3F3: Signal and Pattern Processing

Lecture 2: Regression

Zoubin Ghahramani
zoubin@eng.cam.ac.uk

Department of Engineering
University of Cambridge

Lent Term

Regression

Let $\mathbf{x}$ denote an input point with elements $\mathbf{x} = (x_1, x_2, \ldots, x_D)$. The elements of $\mathbf{x}$, e.g. x_d, represent measured (observed) features of the data point; D denotes the number of measured features of each point.

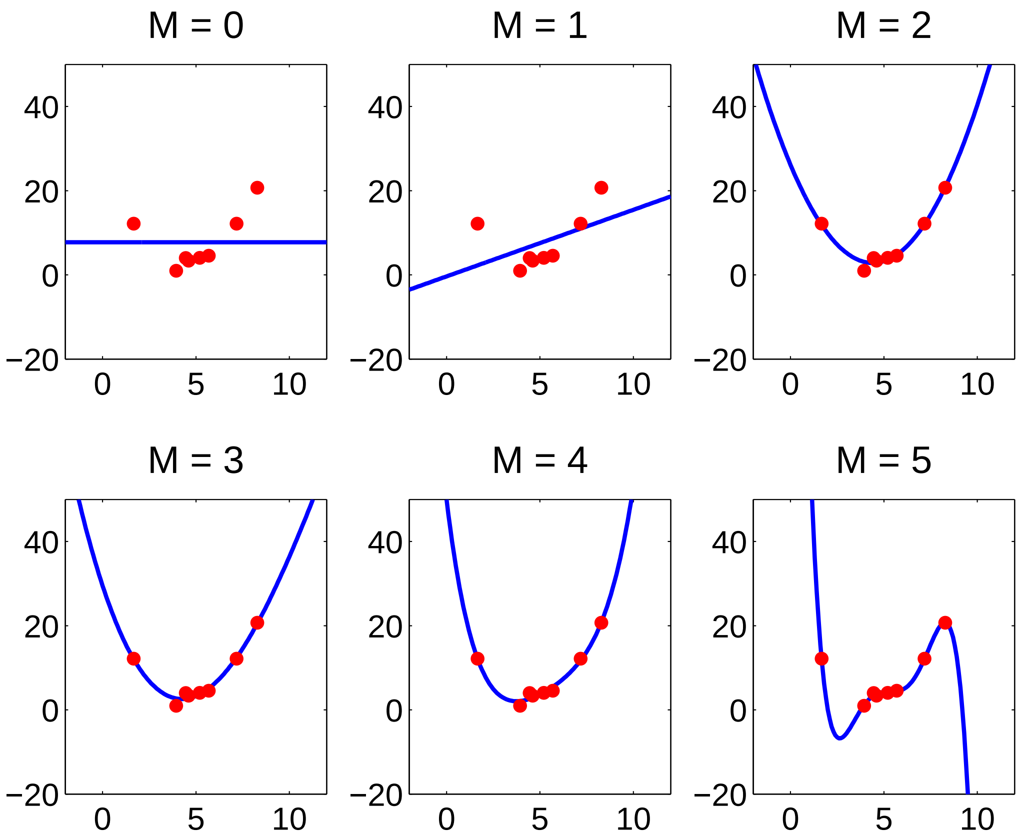
The data set $\mathcal{D}$ consists of N pairs of inputs and corresponding real-valued outputs:

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}) \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$$

where $y^{(n)} \in \Re$.

The goal is to predict with accuracy the output given a new input (i.e. to *generalize*).

Linear and Nonlinear Regression



The Simplest Case...

1-D linear regression, with inputs x and outputs y.

Observed data:

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}) \ldots, (x^{(N)}, y^{(N)})\}$$

Model:

$$y^{(n)} = ax^{(n)} + b$$

Is this reasonable? No!
Let's try again...

Model:

$$y^{(n)} = ax^{(n)} + b + \epsilon_n$$

where ϵ_n is a model for the noise.

- What distribution should we assume for the noise?

- How should we fit a and b?

The Simplest Case...

1-D linear regression, with inputs x and outputs y.

Observed data: $\mathcal{D} = \{(x^{(1)}, y^{(1)}) \ldots, (x^{(N)}, y^{(N)})\}$

Model: $y^{(n)} = ax^{(n)} + b + \epsilon_n$ where ϵ_n is a model for the noise. Assume:

- that the noise is Gaussian with mean zero and variance σ^2, and
- that the data was independently and identically distributed (iid) from this model.

