Probability topics

- Probability as representation of uncertainty in models & data
- Bayes Theorem and its applications
- Law of large numbers and the Gaussian distribution
- Markov and graphical models
Bayes’ theorem for classification:

\[
P(\text{label}|\text{data}) = \frac{P(\text{data}|\text{label})}{P(\text{data})}\]

Training examples give \(P(\text{data}|\text{label})\).

- Understand the symbols \(P(A|B)\) as shorthand for \(P(a_i \in A|b_k \in B)\).
Bayes’ theorem for classification:

\[ P(\text{label}|\text{data}) = \frac{P(\text{data}|\text{label})}{P(\text{data})} \]

Training examples give \( P(\text{data}|\text{label}) \).

- Understand the symbols \( P(A|B) \) as shorthand for \( P(a_i \in A|b_k \in B) \).
- Distributions \( P(\text{data}|\text{label}) \) (discrete and continuous)
Probability: revision

Bayes’ theorem for classification:

\[
P(label|data) = \frac{P(data|label)}{P(data)}
\]

Training examples give \( P(data|label) \).

- Understand the symbols \( P(A|B) \) as shorthand for \( P(a_i \in A|b_k \in B) \).
- Distributions \( P(data|label) \) (discrete and continuous)
  - uniform, Gaussian, Beta
Bayes’ theorem for classification:

\[ P(\text{label}|\text{data}) = \frac{P(\text{data}|\text{label})}{P(\text{data})} \]

Training examples give \( P(\text{data}|\text{label}) \).

- Understand the symbols \( P(A|B) \) as shorthand for \( P(a_i \in A|b_k \in B) \).
- Distributions \( P(\text{data}|\text{label}) \) (discrete and continuous)
  1. uniform, Gaussian, Beta
  2. Empirical distribution – histograms
Bayes’ theorem for classification:

\[ P(\text{label}|\text{data}) = \frac{P(\text{data}|\text{label})}{P(\text{data})} \]

Training examples give \( P(\text{data}|\text{label}) \).

- Understand the symbols \( P(A|B) \) as shorthand for \( P(a_i \in A|b_k \in B) \).
- Distributions \( P(\text{data}|\text{label}) \) (discrete and continuous)
  1. uniform, Gaussian, Beta
  2. Empirical distribution – histograms
  3. Descriptive statistics – mean, std. dev., variance
Bayes’ theorem for classification:

\[ P(\text{label}|\text{data}) = \frac{P(\text{data}|\text{label})}{P(\text{data})} \]

Training examples give \( P(\text{data}|\text{label}) \).

- Understand the symbols \( P(A|B) \) as shorthand for \( P(a_i \in A|b_k \in B) \).
- Distributions \( P(\text{data}|\text{label}) \) (discrete and continuous)
  1. uniform, Gaussian, Beta
  2. Empirical distribution – histograms
  3. Descriptive statistics – mean, std. dev., variance
- Priors and posteriors for binomial, multinomial (pseudo-counts) and Gaussian
Bayes’ theorem for classification:

\[
P(\text{label}|\text{data}) = \frac{P(\text{data}|\text{label})}{P(\text{data})}
\]

Training examples give \(P(\text{data}|\text{label})\).

- Understand the symbols \(P(A|B)\) as shorthand for \(P(a_i \in A|b_k \in B)\).
- Distributions \(P(\text{data}|\text{label})\) (discrete and continuous)
  1. uniform, Gaussian, Beta
  2. Empirical distribution – histograms
  3. Descriptive statistics – mean, std. dev., variance
- Priors and posteriors for binomial, multinomial (pseudo-counts) and Gaussian
- CLT: interpretation of parameters of Gaussian \(\mathcal{N}(\mu, \Sigma)\) in terms of data generating distribution
Bayes’ theorem for classification:

\[
P(label|data) = \frac{P(data|label)}{P(data)}
\]

Training examples give \(P(data|label)\).

- Understand the symbols \(P(A|B)\) as shorthand for \(P(a_i \in A|b_k \in B)\).
- Distributions \(P(data|label)\) (discrete and continuous)
  1. uniform, Gaussian, Beta
  2. Empirical distribution – histograms
  3. Descriptive statistics – mean, std. dev., variance
- Priors and posteriors for binomial, multinomial (pseudo-counts) and Gaussian
- CLT: interpretation of parameters of Gaussian \(\mathcal{N}(\mu, \Sigma)\) in terms of data generating distribution
- Metrics on distributions – entropy, relative entropy, KL divergence.
  Interpretation.
Bayes’ theorem for classification:

\[
P(\text{label}|\text{data}) = \frac{P(\text{data}|\text{label})}{P(\text{data})}
\]

Training examples give \( P(\text{data}|\text{label}) \).

- Understand the symbols \( P(A|B) \) as shorthand for \( P(a_i \in A|b_k \in B) \).
- Distributions \( P(\text{data}|\text{label}) \) (discrete and continuous)
  1. uniform, Gaussian, Beta
  2. Empirical distribution – histograms
  3. Descriptive statistics – mean, std. dev., variance
- Priors and posteriors for binomial, multinomial (pseudo-counts) and Gaussian
- CLT: interpretation of parameters of Gaussian \( \mathcal{N}(\mu, \Sigma) \) in terms of data generating distribution
- Metrics on distributions – entropy, relative entropy, KL divergence. Interpretation.
- Learning: Maximum Likelihood Estimation (optimisation with Lagrange multipliers); minimise KL divergence
Supervised Learning

- Classification using Bayesian principles
- Perceptron Learning
- Neural networks/multi-layer perceptrons
- Features and discriminant analysis
- Logistic regression
Problem formulation: $y = f(x; w)$. Find $w$ so that $\sum_n \|y^n - f(x^n; w)\|^2$ is minimised.
Supervised Learning: Revision

- Problem formulation: $\mathbf{y} = f(\mathbf{x}; \mathbf{w})$. Find $\mathbf{w}$ so that $\sum_n \| \mathbf{y}^n - f(\mathbf{x}^n; \mathbf{w}) \|^2$ is minimised.

