These slides follow closely the (English) course textbook

*Pattern Recognition and Machine Learning*

by Christopher Bishop
Linear Basis Function Models

• Linear regression extended to consider fixed basis functions:

\[
y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x)
\]

where \( w = (w_0, ..., w_{M-1})^T \) and \( \phi = (\phi_0, ..., \phi_{M-1})^T \)

• Possible basis functions include polynomials, Fourier, wavelet, ...

\[
\phi_j(x) = x^j
\]

\[
\phi_j(x) = \exp\left\{\frac{(x - \mu_j)^2}{2s^2}\right\}
\]

where \( \sigma(a) = \frac{1}{1 + \exp(-a)} \)

Polynomial

Gaussian

Sigmoidal
Maximum likelihood and least squares

• Presume target $t$ is generated via deterministic function plus gaussian noise $\epsilon$ having precision $\beta$

$$t = y(x, w) + \epsilon$$

$$p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1})$$

• With Gaussian conditional distribution conditional mean is:

$$\mathbb{E}[t|x] = \int tp(t|x)dt = y(x, w)$$

• With a set of input points $X = \{x_1, ..., x_N\}$ independently drawn:

$$p(t|X, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|w^T \phi(x_n), \beta^{-1})$$
Maximum likelihood and least squares

• Log likelihood:

\[
\ln p(t|w, \beta) = \sum_{n=1}^{N} \ln \mathcal{N}(t_n|w^T \phi(x_n), \beta^{-1})
\]

\[
= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(w)
\]

where \( E_D(w) = \frac{1}{2} \sum_{n=1}^{N} \{t_n - w^T \phi(x_n)\}^2 \)

• Gradient:

\[
\nabla \ln p(t|w, \beta) = \sum_{n=1}^{N} \{t_n - w^T \phi(x_n)\} \phi(x_n)^T
\]
Maximum likelihood and least squares

- Set gradient to 0:
  
  \[
  0 = \sum_{n=1}^{N} t_n \phi(x_n)^T - w^T \left( \sum_{n=1}^{N} \phi(x_n) \phi(x_n)^T \right)
  \]

- Solve for our weights yields *normal equations* for least squares:
  
  \[
  w_{ML} = (\Phi^T \Phi)^{-1} \Phi^T t
  \]

- where \( \Phi \) is the design matrix
  
  \[
  \Phi = \begin{pmatrix}
  \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \\
  \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_{M-1}(x_2) \\
  \vdots & \vdots & \ddots & \vdots \\
  \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_{M-1}(x_N)
  \end{pmatrix}
  \]

- and where \( \Phi^\dagger = (\Phi^T \Phi^{-1}) \Phi^T \) is the pseudo-inverse of \( \Phi \)
Geometry of least squares

• Least-squares regression is obtained by finding the orthogonal projection of the data vector \( \mathbf{t} \) onto the subspace spanned by the basis functions.

• Intuition: Sum of squares error is 1/2 squared Euclidean distance between \( \mathbf{y} \) and \( \mathbf{t} \). Thus least-squares solution would move \( \mathbf{t} \) as close as possible to \( \mathbf{y} \) in the subspace \( S \).
Online learning

• For large datasets may need to learn sequentially on sequences of smaller datasets, summing error

\[ E = \sum_{n} E_n \]

• Sequential gradient descent (also called stochastic gradient descent):

\[ w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_n \]

where \( \tau \) is the iteration number and \( \eta \) is the learning rate

• For sum-of-squares we get Least Mean Squares (LMS):

\[ w^{(\tau+1)} = w^{(\tau)} + \eta (t_n - w^{(\tau)T} \phi_n) \phi_n \]

• Learning rate must be chosen carefully
Regularized least squares

- Regularize magnitude of weights:
  \[ \frac{1}{2} \sum_{n=1}^{N} \left( t_n - w^T \phi(x_n) \right)^2 + \frac{\lambda}{2} w^T w \]