Let $\mathbf{y} = (y^{(1)} \ldots, y^{(N)})$ and $\mathbf{x} = (x^{(1)} \ldots, x^{(N)})$. What is the probability of the observed outputs, $\mathbf{y}$, given the inputs, $\mathbf{x}$, and parameters $\theta = (a, b, \sigma^2)$?

$$P(\mathbf{y}|\mathbf{x}, \theta) = \prod_n P(y^{(n)}|x^{(n)}, a, b, \sigma^2)$$

where

$$P(y^{(n)}|x^{(n)}, a, b, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2\sigma^2}(y^{(n)} - ax^{(n)} - b)^2\right\}$$

$P(\mathbf{y}|\mathbf{x}, \theta)$ is known as the likelihood for the parameters θ.

Solving for Maximum Likelihood (ML) Parameters

The ML method is one way of finding a point estimate for the parameters:

$$\arg\max_\theta P(\mathbf{y}|\mathbf{x},\theta) \equiv \arg\max_\theta \ln P(\mathbf{y}|\mathbf{x},\theta) \stackrel{\text{def}}{=} \arg\max_\theta \mathcal{L}(\theta)$$

Let $\tilde{\mathbf{x}}^{(n)} = \begin{pmatrix} x^{(n)} \\ 1 \end{pmatrix}$ and $\beta = \begin{pmatrix} a \\ b \end{pmatrix}$.

$$\begin{aligned}
\mathcal{L}(\theta) &= \sum_n \ln P(y^{(n)}|x^{(n)}, a, b, \sigma^2) \\
&= -\frac{1}{2\sigma^2}\sum_n (y^{(n)} - \beta^\top \tilde{\mathbf{x}}^{(n)})^2 - \frac{N}{2}\ln(2\pi\sigma^2) \\
&= -\frac{1}{2\sigma^2}\left[\sum_n y^{(n)2} - 2\left(\sum_n y^{(n)}\tilde{\mathbf{x}}^{(n)\top}\right)\beta + \beta^\top\left(\sum_n \tilde{\mathbf{x}}^{(n)}\tilde{\mathbf{x}}^{(n)\top}\right)\beta\right] - \frac{N}{2}\ln(2\pi\sigma^2)
\end{aligned}$$

Solving for Maximum Likelihood (ML) Parameters...

Solve by taking derivatives and setting to zero...

$$\mathcal{L}(\theta) = -\frac{1}{2\sigma^2}\left[\sum_n y^{(n)2} - 2\left(\sum_n y^{(n)}\tilde{\mathbf{x}}^{(n)\top}\right)\beta + \beta^\top\left(\sum_n \tilde{\mathbf{x}}^{(n)}\tilde{\mathbf{x}}^{(n)\top}\right)\beta\right] - \frac{N}{2}\ln(2\pi\sigma^2)$$

$$\frac{\partial\mathcal{L}(\theta)}{\partial\beta} = \frac{1}{\sigma^2}\left[\left(\sum_n y^{(n)}\tilde{\mathbf{x}}^{(n)\top}\right) - \beta^\top\left(\sum_n \tilde{\mathbf{x}}^{(n)}\tilde{\mathbf{x}}^{(n)\top}\right)\right] = 0$$

$$\hat{\beta} = \left(\sum_n \tilde{\mathbf{x}}^{(n)}\tilde{\mathbf{x}}^{(n)\top}\right)^{-1}\left(\sum_n y^{(n)}\tilde{\mathbf{x}}^{(n)}\right)$$ "Normal Equations"

$$\frac{\partial\mathcal{L}(\theta)}{\partial\sigma} = \sigma^{-3}\left[\sum_n \ldots\right] - N\sigma^{-1} = 0$$

$$\hat{\sigma}^2 = \left[\sum_n \ldots\right]/N$$

Some Useful Rules for Derivatives wrt Vectors

- A very useful rule: $\dfrac{\partial\, \mathbf{a}^\top \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}^\top$

  We prove this simply by expanding out the dot product $\mathbf{a}^\top \mathbf{x} = \sum_i a_i x_i$, and using the convention that the derivative of a scalar wrt a column vector is a row vector:
  $\dfrac{\partial \sum_i a_i x_i}{\partial x_j} = a_j$

- Similarly we can use the above and the product rule to show that:

$$\frac{\partial \mathbf{x}^\top A \mathbf{x}}{\partial \mathbf{x}} = \mathbf{x}^\top A + \mathbf{x}^\top A^\top$$

  for symmetric $A = A^\top$ we have $\frac{\partial \mathbf{x}^\top A \mathbf{x}}{\partial \mathbf{x}} = 2\mathbf{x}^\top A$.