- Note: appearance of quadratic form often re-expressed as $\mathbf{v}^T \mathbf{A} \mathbf{v}$.
Supervised Learning: Revision

- Problem formulation: $y = f(x; \mathbf{w})$. Find $\mathbf{w}$ so that $\sum_n \|y^n - f(x^n; \mathbf{w})\|^2$ is minimised.

- Note: appearance of quadratic form often re-expressed as $\mathbf{v}^T \mathbf{A} \mathbf{v}$.
  - Solution: algorithms for perceptrons, neural nets follow from the problem formulation. Use calculus to solve optimisation problem: gradient descent, back-propagation.
Problem formulation: $y = f(x; \mathbf{w})$. Find $\mathbf{w}$ so that $\sum_n \|y^n - f(x^n; \mathbf{w})\|^2$ is minimised.

Note: appearance of quadratic form often re-expressed as $\mathbf{v}^T \mathbf{A} \mathbf{v}$.

- Solution: algorithms for perceptrons, neural nets follow from the problem formulation. Use calculus to solve optimisation problem: gradient descent, back-propagation.

Formulation: Find class label that maximises $P(\text{label}|\text{data})$ using Bayes’ theorem. Where do we place decision boundaries in the data space?
Problem formulation: \( \mathbf{y} = f(\mathbf{x}; \mathbf{w}) \). Find \( \mathbf{w} \) so that \( \sum_n \| \mathbf{y}^n - f(\mathbf{x}^n; \mathbf{w}) \|^2 \) is minimised.

Note: appearance of quadratic form often re-expressed as \( \mathbf{v}^\top \mathbf{A} \mathbf{v} \).

▶ Solution: algorithms for perceptrons, neural nets follow from the problem formulation. Use calculus to solve optimisation problem: gradient descent, back-propagation.

Formulation: Find class label that maximises \( P(\text{label}|\text{data}) \) using Bayes’ theorem. Where do we place decision boundaries in the data space?

▶ Solution: condition for boundary \( f(\mathbf{x}) = \log \left( \frac{P(\text{class}=a|\mathbf{x})}{P(\text{class}=b|\mathbf{x})} \right) = 0 \). Note: we have seen \( \mathbf{x}^n \), but we want to generalise to all \( \mathbf{x} \).
Problem formulation: \( y = f(x; \omega) \). Find \( \omega \) so that \( \sum_n \| y^n - f(x^n; \omega) \|^2 \) is minimised.

Note: appearance of quadratic form often re-expressed as \( v^T A v \).

Solution: algorithms for perceptrons, neural nets follow from the problem formulation. Use calculus to solve optimisation problem: gradient descent, back-propagation.

Formulation: Find class label that maximises \( P(\text{label}|\text{data}) \) using Bayes’ theorem. Where do we place decision boundaries in the data space?

Solution: condition for boundary \( f(x) = \log \left( \frac{P(\text{class}=a|x)}{P(\text{class}=b|x)} \right) = 0 \). Note: we have seen \( x^n \), but we want to generalise to all \( x \).

Special cases: Find the (linear, quadratic) discriminant assuming data in each class \( P(x|\text{class}) \) can be fit to Gaussian.
Formulation: find discriminating directions in high dimensional data. Find directions \( \mathbf{w} \) along which projections of data \( x^n \) to \( y^n = \mathbf{w}^T x^n \) maximises class separation.

\[
\frac{|\mu_a - \mu_b|^2}{n_a \sigma_a^2 + n_b \sigma_b^2}
\]

\( \mu_a, \mu_b, \sigma_a^2, \sigma_b^2 \) means and variances of \( y^n \) in each class.
Supervised Learning (Fisher's LDA) : Revision (cont'd).

Formulation: find discriminating directions in high dimensional data. Find directions $\mathbf{w}$ along which projections of data $\mathbf{x}^n$ to $\mathbf{y}^n = \mathbf{w}^T \mathbf{x}^n$ maximises class separation.

$$\frac{|\mu_a - \mu_b|^2}{n_a \sigma_a^2 + n_b \sigma_b^2}$$

$\mu_a, \mu_b, \sigma_a^2, \sigma_b^2$ means and variances of $\mathbf{y}^n$ in each class.

Finding algorithm: re-express with $\mathbf{w}$ explicit

$$\frac{|\mu_a - \mu_b|^2}{n_a \sigma_a^2 + n_b \sigma_b^2} \leftrightarrow \frac{\mathbf{w}^T \Sigma_B \mathbf{w}}{\mathbf{w}^T \Sigma_W \mathbf{w}} = F(\mathbf{w})$$

$$\mu_a = \frac{1}{n_a} \sum y^n = \frac{1}{n_a} \sum (\mathbf{w}^T \mathbf{x}^n) = \mathbf{w}^T \mathbf{m}_a$$ for data from class $a$.

$$\Sigma_B = (\mathbf{m}_a - \mathbf{m}_b)(\mathbf{m}_a - \mathbf{m}_b)^T, \text{ etc.}$$
Supervised Learning (Fisher’s LDA) : Revision (cont’d).