- Gradient with respect to 0 yields extension of least squares:
  \[ w = \left( \lambda I + \Phi^T \Phi \right)^{-1} \Phi^T t \]

- More general regularizer; when q=1 we have “lasso” regularizer which selects for sparse models:
  \[ \frac{1}{2} \sum_{n=1}^{N} \left( t_n - w^T \phi(x_n) \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{M} |w_j|^q \]
Visualization of regularized least squares

Plot of contours of unregularized error function along with constraint region on the quadratic regularizer (left, $q=2$) versus lasso regularizer (right, $q=1$). For lasso, a sparse solution is generated with $w_1=0$. 

$w_1$ $w_2$ $w_1$ $w_2$ $w^*$ $w^*$
Bias-Variance decomposition

• How to best set the $\lambda$ parameter for regularization?

• Conditional expectation:

$$h(x) = \mathbb{E}[t|x] = \int tp(t|x)dt$$

• Expected squared loss written with noise as second term:

$$\mathbb{E}[L] = \int \{y(x) - h(x)\}^2 p(x) \, dx + \int \{h(x) - t\}^2 p(x, t) \, dx \, dt$$

• We will minimize the first term. But we cannot hope to ever know the perfect regression function $h(x)$

• In a Bayesian model uncertainty is expressed as posterior over $w$

• In frequentist treatment we make a point-estimate of $w$. Assess confidence by making predictions over subsets of data and taking mean mean performance.
Bias-Variance decomposition

• Take integrand of first term using some subset of data $D$.
  $\{y(x, D) - h(x)\}^2$ which varies with data, thus take its mean

• Add and subtract expected value for the data
  $\{y(x; D) - \mathbb{E}_D[y(x; D)] + \mathbb{E}_D[y(x; D)] - h(x)\}^2$
  $= \{y(x; D) - \mathbb{E}_D[y(x; D)]\}^2 + \{\mathbb{E}_D[y(x; D)] - h(x)\}^2$
  $+ 2\{y(x; D) - \mathbb{E}_D[y(x; D)]\}\{\mathbb{E}_D[y(x; D)] - h(x)\}$

• Take expectation with respect to $D$; final term vanishes
  $\mathbb{E}_D[\{y(x; D) - h(x)\}^2]$ 
  $= \{\mathbb{E}_D[y(x; D)] - h(x)\}^2 + \mathbb{E}_D[\{y(x; D) - \mathbb{E}_D[y(x; D)]\}^2]$ 
  $\text{(bias)}^2$  \hspace{1cm}  variance

• First term is bias: extent to which average prediction differs from desired regression function

• Second term is variance: extent to which individual solutions vary around the average. Thus measures sensitivity to data.
Bias-Variance decomposition

- expected loss = (bias)$^2$ + variance + noise where:

  \[
  (\text{bias})^2 = \int \{ \mathbb{E}_D[y(x; D)] - h(x) \}^2 p(x) \, dx
  \]

  \[
  \text{variance} = \int \mathbb{E}_D[\{y(x; D) - \mathbb{E}_D[y(x; D)]\}^2] p(x) \, dx
  \]

  \[
  \text{noise} = \int \{ h(x) - t \}^2 p(x, t) \, dx \, dt
  \]

- Very flexible models have low bias and high variance
- Relatively rigid models have high bias and low variance
- Optimal model balances the two
100 datasets each with 25 data points. Fit with 25 Gaussian basis functions.