- The following is more tricky to prove: $\dfrac{\partial \ln |A|}{\partial A} = A^{-\top}$,
  where $|A|$ denotes the *determinant* of A, and A is square and invertible.[1]

Note, all the above are generalizations of the scalar cases: e.g. $\frac{\partial \ln a}{\partial a} = \frac{1}{a}$.

[1] If A is symmetric then $\frac{\partial \ln |A|}{\partial A} = 2A^{-1} - \mathrm{diag}(\mathrm{A}^{-1})$.

Linear Regression with Vector Valued Inputs

Inputs $\mathbf{x} \in \Re^D$

Model:

$$\begin{aligned} y^{(n)} &= \beta_0 + \beta_1 x_1^{(n)} + \ldots \beta_D x_D^{(n)} + \epsilon_n \\ y^{(n)} &= \beta^\top \tilde{\mathbf{x}}^{(n)} + \epsilon_n \end{aligned}$$

where ϵ_n is Gaussian noise, and $\tilde{\mathbf{x}}^{(n)} = \begin{pmatrix} 1 \\ \mathbf{x}^{(n)} \end{pmatrix}$.

Easy!

We've solved this already.

Polynomial (Nonlinear) Regression

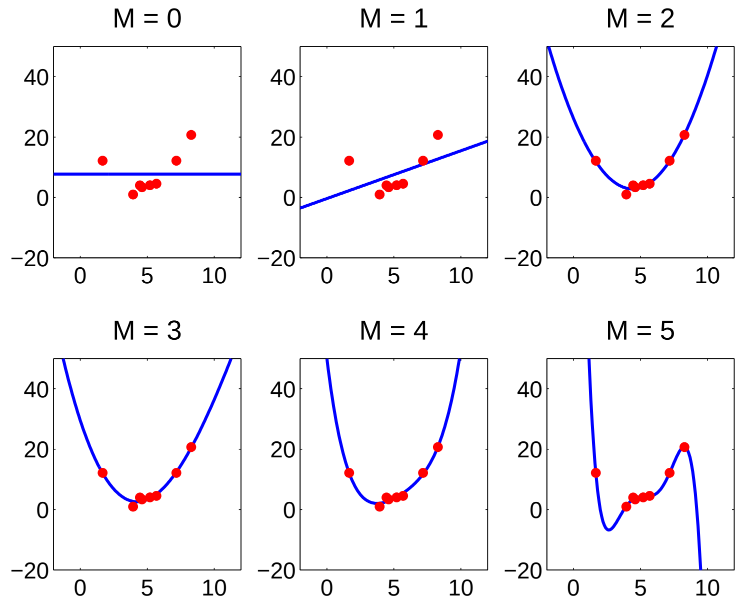
M^{th} Order Polynomial Model:

$$y^{(n)} = \beta_0 + \beta_1 x^{(n)} + \beta_2 x^{(n)2} + \dots \beta_M x^{(n)M} + \epsilon_n$$

where ϵ_n is Gaussian noise.

Easy!

We've solved this already as well.



Basis Function (Nonlinear) Regression

Instead of using polynomial bases, we can use all kinds of other bases (Gaussian radial basis functions, sinusoids, wavelets, splines, etc).

K Basis Function Model:

$$y^{(n)} = \sum_{k=1}^{K} \beta_k \phi_k(\mathbf{x}^{(n)}) + \epsilon_n$$

where ϕ_k can be any function, and ϵ_n is Gaussian noise.