- Formulation: find discriminating directions in high dimensional data. Find directions \( \mathbf{w} \) along which projections of data \( \mathbf{x}^n \) to \( \mathbf{y}^n = \mathbf{w}^T \mathbf{x}^n \) maximises class separation.

\[
\frac{|\mu_a - \mu_b|^2}{n_a \sigma_a^2 + n_b \sigma_b^2}
\]

\( \mu_a, \mu_b, \sigma_a^2, \sigma_b^2 \) means and variances of \( \mathbf{y}^n \) in each class.

- Finding algorithm: re-express with \( \mathbf{w} \) explicit

\[
\frac{|\mu_a - \mu_b|^2}{n_a \sigma_a^2 + n_b \sigma_b^2} \leftrightarrow \frac{\mathbf{w}^T \Sigma_B \mathbf{w}}{\mathbf{w}^T \Sigma_W \mathbf{w}} = F(\mathbf{w})
\]

\( \mu_a = \frac{1}{n_a} \sum y^n = \frac{1}{n_a} \sum (\mathbf{w}^T \mathbf{x}^n) = \mathbf{w}^T \mathbf{m}_a \) for data from class \( a \).

\( \Sigma_B = (\mathbf{m}_a - \mathbf{m}_b)(\mathbf{m}_a - \mathbf{m}_b)^T \), etc.

- Using explicit \( \mathbf{w} \) dependence, calculus solves optimisation problem:

\( \mathbf{w} = \arg \max F(\mathbf{w}) \implies \) generalised eigenvector condition. Interpret \( \mathbf{w} \).
Supervised Learning (Logistic Regression): Revision (cont’d).

- Formulation: for point $\mathbf{x}$ in feature space, find $P(\text{class} | \mathbf{x})$. 

  Procedure: empirically determine logarithms of ratios of probabilities for each training point $\mathbf{x}_n$ (log-odds).

  - Find linear regression: $f(\mathbf{x}) = \log\text{-odds}$. $\mathbf{x}$ can be of dimension more than 1.

  - Map $f(\mathbf{x})$ to $P(\text{class} | \mathbf{x})$ using logistic function.
Supervised Learning (Logistic Regression): Revision (cont’d).

- Formulation: for point \( \mathbf{x} \) in feature space, find \( P(\text{class}|\mathbf{x}) \).
  - Procedure: empirically determine logarithms of ratios of probabilities for each training point \( \mathbf{x}^n \) (log-odds)

\( f(\mathbf{x}) = \log\text{-odds} \).
\( \mathbf{x} \) can be of dimension more than 1.

\( \map{f(\mathbf{x})}{\text{log-odds}}{P(\text{class}|\mathbf{x})} \) using logistic function.
Supervised Learning (Logistic Regression): Revision (cont’d).

Formulation: for point \( x \) in feature space, find \( P(\text{class}|x) \).

- Procedure: empirically determine logarithms of ratios of probabilities for each training point \( x^n \) (log-odds)
- Find linear regression: \( f(x) = \text{log-odds} \). \( x \) can be of dimension more than 1.
Formulation: for point $\mathbf{x}$ in feature space, find $P(\text{class} | \mathbf{x})$.

- Procedure: empirically determine logarithms of ratios of probabilities for each training point $\mathbf{x}^n$ (log-odds)
- Find linear regression: $f(\mathbf{x}) = \text{log-odds}$. $\mathbf{x}$ can be of dimension more than 1.
- Map $f(\mathbf{x})$ to $P(\text{class} | \mathbf{x})$ using logistic function.
Data handling and unsupervised learning

- Principal Components Analysis (PCA)
- Blind source separation using Independent Components Analysis
- K-Means clustering
- Spectral clustering
Unsupervised dimensionality reduction: revision

- Data transformation

\[ \text{PCA problem formulation: find low dimensional representation} \]
\[ \mathbb{R}^p \ni x \rightarrow y \in \mathbb{R}^q, q < p, \text{so that} \] \[ \text{var}(f_y) \approx \text{var}(f_x). \]
\[ \text{How-to:} \]
\[ \rightarrow \text{e}_X, \text{for every feature subtract mean (retain deviations from mean)} \]
\[ \rightarrow \text{cov}(f_x) = e_X e_X^T \]
\[ \rightarrow \text{eigenvalue/eigenvector decomposition of cov}(f_x) \text{achieved by SVD}(e_X) \]
\[ e_X = U \Sigma V^T = e_X e_X^T \rightarrow e_X e_X^T = U \Sigma^2 U^T; \]
\[ \rightarrow \text{First p singular vectors gives optimal rank-p approximation} \]
\[ \|Y - X\|_2 \text{ smallest}. \]

Unlike LDA, PCA does not use labels. Can give poor projections for classification. Also, assumes Gaussian data.
Unsupervised dimensionality reduction: revision

- Data transformation

- PCA problem formulation: find low dimensional representation
  \[ \mathbb{R}^p \ni \mathbf{x}^n \mapsto \mathbf{y}^n \in \mathbb{R}^q, \ q < p, \ \text{so that} \ \text{var}\{\mathbf{y}^n\} \ \text{approximates} \ \text{var}\{\mathbf{x}^n\}. \]
Unsupervised dimensionality reduction: revision

