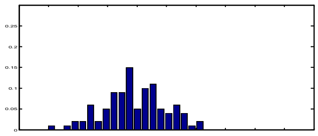
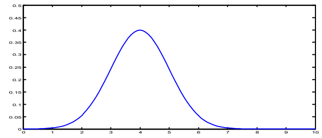
Density Estimation

Data histogram



Learned density

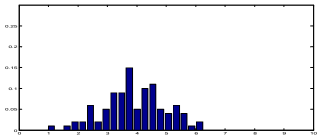


Model the density of the probability from which x is drawn. $\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{w})$, where $\boldsymbol{w}$ represents an unknown parameter vector that needs to be learned.

Decision theory, abnormal event detection.

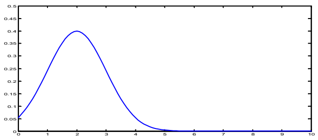
Density Estimation

Data histogram



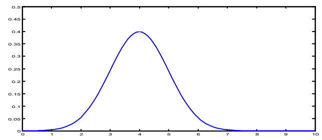
Which model is best?

Model 1



Why?

Model 2



Density Estimation

How do we obtain the model parameters?
Which model is the best?

Suppose we have a *family* of probability models for ***x*** *~ p(**x**;**w**),* where ***w*** represents an unknown parameter vector.

We will consider two forms of learning:

- o Maximum Likelihood
- o Bayesian learning

We will consider Gaussians models. Formally

Let $X = \{x_1, x_2, \ldots, x_N\}$ we wish to fit a density $p(x) = N_x(\mu, \Sigma)$.

$x_n \in R^d, \mu \in R^d, \Sigma \in R^{dxd}$

Maximum Likelihood estimation

Assuming samples are independent...

$$p(\chi \mid w) = \prod_{n=1}^{N} p(\mathbf{x_n} \mid \mathbf{w}) = L(\mathbf{w})$$

$L(\mathbf{w})$ is the likelihood of the observed data given the parameter $\mathbf{w}$

Intuitively choosing $\mathbf{w}$ that makes $L(\mathbf{w})$ large is good (makes the data more likely). So ...

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\text{argmax}}\ L(\mathbf{w}) \qquad \text{(maximum likelihood)}$$

we could argue (and will) that $p(\mathbf{x} \mid \mathbf{w})$ is not the relevant quantity (we know the data exists with probability 1!)

Maximum Likelihood fitting

In practice we minimise:

$$E \;=-\ln \mathrm{L}(\mathrm{w}) \;=\; -\sum_{\mathrm{n}} \ln p(\mathbf{x_n} \mid \mathbf{w})$$

the *negative log likelihood*.

General ML estimates will require *numerical optimization*.
Depending on model structure we could use:

- Gradient descent (back prop.)
- Analytically solve for $\frac{\nabla \mathrm{L}}{\nabla \mathrm{w}}=0$

or alternatives (not covered in ELEM041):

- Levenberg Marquardt
- Conjugate gradient

ML fitting a signle Gaussians

A simple example of *unsupervised learning* is fitting a Gaussian distribution to some data. In one dimension:

let $\boldsymbol{\chi} = \{x_1, x_2, \ldots, x_N\}$ we wish to fit a density $p(x) \;=\; N_{\mathrm{x}}(\mu, \sigma^2)$

$$E \;=\; -\ln\; L(\mathbf{w}) \;=\sum_{\mathrm{n}} \left(\frac{(x_n-\mu)^2}{2\sigma^2} + \ln\sigma + \frac{1}{2}\ln 2\pi \right).$$

Quadratic in μ, hence ML estimate $\hat{\mu}$:

$$\left.\frac{\partial E}{\partial \mu}\right|_{\hat{\mu}} = -\sum_{\mathrm{n}} \frac{(x_n-\hat{\mu})}{\sigma^2} = 0$$

$$\rightarrow \qquad \hat{\mu} = \frac{1}{N}\sum_n x_n \qquad\qquad \text{(sample mean!)}$$

ML fitting a single Gaussian

To solve for ML estimate of σ we use the fact that we already know $\hat{\mu}$:

$$\frac{\partial E}{\partial \sigma} = \sum_n \left(\frac{1}{\sigma} - \frac{(\mathbf{x}_n - \hat{\boldsymbol{\mu}})^2}{\sigma^3} \right)$$

Equating to zero gives :

$$\sigma^2 = \frac{1}{N} \sum_n (\mathbf{x}_n - \hat{\mu})^2 \qquad \text{(sample variance)}$$

Recall that the sample variance is a *biased* estimate for variance. However such ML estimates are consistent and asymptotically optimal.

