Generalized Expectation-Maximization

Lecture 08

A reading list for this week:

Bishop 9.1,9.2,

Hastie et al 13.1, 13.2, 14.3 - 14.3.11

Bonaccorso “Clustering fundamentals”

Unfinished business from before…

Self-organizing maps

A NN that is kinda sorta like K-means but with topology

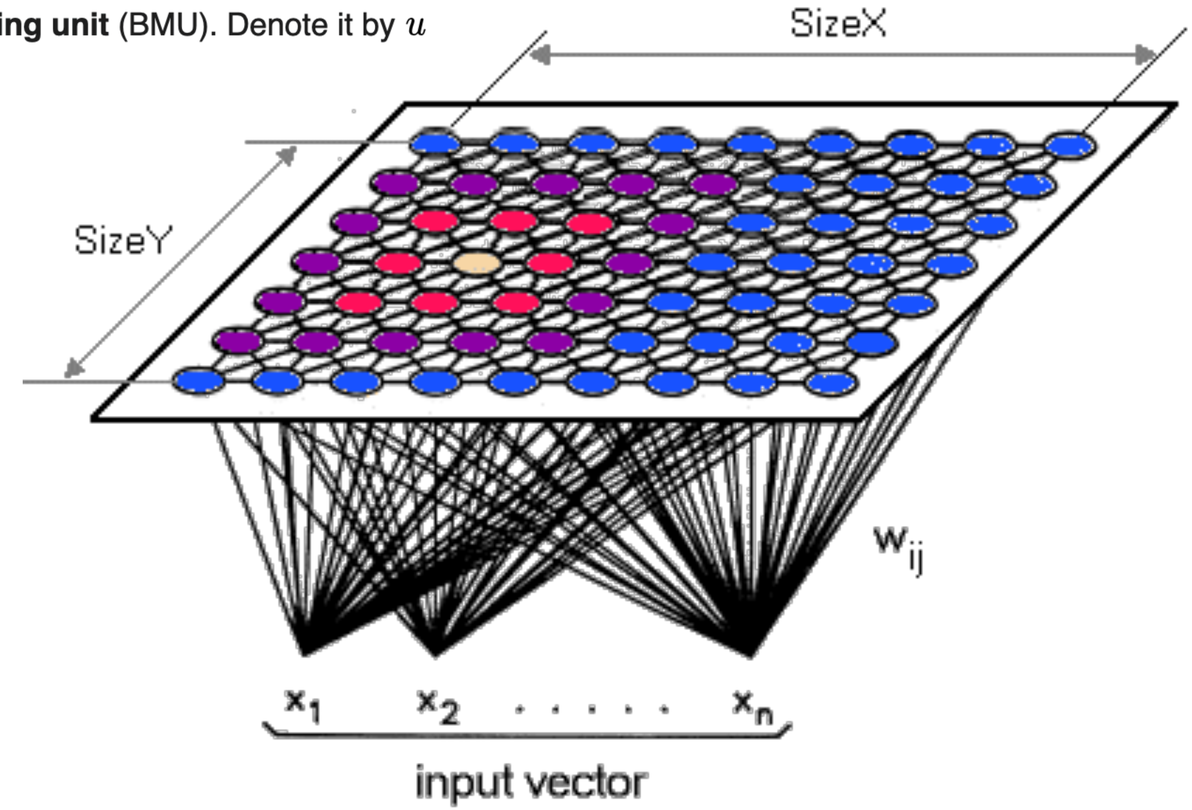
Algorithm [edit]

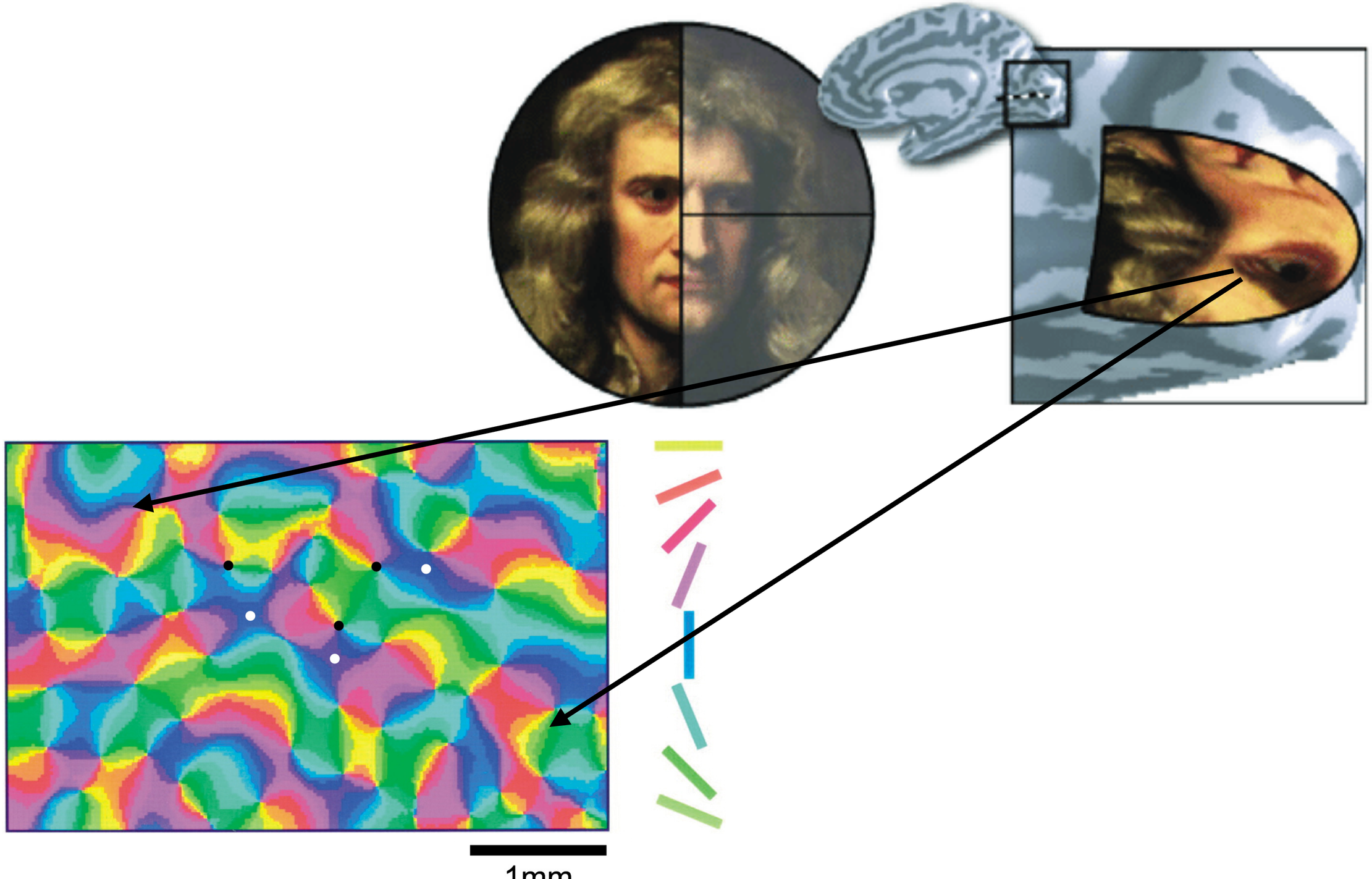
1. Randomize the node weight vectors in a map
2. For $s = 0, 1, 2, \ldots, \lambda$
    1. Randomly pick an input vector $D(t)$
    2. Find the node in the map closest to the input vector. This node is the **best matching unit** (BMU). Denote it by u
    3. For each node v, update its vector by pulling closer to the input vector:

$$W_v(s+1) = W_v(s) + \theta(u, v, s) \cdot \alpha(s) \cdot (D(t) - W_v(s))$$

The variable names mean the following, with vectors in bold,

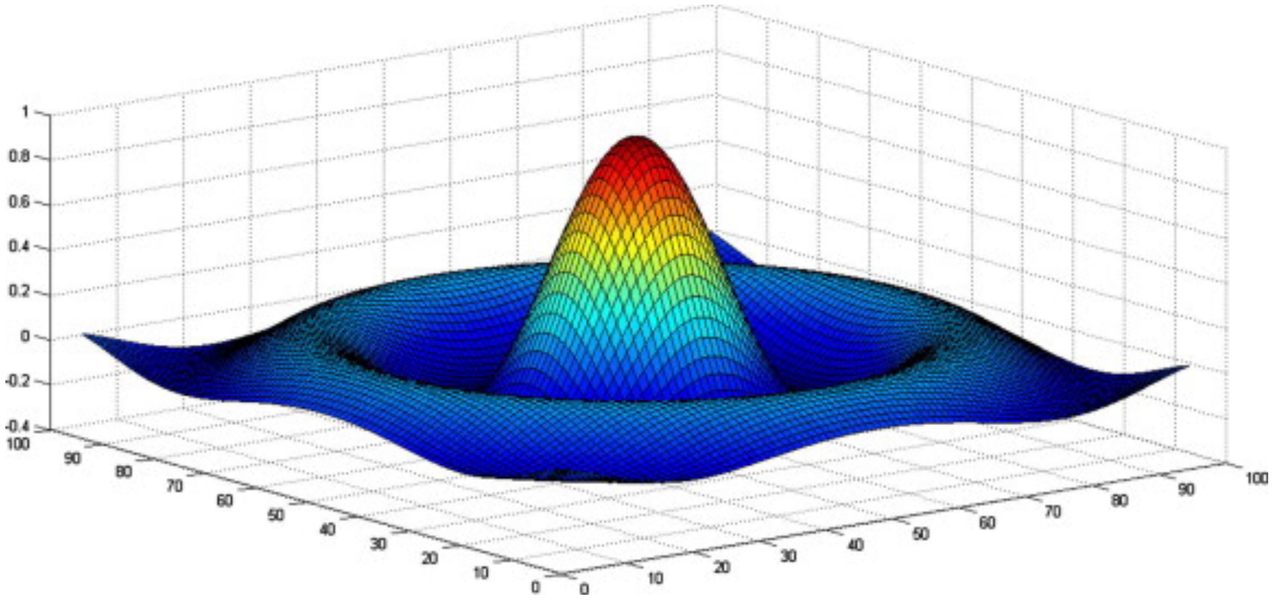
- s is the current iteration
- λ is the iteration limit
- t is the index of the target input data vector in the input data set $\mathbf{D}$
- $D(t)$ is a target input data vector
- v is the index of the node in the map
- $\mathbf{W}_v$ is the current weight vector of node v
- u is the index of the best matching unit (BMU) in the map
- $\theta(u, v, s)$ is the neighbourhood function,
- $\alpha(s)$ is the learning rate schedule.




1mm

Competitive learning

Mexican hat: a competitive neighborhood function for SOM



Algorithm 13.1 *Learning Vector Quantization—LVQ.*

1. Choose R initial prototypes for each class: $m_1(k), m_2(k), \ldots, m_R(k)$, $k = 1, 2, \ldots, K$, for example, by sampling R training points at random from each class.

2. Sample a training point x_i randomly (with replacement), and let (j, k) index the closest prototype $m_j(k)$ to x_i.

   (a) If $g_i = k$ (i.e., they are in the same class), move the prototype towards the training point:

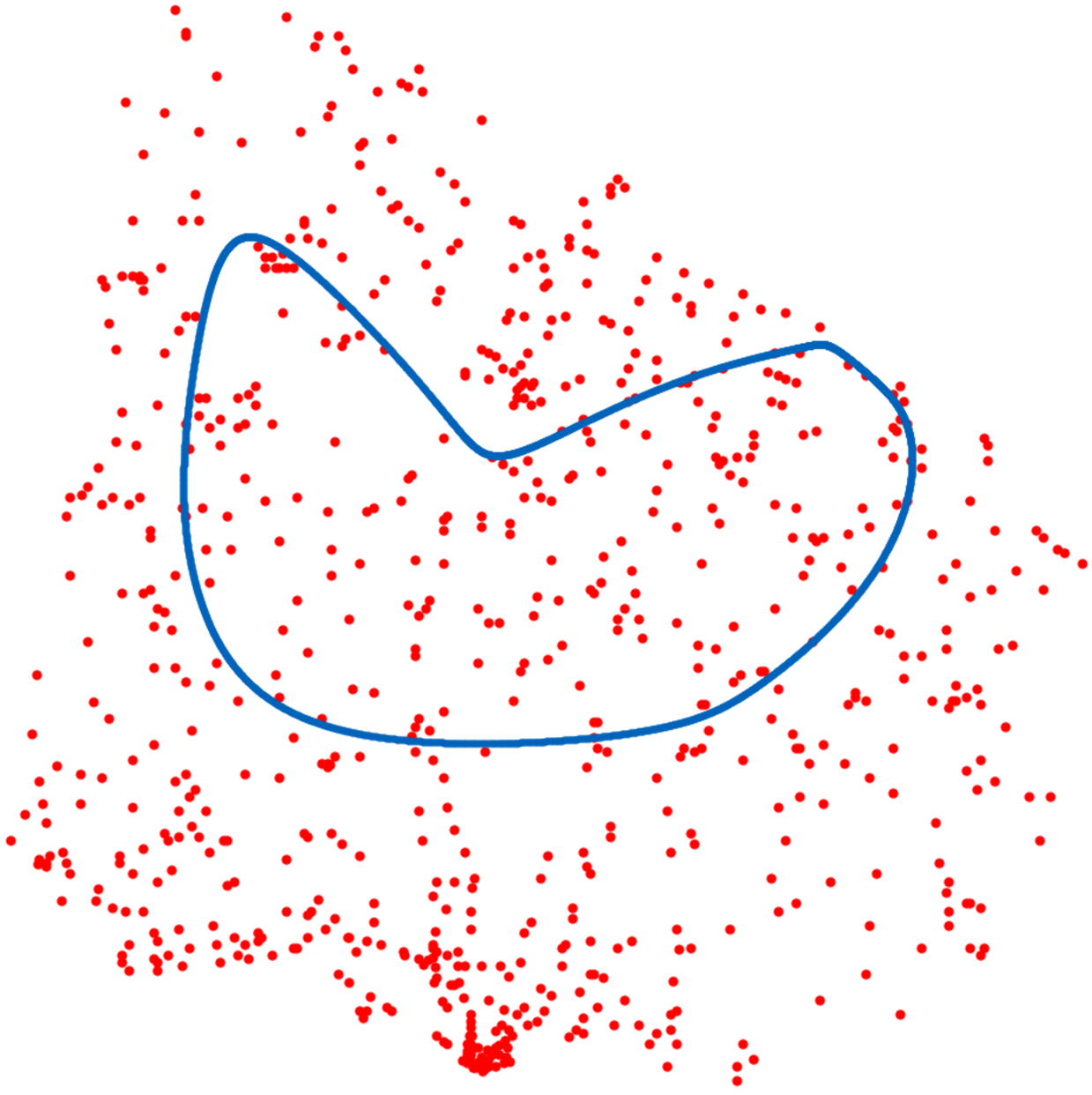
   $$m_j(k) \leftarrow m_j(k) + \epsilon(x_i - m_j(k)),$$