- Data transformation

- PCA problem formulation: find low dimensional representation
  \[ \mathbb{R}^p \ni \mathbf{x}^n \mapsto \mathbf{y}^n \in \mathbb{R}^q, \ \text{q} < \text{p}, \ \text{so that } \text{var}(\{\mathbf{y}^n\}) \approx \text{var}(\{\mathbf{x}^n\}). \]

- How-to:
  - ▶ Subtract mean from each feature
  - ▶ Calculate covariance matrix: \[ \text{cov}(\{\mathbf{x}^n\}) = \mathbf{X} \mathbf{X}^T \]
  - ▶ Perform eigenvalue/eigenvector decomposition via SVD:
    \[ \mathbf{X} = \mathbf{U} \Sigma \mathbf{V}^T = \mathbf{U} \mathbf{U}^T \]
  - ▶ First \( p \) singular vectors provide optimal rank-\( p \) approximation
    \[ \| \mathbf{Y} - \mathbf{X} \|_2 \text{ is minimized.} \]
  - Unlike LDA, PCA does not use labels. Can give poor projections for classification. Also, assumes Gaussian data.
Unsupervised dimensionality reduction: revision

- Data transformation
- PCA problem formulation: find low dimensional representation
  \[ \mathbb{R}^p \ni x^n \mapsto y^n \in \mathbb{R}^q, \ q < p, \ \text{so that } \text{var}(\{y^n\}) \approx \text{var}(\{x^n\}). \]
- How-to:
  - \( \tilde{X} \), for every feature subtract mean (retain deviations from mean)
Unsupervised dimensionality reduction: revision

- Data transformation

- PCA problem formulation: find low dimensional representation
  \[ \mathbb{R}^p \ni \mathbf{x}^n \mapsto \mathbf{y}^n \in \mathbb{R}^q, \ q < p, \ \text{so that} \ \text{var}(\{\mathbf{y}^n\}) \ \text{approximates} \ \text{var}(\{\mathbf{x}^n\}). \]

- How-to:
  - \( \tilde{\mathbf{X}} \), for every feature subtract mean (retain deviations from mean)
  - \( \text{cov}(\{\mathbf{x}^n\}) = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T \)
Unsupervised dimensionality reduction: revision

- Data transformation

- PCA problem formulation: find low dimensional representation
  \( \mathbb{R}^p \ni \mathbf{x}^n \mapsto \mathbf{y}^n \in \mathbb{R}^q, \ q < p \), so that \( \text{var}(\{\mathbf{y}^n\}) \) approximates \( \text{var}(\{\mathbf{x}^n\}) \).

- How-to:
  - \( \tilde{\mathbf{X}} \), for every feature subtract mean (retain deviations from mean)
  - \( \text{cov}(\{\mathbf{x}^n\}) = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T \)
  - eigenvalue/eigenvector decomposition of \( \text{cov}(\{\mathbf{x}^n\}) \) achieved by SVD(\( \tilde{\mathbf{X}} \)):
    \[
    \tilde{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T \implies \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T = \mathbf{U}\Sigma^2\mathbf{U}^T.
    \]
Unsupervised dimensionality reduction: revision

- **Data transformation**

- **PCA problem formulation**: find low dimensional representation
  \[ \mathbb{R}^p \ni x^n \mapsto y^n \in \mathbb{R}^q, \ q < p, \ \text{so that } \text{var}(\{y^n\}) \ \text{approximates} \ \text{var}(\{x^n\}). \]

- **How-to**:
  - \( \tilde{X} \), for every feature subtract mean (retain deviations from mean)
  - \( \text{cov}(\{x^n\}) = \tilde{X}\tilde{X}^T \)
  - eigenvalue/eigenvector decomposition of \( \text{cov}(\{x^n\}) \) achieved by \( \text{SVD}(\tilde{X}) \):
    \[
    \tilde{X} = U\Sigma V^T \quad \Rightarrow \quad \tilde{X}\tilde{X}^T = U\Sigma^2U^T. 
    \]
  - First \( p \) singular vectors gives optimal rank-\( p \) approximation \( X \mapsto Y \):
    \[ \|Y - X\|^2 \ \text{smallest}. \]
Unsupervised dimensionality reduction: revision

- Data transformation

- PCA problem formulation: find low dimensional representation
  \( \mathbb{R}^p \ni x^n \mapsto y^n \in \mathbb{R}^q, \quad q < p, \) so that \( \text{var}(\{y^n\}) \) approximates \( \text{var}(\{x^n\}) \).

- How-to:
  - \( \tilde{X} \), for every feature subtract mean (retain deviations from mean)
  - \( \text{cov}(\{x^n\}) = \tilde{X}\tilde{X}^T \)
  - eigenvalue/eigenvector decomposition of \( \text{cov}(\{x^n\}) \) achieved by SVD(\( \tilde{X} \)):
    \[
    \tilde{X} = U\Sigma V^T \implies \tilde{X}\tilde{X}^T = U\Sigma^2U^T.
    \]
    - First \( p \) singular vectors gives optimal rank-\( p \) approximation \( X \mapsto Y \):
      \[ \| Y - X \|^2 \] smallest.