ML fitting a signle Gaussians

In higher dimensions $p(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$

we get the similar results of:

ML estimate for mean $\hat{\boldsymbol{\mu}}$:

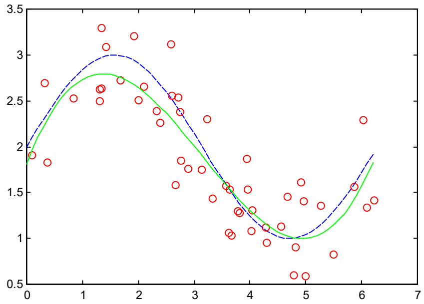
$$\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_n \mathbf{x_n} \qquad \text{(sample mean)}$$

and ML estimate for covariance matrix:

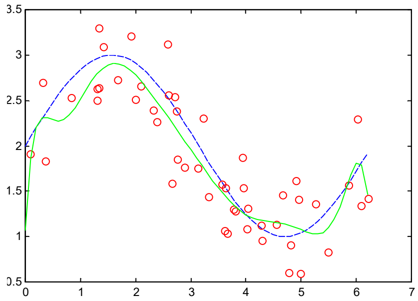
$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_n (\mathbf{x}_n - \hat{\boldsymbol{\mu}})(\mathbf{x}_n - \hat{\boldsymbol{\mu}})^T \qquad \text{(sample covariance)}$$

Overfitting in ML (regression)

Given a limited data set maximising $\mathcal{L}(\boldsymbol{w})$ may lead to *overfitting.* If the model order (dim. $\boldsymbol{w}$) is large enough we can 'exactly' fit model to data.



3rd order polynomial fit



10th order polynomial fit

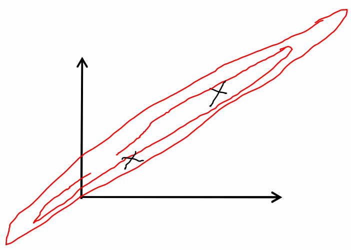
Overfitting in ML density estimation

Let $X = \{x_1, x_2, \ldots, x_N\}$ we wish to fit a density $p(x) = N_x(\mu, \Sigma)$.

$x_n \in \mathbb{R}^d, \mu \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d \times d}$

$$p(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

In the case that the data dimensionality is higher than the data samples the covariance matrix is singular. Non invertible, zero determinant.



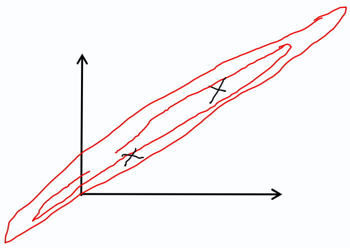
Overfitting in ML density estimation

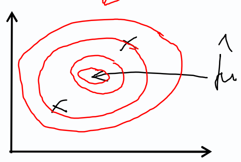
$$p(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

1. Put constraints in the form of the covariance matrix. E.g.

$$\Sigma = \sigma I = \begin{bmatrix} \sigma & & 0 \\ & \sigma & \\ 0 & & \sigma \end{bmatrix} \qquad \sigma^2 = \sum_n \left(x_n - \hat{\mu}\right)^T \left(x_n - \hat{\mu}\right)$$

Concentric circles





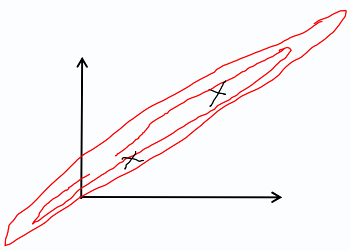
Overfitting in ML density estimation

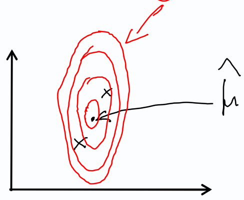
$$p(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

2. Put constraints in the form of the covariance matrix. E.g.

$$\Sigma = [\sigma_1 \;\; \dots \;\; \sigma_d] I = \begin{bmatrix} \sigma_1 & & \\ & \ddots \; \sigma_J \; \ddots & \\ & & \sigma_d \end{bmatrix} \qquad \sigma_j^2 = \sum_n \left(x_n(j) - \hat{\mu}(j)\right)^2$$