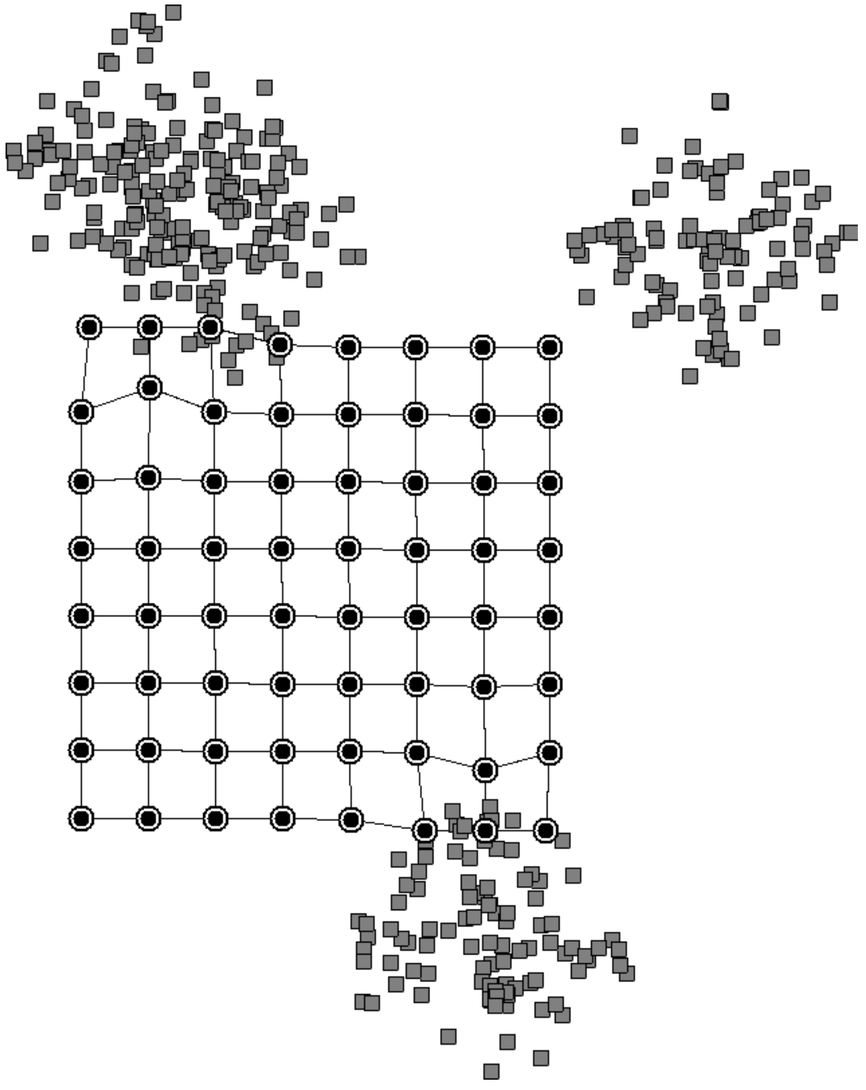
   where ϵ is the *learning rate*.

   (b) If $g_i \neq k$ (i.e., they are in different classes), move the prototype away from the training point:

   $$m_j(k) \leftarrow m_j(k) - \epsilon(x_i - m_j(k)).$$

3. Repeat step 2, decreasing the learning rate ϵ with each iteration towards zero.



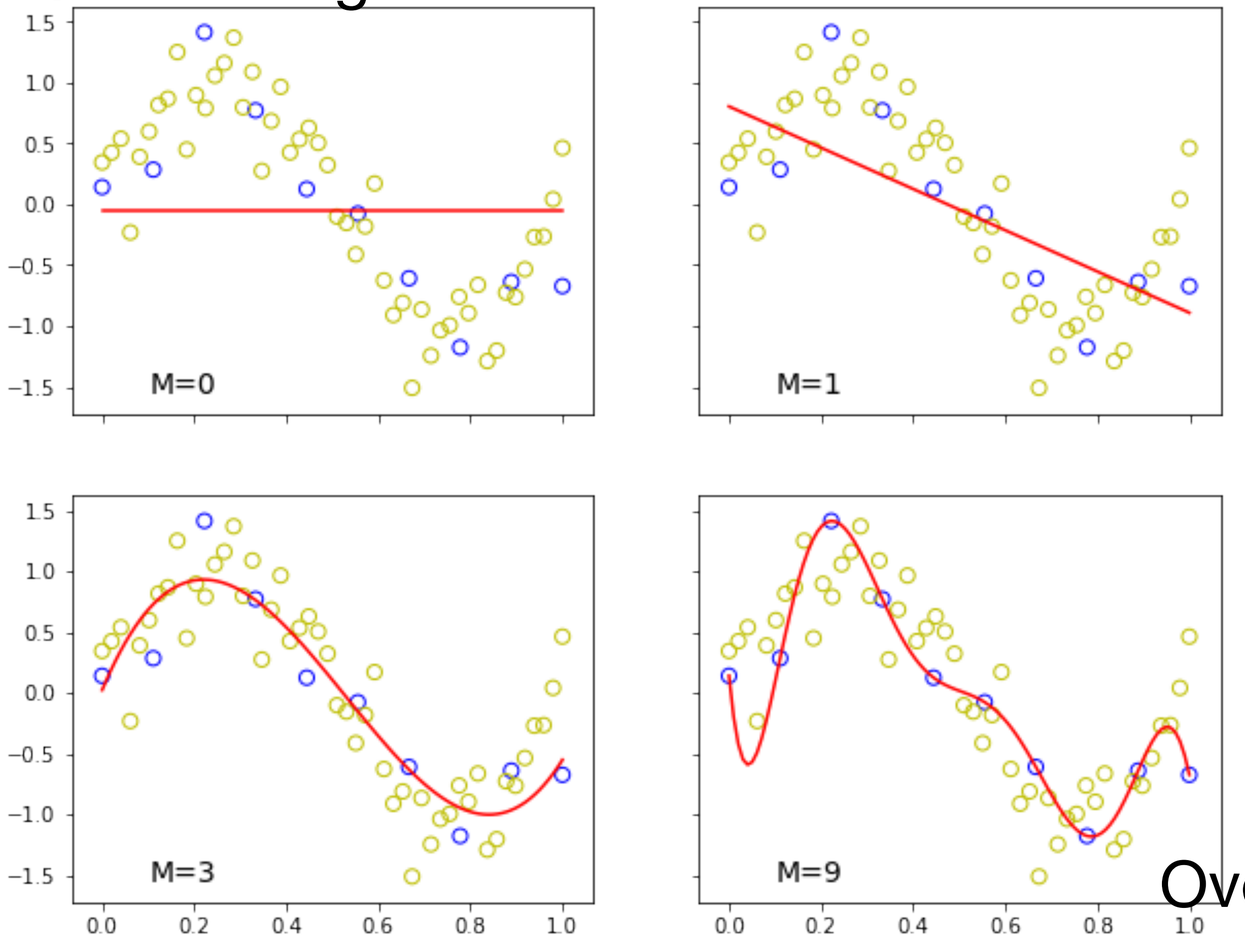


Estimating model performance

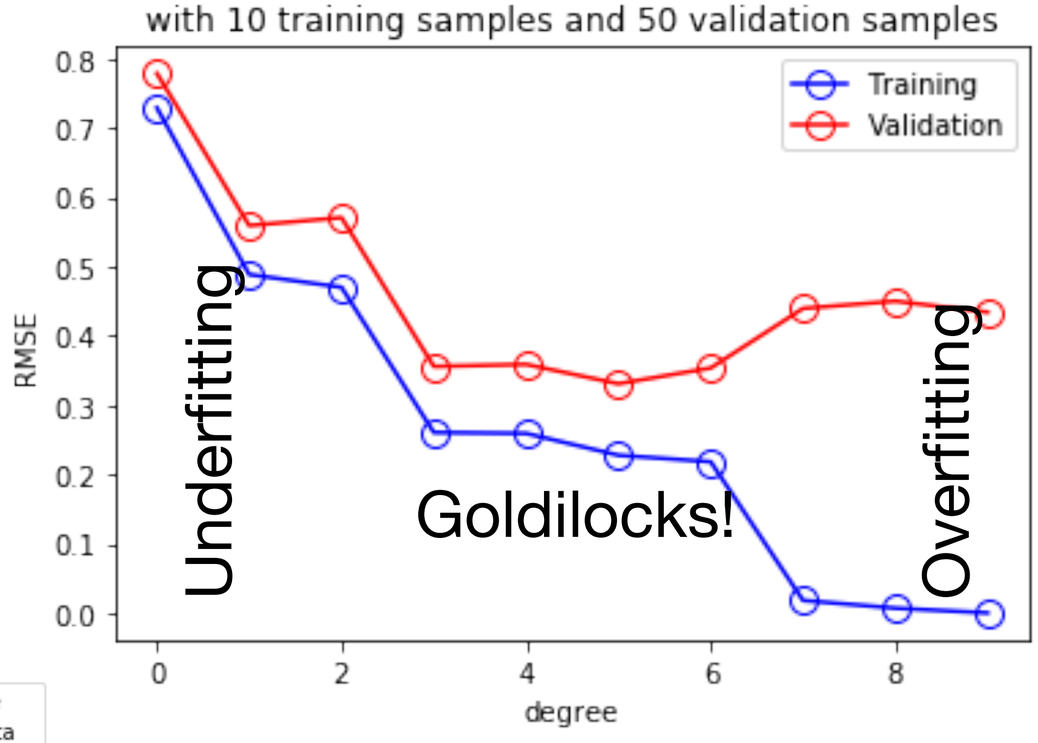
- Terminology
  - Training set - used to fit the model
  - Validation set - used to select which model; NO OVERLAP WITH ABOVE
  - Test set - used to predict performance; NO OVERLAP WITH ABOVE
- Training set performance is ON AVERAGE better than Test set performance
  - Why?

One form of the bias - variance tradeoff!

Underfitting



Overfitting



A “validation curve”

https://scikit-learn.org/stable/auto_examples/model_selection/plot_validation_curve.html#plotting-validation-curves

Selecting number of clusters

Using in-sample methods

- In (many) applications of clustering we don't have a "correct" answer, test set metrics don't make sense
- Start with training set performance and penalize according to model complexity

The AIC criterion is defined as:

$$AIC = -2\log(\hat{L}) + 2d$$

where $\hat{L}$ is the maximum likelihood of the model and d is the number of parameters (as well referred to as degrees of freedom in the previous section).

The definition of BIC replace the constant 2 by $\log(N)$:

$$BIC = -2\log(\hat{L}) + \log(N)d$$

where N is the number of samples.

Selecting number of clusters

Using in-sample methods

For a linear Gaussian model, the maximum log-likelihood is defined as:

$$\log(\hat{L}) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{2\sigma^2}$$

where σ^2 is an estimate of the noise variance, $y_{\imath}$ and $\hat{y}_i$ are respectively the true and predicted targets, and n is the number of samples.

Plugging the maximum log-likelihood in the AIC formula yields:

$$AIC = n\log(2\pi\sigma^2) + \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sigma^2} + 2d$$

Rand Index

- Compare two clusterings with each other
- Or when you know the correct labels can be used to determine cluster performance
- If as a performance metric, use inside a cross-validation or similar (NOT IN SAMPLE!!)
- What happens when there are 2 vs 200 clusters? That's why there's an adjusted Rand score :)

Rand index [edit]

Definition [edit]

Given a set of n elements $S = \{o_1, \ldots, o_n\}$ and two partitions of S to compare, $X = \{X_1, \ldots, X_r\}$, a partition of S into r subsets, and $Y = \{Y_1, \ldots, Y_s\}$, a partition of S into s subsets, define the following:

- a, the number of pairs of elements in S that are in the **same** subset in X and in the **same** subset in Y
- b, the number of pairs of elements in S that are in **different** subsets in X and in **different** subsets in Y
- c, the number of pairs of elements in S that are in the **same** subset in X and in **different** subsets in Y
- d, the number of pairs of elements in S that are in **different** subsets in X and in the **same** subset in Y

The Rand index, R, is:[1][2]