- Unlike LDA, PCA does not use labels. Can give poor projections for classification. Also, assumes Gaussian data.
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised
- k-means clustering: hard clusters
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised
- \(k\)-means clustering: hard clusters
  - Algorithm: initialise \(k\) representatives

Mixture modelling (mixtures of \(k\) Gaussians): soft clusters
  - Algorithm: initialise \(k\) means and \(k\) covariance matrices (parameters)
  - Assign each data point to all of \(k\) clusters: \(f x_n g \to (p_1, \ldots, p_k)
  - Recursive calculation of parameters (using data \(x_n\) weighted by \(p_n\), responsibilities) until convergence (EM algorithm)
  - For single Gaussian parameter estimation is by MLE

Spectral clustering
  - Algorithm: Find representation of data in terms of eigenvectors of graph Laplacian (weighted graph from data similarity followed by finding \(\text{argmax}_w w^T A B w^T A W w\))
  - \(k\)-means on data thus projected
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised
- \( k \)-means clustering: hard clusters
  - Algorithm: initialise \( k \) representatives
  - Assign each data point to one of \( k \) clusters: \( \{x^n\} \rightarrow (0, \ldots, 1, \ldots, 0) \)
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised
- $k$-means clustering: hard clusters
  - Algorithm: initialise $k$ representatives
  - Assign each data point to one of $k$ clusters: $\{x^n\} \rightarrow (0, \ldots, 1, \ldots, 0)$
  - Recursive calculation of $k$ means until convergence

- Mixture modelling (mixtures of $k$ Gaussians): soft clusters
  - Algorithm: initialise $k$ means and $k$ covariance matrices (parameters)
  - Assign each data point to all of $k$ clusters: $\{x^n\}! \left(p_1, \ldots, p_k\right)$
  - Recursive calculation of parameters (using data $x^n$ weighted by $p_n$, responsibilities) until convergence (EM algorithm)
  - For single Gaussian parameter estimation is by MLE

- Spectral clustering
  - Algorithm: Find representation of data in terms of eigenvectors of graph Laplacian (weighted graph from data similarity followed by finding $\text{argmax}_w w^T Aw$)
  - $k$-means on data thus projected
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised
- \(k\)-means clustering: hard clusters
  - Algorithm: initialise \(k\) representatives
  - Assign each data point to one of \(k\) clusters: \(\{\mathbf{x}^n\} \rightarrow (0, \ldots, 1, \ldots, 0)\)
  - Recursive calculation of \(k\) means until convergence
- Mixture modelling (mixtures of \(k\) Gaussians): soft clusters
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised
- $k$-means clustering: hard clusters
  - Algorithm: initialise $k$ representatives
  - Assign each data point to one of $k$ clusters: $\{x^n\} \rightarrow (0, \ldots, 1, \ldots, 0)$
  - Recursive calculation of $k$ means until convergence
- Mixture modelling (mixtures of $k$ Gaussians): soft clusters
  - Algorithm: initialise $k$ means and $k$ covariance matrices (parameters)
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised
- \( k \)-means clustering: hard clusters
  - Algorithm: initialise \( k \) representatives
  - Assign each data point to one of \( k \) clusters: \( \{x^n\} \rightarrow (0, \ldots, 1, \ldots, 0) \)
  - Recursive calculation of \( k \) means until convergence
- Mixture modelling (mixtures of \( k \) Gaussians): soft clusters
  - Algorithm: initialise \( k \) means and \( k \) covariance matrices (parameters)
  - Assign each data point to all of \( k \) clusters: \( \{x^n\} \rightarrow (p_1, \ldots, p_k) \)
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised

- \( k \)-means clustering: hard clusters
  - Algorithm: initialise \( k \) representatives
  - Assign each data point to **one** of \( k \) clusters: \( \{x^n\} \rightarrow (0, \ldots, 1, \ldots, 0) \)
  - Recursive calculation of \( k \) means until convergence

- Mixture modelling (mixtures of \( k \) Gaussians): soft clusters
  - Algorithm: initialise \( k \) means and \( k \) covariance matrices (parameters)
  - Assign each data point to **all** of \( k \) clusters: \( \{x^n\} \rightarrow (p_1, \ldots, p_k) \)
  - Recursive calculation of parameters (using data \( x^n \) weighted by \( p_i^n \), responsibilities) until convergence (EM algorithm)
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised
- \(k\)-means clustering: hard clusters
  - Algorithm: initialise \(k\) representatives
  - Assign each data point to \textbf{one} of \(k\) clusters: \(\{x^n\} \rightarrow (0, \ldots, 1, \ldots, 0)\)
  - Recursive calculation of \(k\) means until convergence
- Mixture modelling (mixtures of \(k\) Gaussians): soft clusters
  - Algorithm: initialise \(k\) means and \(k\) covariance matrices (parameters)
  - Assign each data point to \textbf{all} of \(k\) clusters: \(\{x^n\} \rightarrow (p_1, \ldots, p_k)\)
  - Recursive calculation of parameters (using data \(x^n\) weighted by \(p_i^n\), responsibilities) until convergence (EM algorithm)
  - For single Gaussian parameter estimation is by MLE
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised
- \(k\)-means clustering: hard clusters
  - Algorithm: initialise \(k\) representatives
  - Assign each data point to \textbf{one} of \(k\) clusters: \(\{x^n\} \rightarrow (0, \ldots, 1, \ldots, 0)\)
  - Recursive calculation of \(k\) means until convergence
- Mixture modelling (mixtures of \(k\) Gaussians): soft clusters
  - Algorithm: initialise \(k\) means and \(k\) covariance matrices (parameters)
  - Assign each data point to \textbf{all} of \(k\) clusters: \(\{x^n\} \rightarrow (p_1, \ldots, p_k)\)
  - Recursive calculation of parameters (using data \(x^n\) weighted by \(p_i^n\), responsibilities) until convergence (EM algorithm)
  - For single Gaussian parameter estimation is by MLE
- Spectral clustering
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised
- \( k \)-means clustering: hard clusters
  - Algorithm: initialise \( k \) representatives
  - Assign each data point to one of \( k \) clusters: \( \{\mathbf{x}^n\} \rightarrow (0, \ldots, 1, \ldots, 0) \)
  - Recursive calculation of \( k \) means until convergence
- Mixture modelling (mixtures of \( k \) Gaussians): soft clusters
  - Algorithm: initialise \( k \) means and \( k \) covariance matrices (parameters)
  - Assign each data point to all of \( k \) clusters: \( \{\mathbf{x}^n\} \rightarrow (p_1, \ldots, p_k) \)
  - Recursive calculation of parameters (using data \( \mathbf{x}^n \) weighted by \( p_i^n \), responsibilities) until convergence (EM algorithm)
  - For single Gaussian parameter estimation is by MLE
- Spectral clustering
  - Algorithm: Find representation of data in terms of eigenvectors of graph Laplacian (weighted graph from data similarity followed by finding \( \arg\max_w w^\top A_B w \frac{w^\top A w}{w^\top A w} \))
Unsupervised learning (clustering): revision

