with division 1 prize of \$22m. The cost of a single entry is about \$0.65.

(b) A Bayesian inference approach requires us to find the Likelihood and prior and then multiply them together. Lets assume our prior is uniform between $1 < n < 3 \times 10^8$ which is a safe assumption. The likelihood is the probability of the data given the model, i.e. the probability that there would be d_i winners of division i and $n - d_i$ non winners when there are n tickets sold. The binomial theorem tells us that this probability, $p(d_i|n)$, is given by

$$p(d_i|n) = \frac{n!}{d_i!(n-d_i)!} \times p_i^{d_i}(1-p_i)^{n-d_i}$$

All values in this expression are known except the single unknown n. Since the number of winners in each division provides independent data, the total likelihood is the product of similar terms for each division, i.e.

$$p(\mathbf{d}|n) = \prod_{i=1}^{N_{div}} p(d_i|n) \tag{1}$$

Bayes' theorem says that to find the *a posteriori* probability distribution for the unknown n we just multiply the likelihood by the prior. Since the prior is a constant the result is

$$p(n|\mathbf{d}) \propto \prod_{i=1}^{N_{div}} \frac{n!}{(n-d_i)!} \times (1-p_i)^{n-d_i}$$

which holds for $1 \leq n \leq 3 \times 10^8$. Outside this range the posterior PDF is zero because the prior is zero. Our only interest is in the unknown n and so the constant of proportionality is used to absorb all quantities independent of n.

Your task is to use the values of $(d_i, p_i), i = 1, \ldots, 6$ from the table and plot the probability distribution as a function of n. Do this in the range 112.5m - 114.5m. Compare this curve to the single frequentist estimate of n you obtained in part (a), what do you notice?

Hint: In any computer program it is always best to calculate $\log p(n|\mathbf{d})$ first and then take an exponent to evaluate the curve as a function of n. Stirling's formulae for the approximation to $n!$ may be useful.

(c) Repeat the problem using the Maximum Likelihood (ML) approach. This is done by finding the value of n which maximises the the likelihood expression in eqn. (1). Since the prior is a constant for this problem the likelihood is proportional to the curve you produced in part (b). You could probably do it visually. Plot the average estimate you obtained in part (a) on top of the curve from part (b). How does the ML solution compare to the solution from part (a)?

18. *The Bayesian derivation of least squares* – Suppose that m is a random vector encapsulating our state of knowledge about a model. We assert that our *prior* knowledge can be represented by a Gaussian centred on some point $\mathbf{m_p}$ and with covariance matrix $\mathbf{C_m}$, so that $\mathsf{m} \sim \mathcal{N}(\mathbf{m_p}, \mathbf{C_m})$. Assuming that observations are linearly related to model parameters via $\mathbf{d} = \mathbf{Gm}$, and that observational noise can be modelled by a zero-mean Gaussian with covariance matrix $\mathbf{C_d}$, show that the *posterior* distribution is given by

$$\mathsf{m}\,|\,\mathbf{d_0} \sim \mathcal{N}\left(\mathbf{m_p} + \left(\mathbf{G^TC_d^{-1}G} + \mathbf{C_m^{-1}}\right)^{-1}\mathbf{G^TC_d^{-1}}\left(\mathbf{d} - \mathbf{Gm_p}\right), \left(\mathbf{G^TC_d^{-1}G} + \mathbf{C_m^{-1}}\right)^{-1}\right).$$

How does this expression compare to the one you obtained for Exercise 6? What assumptions would the Bayesian claim are implicit within the form of the least-squares algorithm given in that question?

Hint: Begin by obtaining an expression for the joint prior distribution of data and models. The following results may be useful.

- Linear transformations of Gaussians: If $\mathsf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ then $\mathbf{M}\mathsf{x} \sim \mathcal{N}\left(\mathbf{M}\boldsymbol{\mu}, \mathbf{M}\boldsymbol{\Sigma}\mathbf{M^T}\right)$.
- The Woodbury matrix identity: see Exercise 8b.
- Conditioning of Gaussian distributions: see Exercise 8e.

Assessing and interpreting results

19. *Resolution and posterior covariance* – This exercise is available as a Jupyter notebook.

20. *Resolution and the SVD* – This question builds on Exercise 13. Write down an expression for the resolution operator in a (Tikhonov-regularised) linear inverse problem, in terms of the SVD of the linear operator $\mathbf{G}$. How does regularisation manifest itself in the resolution of an inversion?