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}}$$

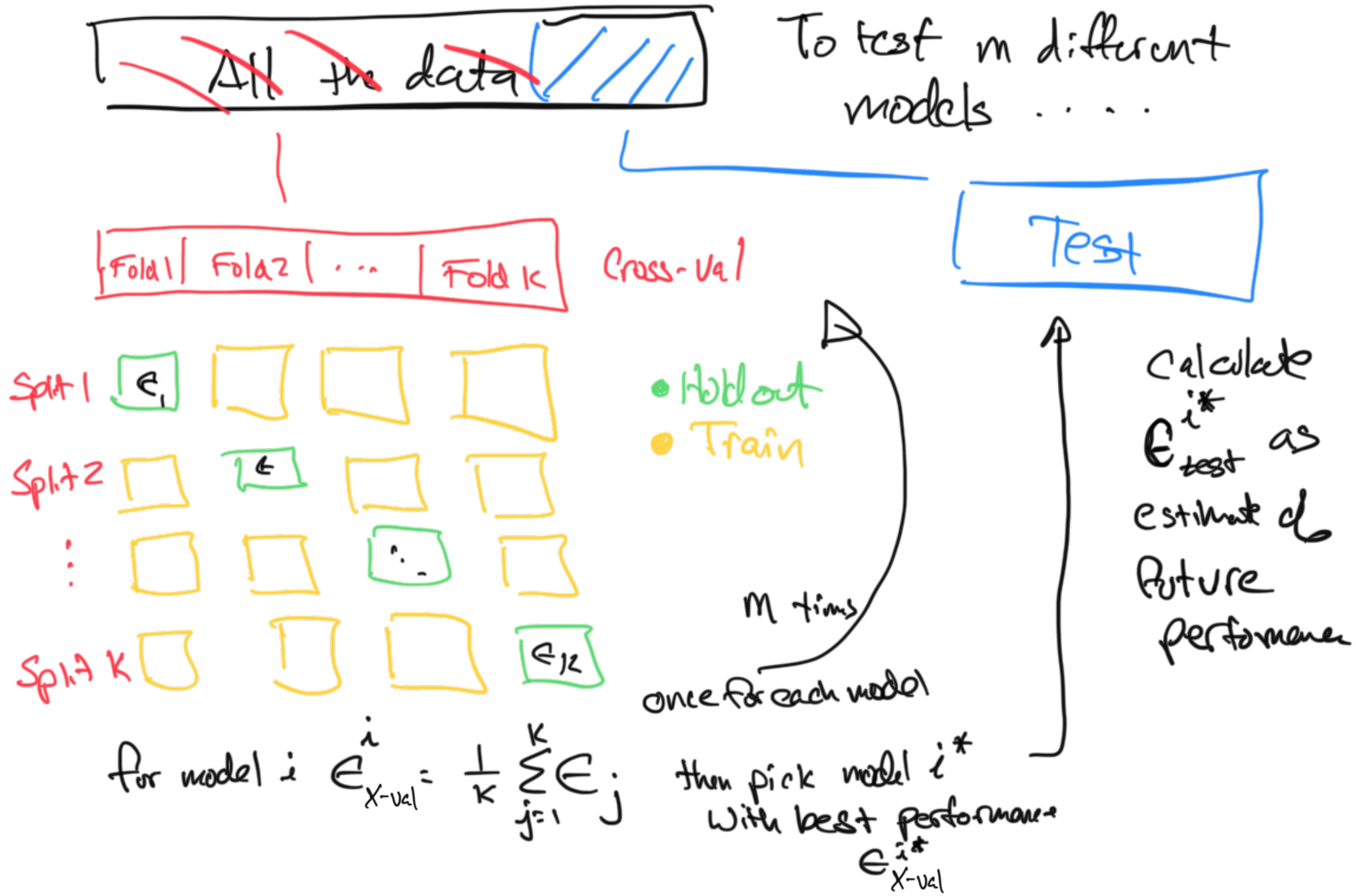
Intuitively, $a + b$ can be considered as the number of agreements between X and Y and $c + d$ as the number of disagreements between X and Y.

Since the denominator is the total number of pairs, the Rand index represents the *frequency of occurrence* of agreements over the total pairs, or the probability that X and Y will agree on a randomly chosen pair.

Cross validation for model selection

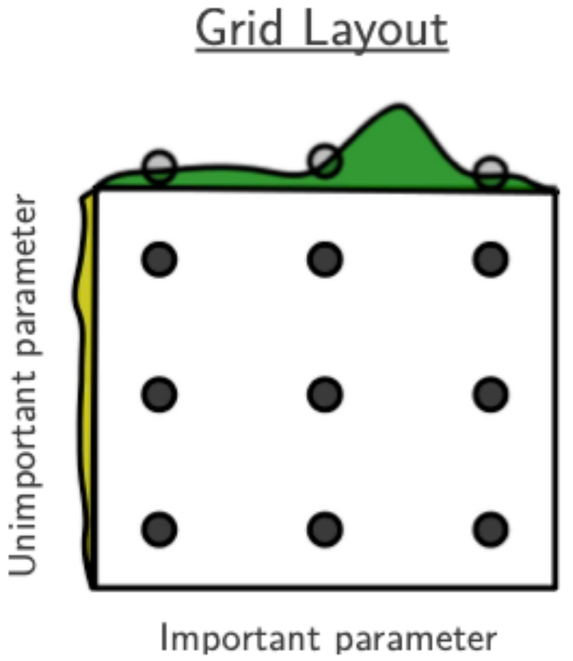
Let's say you had around 8k samples in a dataset

- training set ~ sample 5k
- Grid search of hyper parameters using k-fold cross validation on the training set
- Select best model from grid, train on entire training set
- Evaluate best model on the test set (everything not sampled for training)



Grid Search

- Exhaustive search
- Thorough but expensive
- Specify grid for parameter search
- Can be run in parallel
- Can suffer from poor coverage
- Often run with multiple resolutions

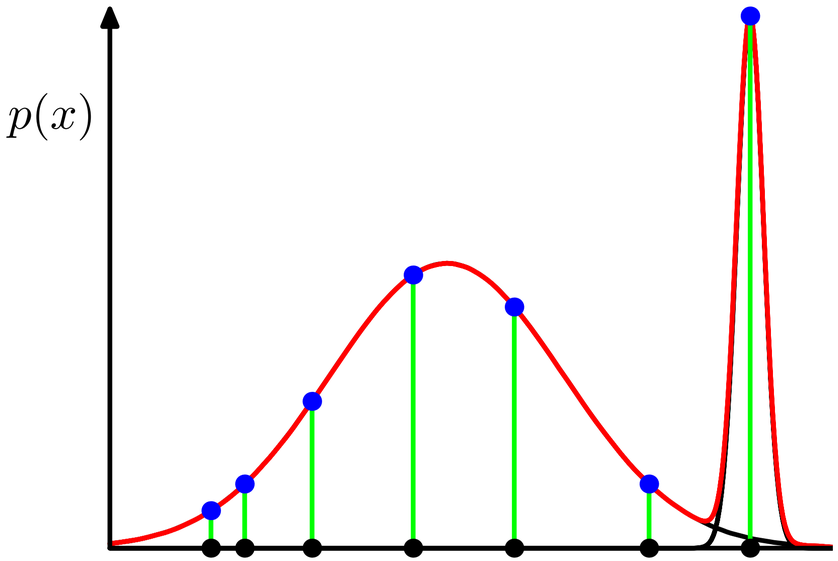


Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, *13*(1), 281-305.

Notebook L08

When do GMM break down?