Axes aligned elipse





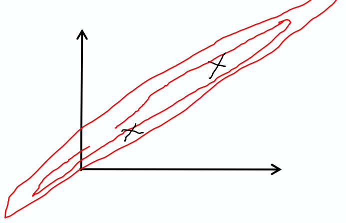
Overfitting in ML density estimation

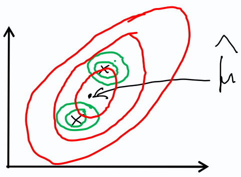
$$p(x\,|\,\mu,\Sigma)=\frac{1}{(2\pi)^{d/2}\,|\,\Sigma\,|^{1/2}}\,e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

2. Regularise the covariance matrix by adding a diagonal matrix $\Sigma_0=\sigma_0{}^2 I$

$$\hat{\Sigma}=\sum_n\left(x_n-\hat{\mu}\right)\left(x_n-\hat{\mu}\right)^T+\sigma_0{}^2 I=\begin{bmatrix}\sum_n\left(x_n(1)-\hat{\mu}_n(1)\right)^2+\sigma_0 & & \\ & \ddots & \\ & & \sum_n\left(x_n(d)-\mu_n(d)\right)^2+\sigma_0\end{bmatrix}$$

Leads to "thicker" Gaussian. Equivalent to adding zero mean Gaussian noise (with $\Sigma_0=\sigma_0{}^2 I$) to the original dataset.





Bayesian learning I

ML generates an estimator $\mathbf{w}$ for $\mathrm{p}(\mathrm{x})$ from a family of densities $\mathrm{p}(\mathbf{x}\,|\,\mathbf{w})$ by optimizing $\mathrm{L}(\mathbf{w})$. However the data already exists!

We are really interested in the probability of $\mathbf{w}$ given the data.
This extends the notion of RV to the parameter $\mathbf{w}$.

Bayes rule gives :

$$p(\mathbf{w}\,|\,\chi)=\frac{\mathrm{p}(\chi\,|\,\mathbf{w})\,\mathrm{p}(\mathbf{w})}{p(\chi)}$$

Requires a prior distribution on $\mathbf{w}$, $\mathrm{p}(\mathbf{w})$. Note also the role played by the likelihood, $\mathrm{p}(\chi\,|\,\mathbf{w})$.

Bayesian learning II

We can also evaluate the probability of new data

$$p(x^{\text{(new)}} \mid \chi) = \int p(x \mid \mathbf{w})\, p(\mathbf{w} \mid \chi)\, \mathrm{d}\,\mathbf{w}$$

Not just one value of w is used, instead a weighted average of values (this is the basis of the popular Kalman filter).

Bayesian Inference → Integration NOT optimization

In practice integrating and combining densities is difficult. One option is to use *conjugate priors* (**p(w)** is a conjugate prior if the posterior **p(w|χ)** has the same functional form – see the Gaussian example below).

Modern Bayesian methods have concentrated on using Monte Carlo integration (computationally intensive but very general).

Bayesian learning: 1-d Gaussians

Suppose $p(x \mid \mu) \sim N_x(\mu, \sigma^2)$ where σ^2 is known. We wish to estimate μ from data. Assume Gaussian prior on μ:

$$p(\mu) \sim N_\mu(\mu_0, {\sigma_0}^2)$$

(intuitively we believe that μ belongs within some range of values)

Inferring μ given some data : $\chi = \{x_1, x_2, ..., x_N\}$:

$$p(\mu \mid \chi) = \frac{p(\chi \mid \mu)\, p(\mu)}{p(\chi)}$$

Product of Gaussians = Gaussian (why?)

$$-\ln p(\mu \mid \chi) = \underbrace{\left[\sum_n \frac{(x_n - \mu)^2}{2\sigma^2}\right]}_{-\text{loglikelihood}} + \underbrace{\left[\frac{(\mu - \mu_0)^2}{2{\sigma_0}^2}\right]}_{-\text{logprior}} + \textit{const.}$$

Still quadratic in μ, hence Gaussian : $\mathrm{p}(\mu \mid \chi) = N_\mu(\mu_N, {\sigma_N}^2)$

Bayesian learning: 1-d Gaussians

Equating the quadratic terms, μ^2 :

$$\frac{\mu^2}{2\sigma_N^2} = \frac{N\mu^2}{2\sigma^2} + \frac{\mu^2}{2\sigma_0^2}$$

giving

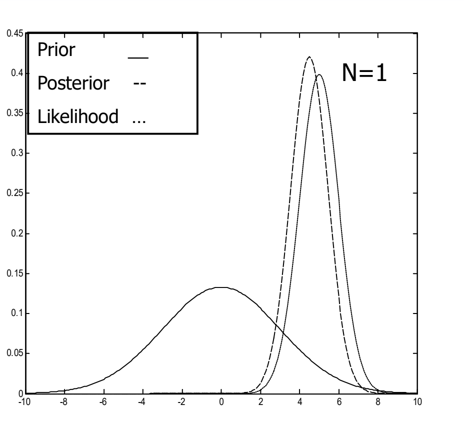
$$\sigma_N^2 = \left(\frac{N}{\sigma^2} + \frac{1}{\sigma_0^2}\right)^{-1}$$