- Clustering formulation: within cluster variation minimised, between cluster variation maximised

  - \( k \)-means clustering: hard clusters
    - Algorithm: initialise \( k \) representatives
    - Assign each data point to one of \( k \) clusters: \( \{x^n\} \rightarrow (0, \ldots, 1, \ldots, 0) \)
    - Recursive calculation of \( k \) means until convergence

- Mixture modelling (mixtures of \( k \) Gaussians): soft clusters
  - Algorithm: initialise \( k \) means and \( k \) covariance matrices (parameters)
  - Assign each data point to all of \( k \) clusters: \( \{x^n\} \rightarrow (p_1, \ldots, p_k) \)
  - Recursive calculation of parameters (using data \( x^n \) weighted by \( p^n_i \), responsibilities) until convergence (EM algorithm)
  - For single Gaussian parameter estimation is by MLE

- Spectral clustering
  - Algorithm: Find representation of data in terms of eigenvectors of graph Laplacian (weighted graph from data similarity followed by finding
    \[ \text{argmax}_w \frac{w^\top A_B w}{w^\top A_w w} \])
  - \( k \)-means on data thus projected
Regression and Model-fitting Techniques

- Linear regression (including regularisation)
- Polynomial Fitting
- Kernel Based Networks (RBF)
Regression and Model-fitting Techniques: revision

- Data \( \{x^n, y^n\} \), \( x^n \in \mathbb{R}^p \) \( p \)-dimensional data, \( y \in \mathbb{R} \).
Regression and Model-fitting Techniques: revision

- Data \( \{x^n, y^n\}, x^n \in \mathbb{R}^p \) \( p \)-dimensional data, \( y \in \mathbb{R} \).
- Problem formulation:
Regression and Model-fitting Techniques: revision

- Data \( \{\mathbf{x}^n, y^n\}, \mathbf{x}^n \in \mathbb{R}^p \) p-dimensional data, \( y \in \mathbb{R} \).
- Problem formulation:
  - Linear regression (including regularisation) \( y^n = w_0 + w_1 x_1 + \cdots + w_p x_p \),
    - Find \( w = (w_0; w_1; \ldots; w_p) \) to minimise \( \sum_n (y^n - w \mathbf{x}^n)^2 + \lambda \sum_i (w_i)^2 + \lambda \sum_{ij} (w_i w_j) \), where \( \mathbf{x}^n = (1; \phi_1(x^n); \ldots; \phi_q(x^n)) \) and \( \phi_i(x^n) \) is a function of your choice mapping each \( p \)-dimensional data point to a real number (e.g., \( \phi((x_1; x_2)) = x_1 x_2 \)).
  - Solution of optimisation by calculus:
    - Obtained from pseudo-inverse of design matrix or by gradient descent.
Regression and Model-fitting Techniques: revision

- Data \( \{x^n, y^n\}, x^n \in \mathbb{R}^p \) \( p \)-dimensional data, \( y \in \mathbb{R} \).
- Problem formulation:
  - Linear regression (including regularisation) \( y^n = w_0 + w_1 x_1 + \cdots + w_p x_p \),
    - Find \( w = (w_0, w_1, \ldots, w_p) \) to minimise
      \[
      \sum_n (y^n - w \cdot x^n)^2 + \lambda_2 \sum_{i=0}^{p} w_i^2 + \lambda_1 \sum_{i=0}^{p} |w_i|
      \]
Regression and Model-fitting Techniques: revision

- Data \( \{ \mathbf{x}^n, y^n \} \), \( \mathbf{x}^n \in \mathbb{R}^p \) \( p \)-dimensional data, \( y \in \mathbb{R} \).
- Problem formulation:
  - Linear regression (including regularisation) \( y^n = w_0 + w_1 x_1 + \cdots + w_p x_p \),
  - Find \( \mathbf{w} = (w_0, w_1, \ldots, w_p) \) to minimise
    \[
    \sum_n (y^n - \mathbf{w} \cdot \mathbf{x}^n)^2 + \lambda_2 \sum_{i=0}^{p} w_i^2 + \lambda_1 \sum_{i=0}^{p} |w_i| \]
  - Fitting Polynomials and RBFs
Regression and Model-fitting Techniques: revision

- Data \( \{x^n, y^n\}, x^n \in \mathbb{R}^p \) \( p \)-dimensional data, \( y \in \mathbb{R} \).