- Singularity: when one component fits a single datapoint. $\sigma \to 0$, $P(D \mid \theta) \to \infty$
- Risks
  - # components >> # processes in data
  - high dimensionality, low density of datapoints



30-d, 2 class dataset

```
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler

bc = load_breast_cancer()
scaler = StandardScaler()
sbc = scaler.fit_transform(bc.data)

for nclust in [2, 5, 10, 15, 20, 25, 30]:
    gmm = GaussianMixture(n_components=nclust, covariance_type='spherical')
    gmm.fit(sbc)
    print(f'smallest variance in {nclust:2} gaussian mixture model: {gmm.covariances_.min():3.2
```

```
smallest variance in  2 gaussian mixture model: 3.56e-01; model BIC=4.0562e+04
smallest variance in  5 gaussian mixture model: 2.03e-01; model BIC=3.6397e+04
smallest variance in 10 gaussian mixture model: 1.33e-01; model BIC=3.3517e+04
smallest variance in 15 gaussian mixture model: 1.00e-06; model BIC=3.2643e+04
smallest variance in 20 gaussian mixture model: 1.00e-06; model BIC=3.2153e+04
smallest variance in 25 gaussian mixture model: 1.00e-06; model BIC=3.1666e+04
smallest variance in 30 gaussian mixture model: 1.00e-06; model BIC=3.1353e+04
```

A generalized EM algorithm

- Does it converge?
- If so, are rates of convergence known?
- Are its solutions optimal? If not, can we bound their suboptimality?
- What is its relationship to the likelihood and our goal of maximizing it?

General formulation of EM

The General EM Algorithm

Given a joint distribution $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ over observed variables $\mathbf{X}$ and latent variables $\mathbf{Z}$, governed by parameters $\boldsymbol{\theta}$, the goal is to maximize the likelihood function $p(\mathbf{X}|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

1. Choose an initial setting for the parameters $\boldsymbol{\theta}^{\text{old}}$.

2. **E step** Evaluate $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$.

3. **M step** Evaluate $\boldsymbol{\theta}^{\text{new}}$ given by

$$\boldsymbol{\theta}^{\text{new}} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) \tag{9.32}$$

   where

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \tag{9.33}$$

4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let

$$\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}^{\text{new}} \tag{9.34}$$

   and return to step 2.

Latent variables

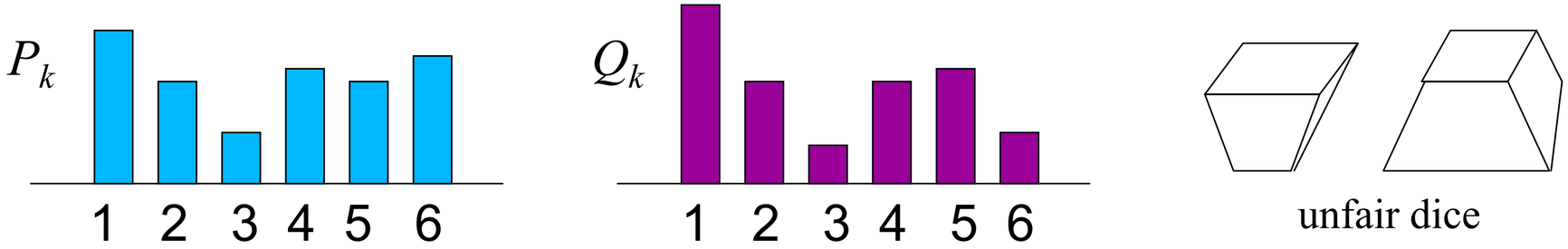
Consider a probabilistic model in which we collectively denote all of the observed variables by $\mathbf{X}$ and all of the hidden variables by $\mathbf{Z}$. The joint distribution $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ is governed by a set of parameters denoted $\boldsymbol{\theta}$. Our goal is to maximize the likelihood function that is given by

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \tag{9.69}$$

- Latents could be anything we cannot directly observe:
  - Cluster assignment of data points
  - Missing values in data
  - Nuisance variable

Kullback Leibler Divergence

Idea: Consider you want to compare two probability distributions P and Q that are defined over the same set of outcomes.



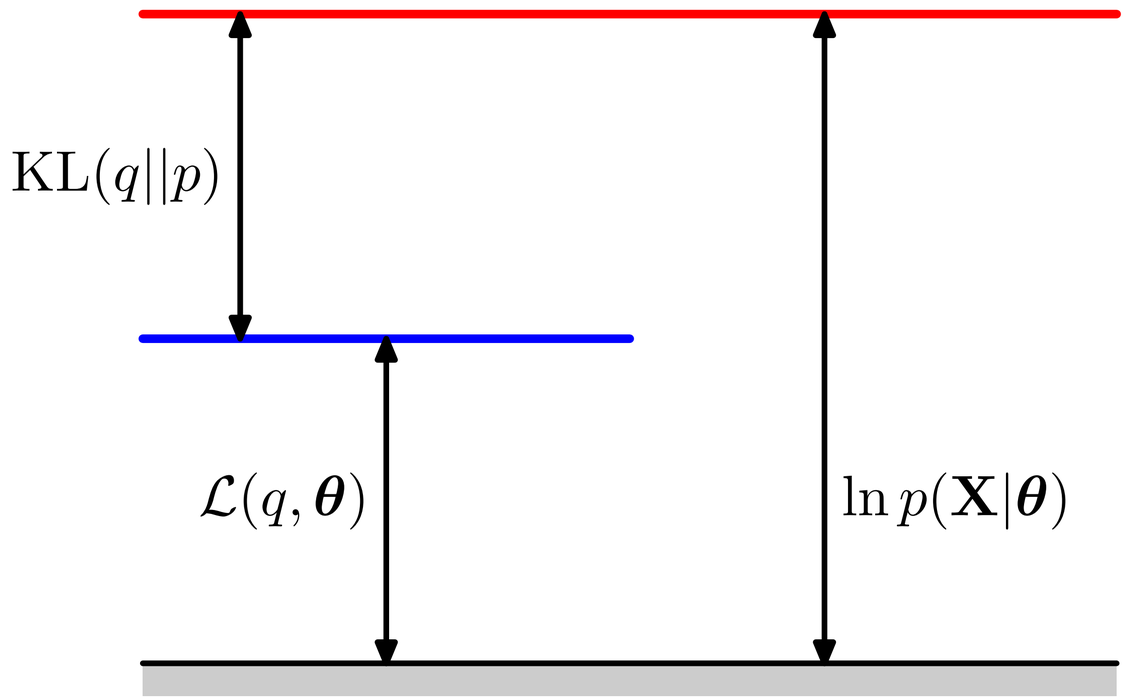
A "natural" way of defining a "distance" between two distributions is the so-called *Kullback-Leibler divergence (KL-distance), or relative entropy*:

$$D(P \,\|\, Q) \equiv E_P\left[\log \frac{P(X)}{Q(X)}\right] = \sum_k P(x_k) \log \frac{P(x_k)}{Q(x_k)}$$

We shall suppose that direct optimization of $p(\mathbf{X}|\boldsymbol{\theta})$ is difficult, but that optimization of the complete-data likelihood function $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ is significantly easier. Next we introduce a distribution $q(\mathbf{Z})$ defined over the latent variables, and we observe that, for any choice of $q(\mathbf{Z})$, the following decomposition holds

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q\|p) \tag{9.70}$$

Illustration of the decomposition given by (9.70), which holds for any choice of distribution $q(\mathbf{Z})$. Because the Kullback-Leibler divergence satisfies $\mathrm{KL}(q\|p) \geqslant 0$, we see that the quantity $\mathcal{L}(q, \boldsymbol{\theta})$ is a lower bound on the log likelihood function $\ln p(\mathbf{X}|\boldsymbol{\theta})$.