For example, Gaussian radial basis functions with center $\mathbf{c}_k$ and width s_k:

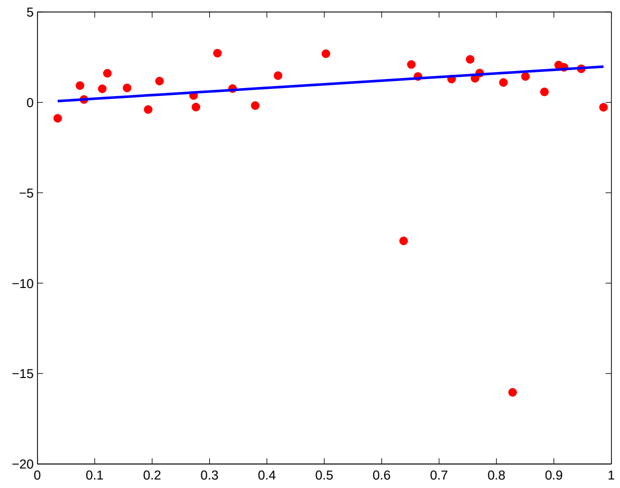
$$\phi_k(\mathbf{x}) = \exp\{-\frac{1}{2s_k}\|\mathbf{x} - \mathbf{c}_k\|^2\}$$

Easy! We've solved this already as well. Let $\tilde{\mathbf{x}} \equiv \boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}) \ldots \phi_K(\mathbf{x}))$.

$$\hat{\beta} = \left(\sum_n \tilde{\mathbf{x}}^{(n)} \tilde{\mathbf{x}}^{(n)\top}\right)^{-1} \left(\sum_n y^{(n)} \tilde{\mathbf{x}}^{(n)}\right)$$

related to "kernel trick"

Linear Regression with Non-Gaussian Noise



Model:

$$y^{(n)} = \beta^\top \mathbf{x}^{(n)} + \epsilon_n$$

where ϵ_n is non-Gaussian noise.

For example, we might want to use heavy-tailed noise to model outliers.

Gaussian noise:

$$P(y^{(n)}|\mathbf{x}^{(n)}, \beta, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2\sigma^2}(y^{(n)} - \beta^\top \mathbf{x}^{(n)})^2\right\}$$

vs **Laplacian noise:**

$$P(y^{(n)}|\mathbf{x}^{(n)}, \beta, \sigma) = \frac{1}{Z} \exp\left\{-\frac{1}{\sigma}\left|y^{(n)} - \beta^\top \mathbf{x}^{(n)}\right|\right\}$$

Bayesian Learning

Apply the basic rules of probability to learning from data.

Data set: $\mathcal{D} = \{y_1, \ldots, y_n\}$ Models: m, m' etc. Model parameters: θ

Prior probability of models: $P(m)$, $P(m')$ etc.

Prior probabilities of model parameters: $P(\theta|m)$

Model of data given parameters (likelihood model): $P(y|\theta, m)$

If the data are independently and identically distributed then:

$$P(\mathcal{D}|\theta, m) = \prod_{i=1}^{n} P(y_i|\theta, m)$$

Posterior probability of model parameters:

$$P(\theta|\mathcal{D}, m) = \frac{P(\mathcal{D}|\theta, m)P(\theta|m)}{P(\mathcal{D}|m)}$$

Posterior probability of models:

$$P(m|\mathcal{D}) = \frac{P(m)P(\mathcal{D}|m)}{P(\mathcal{D})}$$

Maximum Likelihood, Maximum A Posteriori, Regularization, and Bayesian Learning

Maximum Likelihood:

$$\hat{\theta} = \arg\max_{\theta} P(\mathcal{D}|\theta, m)$$

Maximum A Posteriori: Assume a prior $P(\theta|m)$

$$\hat{\theta} = \arg\max_{\theta} P(\theta|\mathcal{D}, m) = \arg\max_{\theta} [\ln P(\mathcal{D}|\theta, m) + \ln P(\theta|m)]$$

Regularization:

$$\hat{\theta} = \arg\min_{\theta} [\ell(\mathcal{D}, \theta) + \lambda R(\theta)]$$

where $\ell(\mathcal{D}, \theta)$ is some loss on the training data, $\lambda > 0$, and $R(\theta)$ is called a regularizer for θ, e.g. $R(\theta) = \|\theta\|^2$.

Bayesian Learning:

$$P(\theta|\mathcal{D}, m) = \frac{P(\mathcal{D}|\theta, m)P(\theta|m)}{P(\mathcal{D}|m)}$$