and the linear terms, μ :

$$\frac{2\mu_N\mu}{2\sigma_N^2} = \frac{2\mu_0\mu}{2\sigma_0^2} + \frac{1}{2\sigma^2}\sum_n 2x_n\mu$$

giving :

$$\mu_N = \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2}\bar{x} + \frac{\sigma^2\mu_0}{N\sigma_0^2 + \sigma^2}$$



Bayesian choice of *w*

Often we may wish to get a single value for **w** as an estimate from some data. From a Bayesian perspective this involves defining a loss function (c.f. Bayesian decision theory). 2 common choices are:

1. *Posterior Mean*:

$$\mathbf{w} = E\{\mathbf{w} \mid \chi\} = \int \mathbf{w}\, \mathrm{p}(\mathbf{w} \mid \chi)\, \mathrm{d}\,\mathbf{w}$$

   this comes from a Mean Squared Error loss function.

2. *Maximum a Posteriori* (MAP) estimates:

$$\mathbf{w} = \underset{\mathbf{w}}{\operatorname{argmax}}\left[p(\mathbf{w} \mid \chi)\right]$$

   closely related to ML estimation – not really *very* Bayesian (comes from a 0-1 loss function)

The Bayes - ML link

We can see that ML and Bayesian learning are linked through the likelihood :

$$p(\mathbf{w} \mid \chi) \propto L(\mathbf{w})\, p(\mathbf{w})$$

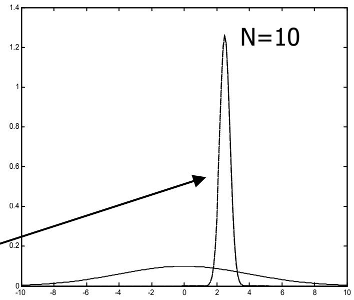
If prior is very 'weak' (i.e., σ_0^2 is large) then likelihood dominates.

Typically as $N \to \infty$

$$p(\mathbf{w} \mid \chi) \to L(\mathbf{w})$$

Also, as $N \to \infty$, $L(\mathbf{w})$ becomes very narrow (variance $\to 0$), so the maximum value characterises the distribution well.

Likelihood ≈ posterior



Gaussian Classifiers

- Assume some training data $X = \{(x_i, y_i\}$.
    - Produce a model y=f(x*) that can classify new data

$$P(y = j|x) = \frac{p(x|y = j)p(j)}{p(x)}$$

- Assign the new data point to class j iff

$$P(y = j|x) \geq P(y = i|x) \forall i \neq j$$

$$p(x|y = j)p(j) \geq p(x|y = i)p(i) \forall i \neq j$$

Gaussian Classifiers

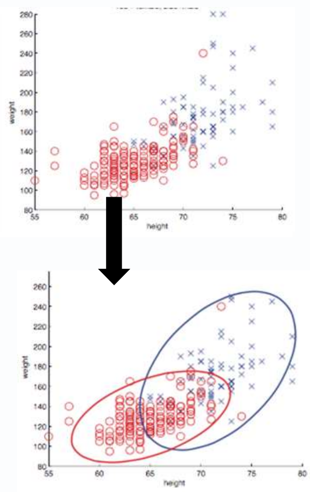
- Assign the new data point to class j iff

$$p(x|y=j)p(j) \geq p(x|y=i)p(i) \forall i \neq j$$

- How to get *p(x/y=i)*? Maximum Likelihood Estimation:
  - Fit a multivariate normal to $\{x_i\}_i$ where $y_i=1$
  - Fit a multivariate normal to $\{x_i\}_i$ where $y_i=2$

Maximum Likelihood Estimation

- Suppose we want to fit $p(x|\theta)$ given data $X=\{x_1,\ldots,x_n\}$:
  - Use samples X to estimate
  - Differentiate likelihood $p(X|\theta)$ wrt
  - Set to zero and solve for θ
  - In the case of Gaussians, this is:



$$\hat{u} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$$

$$\hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N} (\mathbf{x}_i - \hat{u})^T (\mathbf{x}_i - \hat{u})$$

Gaussian Conditional Classifier

- 1. Fit Gaussians from Data

$$\hat{u} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

$$\hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \hat{u})^T(\mathbf{x}_i - \hat{u})$$