We shall suppose that direct optimization of $p(\mathbf{X}|\boldsymbol{\theta})$ is difficult, but that optimization of the complete-data likelihood function $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ is significantly easier. Next we introduce a distribution $q(\mathbf{Z})$ defined over the latent variables, and we observe that, for any choice of $q(\mathbf{Z})$, the following decomposition holds

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q\|p) \tag{9.70}$$

where we have defined

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})} \right\} \tag{9.71}$$

$$\mathrm{KL}(q\|p) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}. \tag{9.72}$$

The General EM Algorithm

Given a joint distribution $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ over observed variables $\mathbf{X}$ and latent variables $\mathbf{Z}$, governed by parameters $\boldsymbol{\theta}$, the goal is to maximize the likelihood function $p(\mathbf{X}|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

1. Choose an initial setting for the parameters $\boldsymbol{\theta}^{\text{old}}$.

2. **E step** Evaluate $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$.

3. **M step** Evaluate $\boldsymbol{\theta}^{\text{new}}$ given by

$$\boldsymbol{\theta}^{\text{new}} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) \tag{9.32}$$

   where

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \tag{9.33}$$

4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let

$$\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}^{\text{new}} \tag{9.34}$$

   and return to step 2.

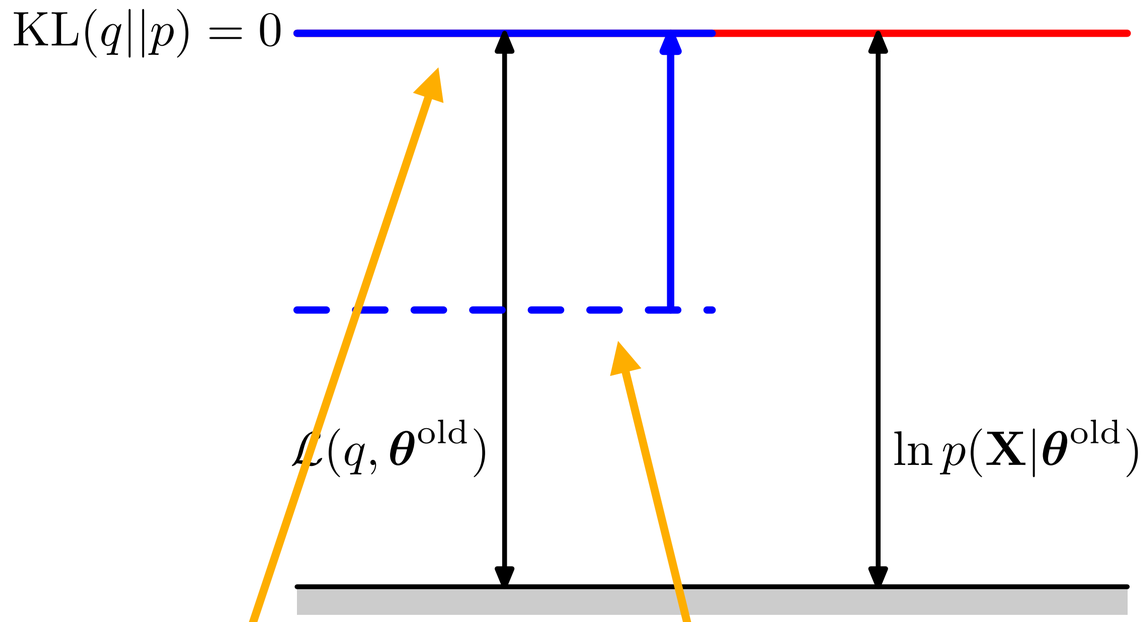


Illustration of the E step of the EM algorithm. The q distribution is set equal to the posterior distribution for the current parameter values $\boldsymbol{\theta}^{\mathrm{old}}$, causing the lower bound to move up to the same value as the log likelihood function, with the KL divergence vanishing.

shown in Figure 9.13. If we substitute $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\mathrm{old}})$ into (9.71), we see that, after the E step, the lower bound takes the form

$$\begin{aligned} \mathcal{L}(q, \boldsymbol{\theta}) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\mathrm{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\mathrm{old}}) \ln p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\mathrm{old}}) \\ &= \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}}) + \mathrm{const} \end{aligned} \tag{9.74}$$

where the constant is simply the negative entropy of the q distribution and is therefore independent of $\boldsymbol{\theta}$. Thus in the M step, the quantity that is being maximized is the

fore independent of $\boldsymbol{\theta}$. Thus in the M step, the quantity that is being maximized is the expectation of the complete-data log likelihood, as we saw earlier in the case of mixtures of Gaussians. Note that the variable $\boldsymbol{\theta}$ over which we are optimizing appears only inside the logarithm. If the joint distribution $p(\mathbf{Z}, \mathbf{X}|\boldsymbol{\theta})$ comprises a member of the exponential family, or a product of such members, then we see that the logarithm will cancel the exponential and lead to an M step that will be typically much simpler than the maximization of the corresponding incomplete-data log likelihood function $p(\mathbf{X}|\boldsymbol{\theta})$.

The operation of the EM algorithm can also be viewed in the space of parameters, as illustrated schematically in Figure 9.14. Here the red curve depicts the (in-