Regularization parameter $\lambda$ is varied. Top are individual fits. Bottom is average fit along with generating sine function in green.
Bias variance example

average \quad \bar{y}(x) = \frac{1}{L} \sum_{l=1}^{L} y^{(l)}(x)

(bias)^2 \quad = \quad \frac{1}{N} \sum_{n=1}^{N} \{ \bar{y}(x_n) - h(x_n) \}^2

variance \quad = \quad \frac{1}{N} \sum_{n=1}^{N} \frac{1}{L} \sum_{l=1}^{L} \{ y^{(l)}(x_n) - \bar{y}(x_n) \}^2

Plot of squared bias and variance together with their sum. The minimum is at \( \lambda = -0.31 \) which is close to the value yielding minimum test error.
Bayesian Linear Regression

- Bias-variance decomposition requires splitting data. Inefficient.
- Avoids overfitting of maximum likelihood
- Leads to automatic way of determining model complexity
- Now we look quickly at Bayesian approach. Will return to it later in semester.
- Define prior over weights using zero-mean Gaussian prior:
  \[ p(w|\alpha) = \mathcal{N}(w|0, \alpha^{-1}I) \]
- Log of posterior is sum of log likelihood and log of prior:
  \[
  \ln p(w|t) = -\frac{\beta}{2} \sum_{n=1}^{N} \{t_n - w^T\phi(x_n)\}^2 - \frac{\alpha}{2} w^Tw + \text{const}
  \]
  with quadratic regularization term \( \lambda = \alpha/\beta \) in least squares sense
Sequential Bayesian Learning

- Consider simple input variable $x$, single target $t$ and a linear model of form $y(x, w) = w_0 + w_1 x$

- Just two weights, can plot prior and posteriors

- Generate synthetic data using $f(x_n, a) = -0.3 + 0.5x_n + \epsilon$

- Goal is to recover $a = \{-0.3, 0.5\}$ from data

- Basic algorithm:
  - observe point $(x,t)$ from dataset
  - calculate likelihood $p(t|x,w)$ based on estimate of noise precision $\beta$
  - multiply likelihood by previous prior over $w$ to yield new posterior
Sequential Bayesian Learning

**Basic algorithm:**

- Observe point \((x,t)\) from dataset
- Calculate likelihood \(p(t|x,w)\) based on estimate of noise variance \(\beta\)
- Multiply likelihood by previous prior over \(w\) to yield new posterior
- Observe another point ...

Samples from posterior are shown on right
Predictive distribution

• We are generally not interested in priors over $w$ but rather for predicting new values $t$ from $x$.

• Evaluate predictive distribution

\[
p(t|t, \alpha, \beta) = \int p(t|w, \beta)p(w|t, \alpha, \beta) \, dw
\]

• This is convolution of conditional distribution of target with posterior $w$. For our problem (2 Gaussians) results in:

\[
p(t|x, t, \alpha, \beta) = \mathcal{N}(t|m_N^T \phi(x), \sigma_N^2(x))
\]

\[
\sigma_N^2(x) = \frac{1}{\beta} + \phi(x)^T S_N \phi(x)
\]
Predictive distributions for 9 Gaussian basis functions fitting $f(x) = \sin(2\pi x) + \epsilon$ in green. Red curve is mean of predictive distributions. Red shaded regions are 1 std dev. away from mean.
Plots of the functions $y(x, \mathbf{w})$ using samples from the posterior distributions over $\mathbf{w}$ corresponding to the previous plots.
Equivalent kernel

- Posterior means can be interpreted as kernels; sets stage for kernel methods including Gaussian processes.

- Predictive mean can be written as:

\[ y(x, m_N) = m_N^T \phi(x) = \beta \phi(x)^T S_N \Phi^T t = \sum_{n=1}^{N} \beta \phi(x)^T S_N \phi(x) t_n \]

- We can also rewrite this as a kernel function:

\[ y(x, m_N) = \sum_{n=1}^{N} k(x, r_b x_n) t_n \]

where the function \( k(x, x') = \beta \phi(x)^T S_N \phi(x') \) is known as the smoother matrix or equivalent kernel

- Regression functions which predict using linear combinations of target values are known as linear smoothers
• Above $k(x,x')$ is plotted as a function of $x$. Note that it is localized around $x$.

• Mean of predictive distribution at $x$ given by $y(x,m_N)$ is obtained using a weighted combination where points close to $x$ are given higher weight.

• Idea of using a localized kernel in place of a set of basis functions leads to Gaussian processes (to be covered later).