21. By combining the two expressions derived in Exercise 8b, show that the posterior covariance matrix is related to the resolution operator and the prior covariance matrix by

$$\left(\mathbf{G^TC_d^{-1}G} + \mathbf{C_m^{-1}}\right)^{-1} = \left(\mathbf{I} - \mathbf{R}\right)\mathbf{C_m}\,.$$

Weakly non-linear problems

22. *Travel-time tomography* – This exercise is available as a Jupyter notebook.

23. *Gradient descent* – Another strategy for solving weakly non-linear inverse problems is to employ gradient-based optimisation algorithms to ‘walk downhill’ on the misfit surface. Try implementing this for the travel-time tomography problem. A variety of optimisation routines are available in `scipy.optimize`; one good choice is `scipy.optimize.fmin_l_bfgs_b`.

24. *Deriving the iterative least-squares algorithm* – Suppose that $\mathbf{g}(\mathbf{m})$ is a weakly non-linear forward model. Starting from a Taylor expansion of $\mathbf{g}$ about some point, $\mathbf{m}_i$, derive an expression for the optimal model update.

25. *The Bayesian update* – A Bayesian analysis of the iterative solution to a weakly non-linear inverse problem leads to the following algorithm:

$$\mathbf{m}_{i+1} = \mathbf{m}_i + \left(\mathbf{G^TC_d^{-1}G} + \mathbf{C_m^{-1}}\right)^{-1}\left[\mathbf{G^TC_d^{-1}}\left(\mathbf{d} - \mathbf{s}(\mathbf{m}_i)\right) - \mathbf{C_m^{-1}}\left(\mathbf{m}_i - \mathbf{m}_0\right)\right]$$

where $\mathbf{m}_0$ is the prior model, and where the linear operator $\mathbf{G}$ is understood to have been computed for the model $\mathbf{m}_i$. By employing the result of Exercise 21, explain the role of the ‘extra’ term involving $\mathbf{C_m^{-1}}\left(\mathbf{m}_i - \mathbf{m}_0\right)$.

26. By computing the necessary partial derivatives using a finite-difference approach, implement and test a source-location algorithm for the X-ray tomography problem.

Fully non-linear problems

27. *Receiver functions* – This exercise is available as a Jupyter notebook.

28. *Monte Carlo search* – By generating and testing candidate models completely at random, try and find good solutions to the linear regression problem of Exercise 1. How well does it perform? How many samples do you need to test to be reasonably confident that you've found a 'good' solution? How does this scale as you increase the number of model parameters sought?

Probabilistic sampling

29. *McMC on Receiver functions* – This exercise is available as a Jupyter notebook.

Avoiding assumptions

30. *'Denuisanced' least squares* – A dataset is assumed to be linearly dependent on some parameters of interest, $\mathbf{x}$, and also on some unwanted 'nuisance' parameters, $\mathbf{y}$, such that $\mathbf{d} = \mathbf{Ax} + \mathbf{By}$. Show that the best-fitting value of $\mathbf{x}$ is given by

$$\hat{\mathbf{x}} = \left[\mathbf{A^TA} - \mathbf{A^TB}\left(\mathbf{B^TB}\right)^{-1}\mathbf{B^TA}\right]^{-1}\left[\mathbf{A^Td} - \mathbf{A^TB}\left(\mathbf{B^TB}\right)^{-1}\mathbf{B^Td}\right]$$

How does this expression change in the case of a regularised, weakly non-linear inversion?

Exotic techniques

31. Try to code up one or more of the methods discussed in lectures. You can make use of an appropriate forward model from the `inversionCourse` module.

Unsupervised methods

32. The file `clusters.dat` contains (x, y) pairs drawn from three distinct clusters (the cluster label is given as the third column in the file).

    (a) Use the Python routine `scipy.cluster.vq.kmeans2` to perform k-means clustering of this dataset. How well does it perform?

    Hint: There are (at least) two different implementations of k-means available in Python: `scipy.cluster.vq.kmeans` and `scipy.cluster.vq.kmeans2`. They employ different convergence criteria, but ought to give similar results. The advantage of `kmeans2` here is that it returns a list of which points have been assigned to each cluster, making it easier to assess performance.

    (b) Write and test your own version of the k-means algorithm.