- Problem formulation:
  - Linear regression (including regularisation) \( y^n = w_0 + w_1 x_1 + \cdots + w_p x_p \),
    - Find \( w = (w_0, w_1, \ldots, w_p) \) to minimise
      \[
      \sum_n (y^n - w \cdot x^n)^2 + \lambda_2 \sum_{i=0}^{p} w_i^2 + \lambda_1 \sum_{i=0}^{p} |w_i|
      \]
  - Fitting Polynomials and RBFs
    - Find \( w = (w_0, w_1, \ldots, w_q) \) to minimise
      \[
      \sum_n (y^n - w \cdot \Phi(x^n))^2 + \lambda_2 \sum_{i=0}^{q} w_i^2 + \lambda_1 \sum_{i=0}^{q} |w_i|,
      \]
      where \( \Phi(x^n) = (1, \phi_1(x^n), \ldots, \phi_q(x^n)) \) and \( \phi_i(x^n) \) is a function of your choice mapping each \( p \)-dimensional data point to a real number. (Eg., \( \phi((x_1, x_2)) = x_1 x_2 \).)
Regression and Model-fitting Techniques: revision

- Data \( \{x^n, y^n\} \), \( x^n \in \mathbb{R}^p \) \( p \)-dimensional data, \( y \in \mathbb{R} \).

- Problem formulation:
  - Linear regression (including regularisation) \( y^n = w_0 + w_1 x_1 + \cdots + w_p x_p \),
    - Find \( w = (w_0, w_1, \ldots, w_p) \) to minimise
      \[
      \sum_n (y^n - w \cdot x^n)^2 + \lambda_2 \sum_{i=0}^p w_i^2 + \lambda_1 \sum_{i=0}^p |w_i|
      \]
  - Fitting Polynomials and RBFs
    - Find \( w = (w_0, w_1, \ldots, w_q) \) to minimise
      \[
      \sum_n (y^n - w \cdot \Phi(x^n))^2 + \lambda_2 \sum_{i=0}^q w_i^2 + \lambda_1 \sum_{i=0}^q |w_i|,
      \]
      where \( \Phi(x^n) = (1, \phi_1(x^n), \ldots, \phi_q(x^n)) \) and \( \phi_i(x^n) \) is a function of your choice mapping each \( p \)-dimensional data point to a real number. (Eg., \( \phi((x_1, x_2)) = x_1 x_2 \).)

- Solution of optimisation by calculus:
Regression and Model-fitting Techniques: revision

- Data \(\{\mathbf{x}^n, y^n\}, \mathbf{x}^n \in \mathbb{R}^p\) \(p\)-dimensional data, \(y \in \mathbb{R}\).

- Problem formulation:
  - Linear regression (including regularisation) \(y^n = w_0 + w_1 x_1 + \cdots + w_p x_p\),
    - Find \(\mathbf{w} = (w_0, w_1, \ldots, w_p)\) to minimise
      \[
      \sum_{n} (y^n - \mathbf{w} \cdot \mathbf{x}^n)^2 + \lambda_2 \sum_{i=0}^{p} w_i^2 + \lambda_1 \sum_{i=0}^{p} |w_i|
      \]
  - Fitting Polynomials and RBFs
    - Find \(\mathbf{w} = (w_0, w_1, \ldots, w_q)\) to minimise
      \[
      \sum_{n} (y^n - \mathbf{w} \cdot \Phi(\mathbf{x}^n))^2 + \lambda_2 \sum_{i=0}^{q} w_i^2 + \lambda_1 \sum_{i=0}^{q} |w_i|,
      \]
    where \(\Phi(\mathbf{x}^n) = (1, \phi_1(\mathbf{x}^n), \ldots, \phi_q(\mathbf{x}^n))\) and \(\phi_i(\mathbf{x}^n)\) is a function of your choice mapping each \(p\)-dimensional data point to a real number. (Eg., \(\phi((x_1, x_2)) = x_1 x_2\).)

- Solution of optimisation by calculus:
  - \(\mathbf{w}\) obtained from pseudo-inverse of design matrix \(A\) or by gradient descent
Regularisation: revision

For data \( \{x^n, y^n\}, x^n \in \mathbb{R}^p, y \in \mathbb{R} \), regularised optimisation problem: Find \( w = (w_0, w_1, \ldots, w_q) \) to minimise

\[
\sum_n (y^n - w \cdot \Phi(x^n))^2 + \lambda_2 \sum_{i=0}^q w_i^2,
\]

where \( \Phi(x^n) = (1, \phi_1(x^n), \ldots, \phi_q(x^n)) \).

- **Solution how-to:**
  - Stack \( \Phi(x^n) \) column-wise to form design matrix \( A \)
  - Consider \( w \) obtained from pseudo-inverse of \( A \), and \( y = A \hat{w} \).

- **SVD of design matrix exposes need for regularisation:**
  - \( A = U \Sigma V^T = \sum_k u_k \sigma_k v_k^T \).
Regularisation: revision

For data \( \{x^n, y^n\}, x^n \in \mathbb{R}^p, y \in \mathbb{R} \), regularised optimisation problem: Find \( w = (w_0, w_1, \ldots, w_q) \) to minimise

\[
\sum_n (y^n - w \cdot \Phi(x^n))^2 + \lambda_2 \sum_{i=0}^q w_i^2,
\]

where \( \Phi(x^n) = (1, \phi_1(x^n), \ldots, \phi_q(x^n)) \).