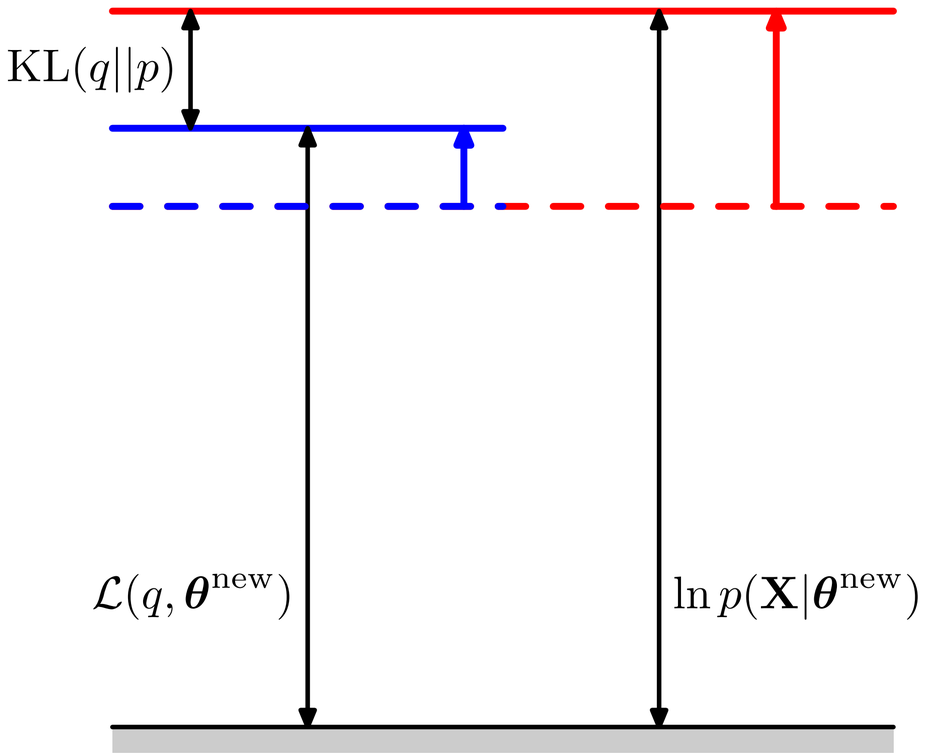
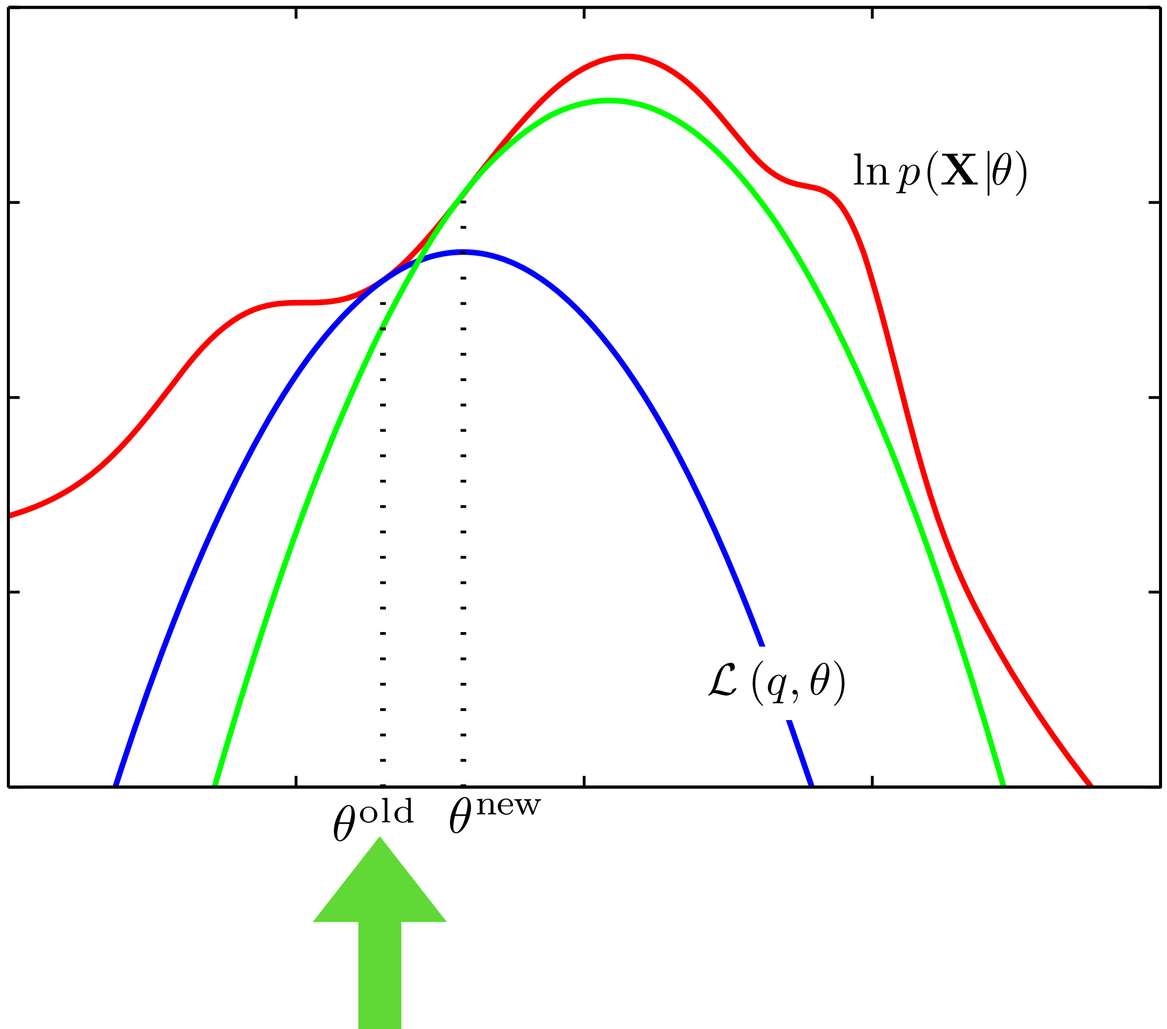


Illustration of the M step of the EM algorithm. The distribution $q(\mathbf{Z})$ is held fixed and the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$ is maximized with respect to the parameter vector $\boldsymbol{\theta}$ to give a revised value $\boldsymbol{\theta}^{\text{new}}$. Because the KL divergence is nonnegative, this causes the log likelihood $\ln p(\mathbf{X}|\boldsymbol{\theta})$ to increase by at least as much as the lower bound does.

EM over one cycle

The EM algorithm involves alternately computing a lower bound on the log likelihood for the current parameter values and then maximizing this bound to obtain the new parameter values. See the text for a full discussion.



For the particular case of an independent, identically distributed data set, $\mathbf{X}$ will comprise N data points $\{\mathbf{x}_n\}$ while $\mathbf{Z}$ will comprise N corresponding latent variables $\{\mathbf{z}_n\}$, where $n = 1, \ldots, N$. From the independence assumption, we have $p(\mathbf{X}, \mathbf{Z}) = \prod_n p(\mathbf{x}_n, \mathbf{z}_n)$ and, by marginalizing over the $\{\mathbf{z}_n\}$ we have $p(\mathbf{X}) = \prod_n p(\mathbf{x}_n)$. Using the sum and product rules, we see that the posterior probability that is evaluated in the E step takes the form

$$p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) = \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{\sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})} = \frac{\prod_{n=1}^{N} p(\mathbf{x}_n, \mathbf{z}_n|\boldsymbol{\theta})}{\sum_{\mathbf{Z}} \prod_{n=1}^{N} p(\mathbf{x}_n, \mathbf{z}_n|\boldsymbol{\theta})} = \prod_{n=1}^{N} p(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\theta}) \quad (9.75)$$

E-M in general

- Pros:
  - Forces you to explicitly choose your model form
  - EM is guaranteed to converge (for the exponential family and some others) to a (local) minimum and the log-likelihood is guaranteed to decrease at each iteration.
  - EM is more stable than arbitrary gradient descent (e.g., some complicated NN) because log likelihood is a pretty good function to optimize.
- Cons:
  - The optimal number of components K can be hard to find
  - Singularities. EM is prone to collapsed solutions (component k fits a single data point, $\sigma_k = 0$), causing the log-likelihood to blow up to infinity.
  - Parameter initialization matters as there may be local minima. To get reliable results comparing models you must average over different fits.