- 2. Evaluate the posterior of each class for the new point, and pick the max

$$P(y = j|x) = \frac{p(x|y = j)p(j)}{p(x)}$$

Other Classifiers…

- Change the likelihood according to the data type:
- For continuous data: *p(x/w)* is commonly Gaussian.

$$\hat{u} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \qquad \hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \hat{u})^T(\mathbf{x}_i - \hat{u})$$

- For discrete data: *p(x/w)* is Bernoulli / multinomial
  - Common for: spam classifiers, sentiment recognition, etc.

$$\theta = \frac{1}{N}\sum_{i=1}^{N}x_i$$

$$w_{jk} = N_{jk} / \sum_j N_{jk}$$

$$N_{jk} = \sum_i I(x_{ik} = j)$$

Anomaly Detection: Motivation

- Sometimes you are interested in finding unusual items
  - "Anomalies", "Outliers"
- …As a pre-processing step
  - E.g., algorithms that use Sum-Squared objectives are not robust to outliers. Use outlier detection first to find and discard such rows
- …As an end goal.

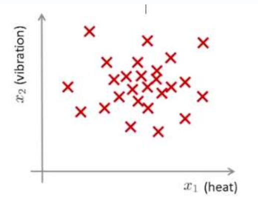
Anomaly Detection: Motivation

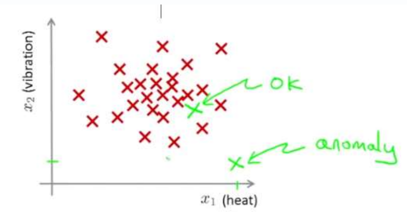
Applications

- Fraud detection
- Cyber-security
- Machine / Factory maintenance
- Data-center maintenance
- Process-control
- Security and surveillance
- Anti-terrorism

Anomaly Detection: Example

- Example: Manufacturing Quality Control: Aircraft Engines
  - x_1 = heat, x_2 = vibration.



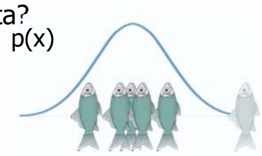


Anomaly Detection:

- Continuous variables: Gaussian
  - x is real vector. u is a real vector. S is a matrix.

$$p(\mathbf{x};\mathbf{u},S)=\frac{1}{Z}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{u})^T S^{-1}(\mathbf{x}-\mathbf{u})\right) \Rightarrow \mathbf{u}=\frac{1}{N}\sum_i \mathbf{x}_i \quad S=\frac{1}{N}\sum_i(\mathbf{x}-\mathbf{u})(\mathbf{x}-\mathbf{u})^T$$

- Tells us:
  - For a specified Gaussian:
    - What data do we expect?
    - How likely is any particular piece of data?
  - For a specified Dataset:
    - What Gaussian best explains it?
- Algorithm:
  - Anomaly if p(x)<T



Anomaly Detection Algorithm

- Algorithm:
  - Read in normal training data, {x}
  - Compute the Gaussian (u,S) that best explains the data {x}

$$\mathbf{u} = \frac{1}{N}\sum_i \mathbf{x}_i \qquad S = \frac{1}{N}\sum_i (\mathbf{x}-\mathbf{u})(\mathbf{x}-\mathbf{u})^T$$

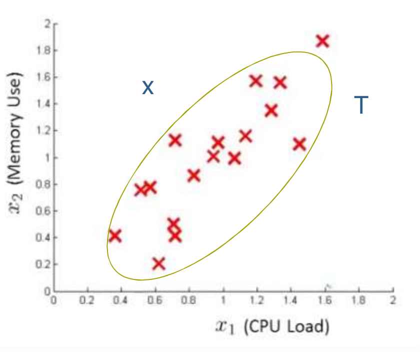
$$p(\mathbf{x};\mathbf{u},S) = \frac{1}{Z}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{u})^T S^{-1}(\mathbf{x}-\mathbf{u})\right)$$

  - Given a new example x and estimated u,S, compute p(x)
  - If p(x)<T, then Anomaly
    else Ok

Anomaly Detection Example: Data Center: Multivariate

Algorithm:

Anomaly if p(x)<T



Summary

- o Introduced the notion of Learning probability models.
- o Maximum Likelihood method:
  - – Solved through optimization
  - – There is a danger of overfitting
- o Bayesian method:
  - – Uses a distribution of parameters
  - – Full Bayesian estimators involve integration
  - – ML and Bayesian methods linked through likelihood function
- o Overfitting of models is a important practical concern