- **Solution how-to:**
  - Stack \( \Phi(x^n) \) column-wise to form design matrix \( A \)
  - Consider \( w \) obtained from pseudo-inverse of \( A \), and \( y = A \hat{w} \).
- **SVD of design matrix exposes need for regularisation:**
  - \( A = U \Sigma V^T = \sum_k u_k \sigma_k v_k^T \)
  - \( w = (A^T A)^{-1} A^T y = \sum_k \frac{u_k^T y}{\sigma_k} v_k \).
Regularisation: revision

For data \( \{\mathbf{x}^n, y^n\} \), \( \mathbf{x}^n \in \mathbb{R}^p \), \( y \in \mathbb{R} \), regularised optimisation problem: Find \( \mathbf{w} = (w_0, w_1, \ldots, w_q) \) to minimise

\[
\sum_n (y^n - \mathbf{w} \cdot \Phi(\mathbf{x}^n))^2 + \lambda_2 \sum_{i=0}^q w_i^2,
\]

where \( \Phi(\mathbf{x}^n) = (1, \phi_1(\mathbf{x}^n), \ldots, \phi_q(\mathbf{x}^n)) \).

- **Solution how-to:**
  - Stack \( \Phi(\mathbf{x}^n) \) column-wise to form design matrix \( \mathbf{A} \)
  - Consider \( \mathbf{w} \) obtained from pseudo-inverse of \( \mathbf{A} \), and \( \mathbf{y} = \mathbf{A} \hat{\mathbf{w}} \).

- **SVD of design matrix exposes need for regularisation:**
  - \( \mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T = \sum_k \mathbf{u}_k \sigma_k \mathbf{v}_k^T \).
  - \( \mathbf{w} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y} = \sum_k \frac{\mathbf{u}_k^T \mathbf{y}}{\sigma_k} \mathbf{v}_k. \)
  - Small \( \sigma_k \) large component of \( \mathbf{w} \).
Regularisation: revision

For data \( \{x^n, y^n\}, x^n \in \mathbb{R}^p, y \in \mathbb{R} \), regularised optimisation problem: Find \( w = (w_0, w_1, \ldots, w_q) \) to minimise

\[
\sum_{n} (y^n - w \cdot \Phi(x^n))^2 + \lambda_2 \sum_{i=0}^{q} w_i^2,
\]

where \( \Phi(x^n) = (1, \phi_1(x^n), \ldots, \phi_q(x^n)) \).

- Solution how-to:
  - Stack \( \Phi(x^n) \) column-wise to form design matrix \( A \)
  - Consider \( w \) obtained from pseudo-inverse of \( A \), and \( y = A \hat{w} \).

- SVD of design matrix exposes need for regularisation:
  - \( A = U \Sigma V^T = \sum_k u_k \sigma_k v_k^T \).
  - \( w = (A^T A)^{-1} A^T y = \sum_k \frac{u_k^T y}{\sigma_k} v_k \).
  - Small \( \sigma_k \) large component of \( w \)
  - Regularised optimisation yields \( \hat{w} = (A^T A + \lambda_2 \mathbb{I})^{-1} A^T y \),

\[
\hat{w} = \sum_k \frac{\sigma_k}{\sigma_k^2 + \lambda_2} (u_k^T y) v_k \implies y = \sum_k \frac{\sigma_k^2}{\sigma_k^2 + \lambda_2} u_k u_k^T
\]
Linear Algebra and Optimisation

- Linear Algebra
  - Using matrices to find solutions of linear equations
  - Properties of matrices and vector spaces
  - Eigenvalues, eigenvectors and singular value decomposition

- Optimisation
  - Convexity
  - 1-D minimisation
  - Gradient methods in higher dimensions (co-ordinate descent)
  - Constrained optimisation
Linear Algebra: revision

- Linear Algebra

- Using matrices to find solutions of linear equations:
  \[ w = (A^T A)^{-1} A^T y, \]

- Properties of matrices and vector spaces:
  \[ x = \sum_i i u_i \]
  for a basis.

- Eigenvalues, eigenvectors and singular value decomposition:
  \[ A = U V^T \]
  \[ A^T A = V \Sigma^2 V^T \]
  \[ \Lambda = U \Sigma \]
  \[ (A^T A) v_k = \lambda_k v_k. \]
Linear Algebra

- Using matrices to find solutions of linear equations: \( \mathbf{w} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y} \), residual orthogonal to optimal weight vector
Linear Algebra

- Linear Algebra
  - Using matrices to find solutions of linear equations: \( \mathbf{w} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y} \), residual orthogonal to optimal weight vector
  - Properties of matrices and vector spaces: \( \mathbf{x} = \sum_i \alpha_i \mathbf{u}_i \) for \( \{\mathbf{u}_i\} \) basis.
Linear Algebra: revision

- Linear Algebra
  - Using matrices to find solutions of linear equations: \( \mathbf{w} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y} \), residual orthogonal to optimal weight vector
  - Properties of matrices and vector spaces: \( \mathbf{x} = \sum_i \alpha_i \mathbf{u}_i \) for \( \{\mathbf{u}_i\} \) basis.
  - Eigenvalues, eigenvectors and singular value decomposition:
Linear Algebra: revision

- Linear Algebra
  - Using matrices to find solutions of linear equations: \( \mathbf{w} = (A^T A)^{-1} A^T \mathbf{y} \), residual orthogonal to optimal weight vector
  - Properties of matrices and vector spaces: \( \mathbf{x} = \sum_i \alpha_i \mathbf{u}_i \) for \( \{\mathbf{u}_i\} \) basis.
  - Eigenvalues, eigenvectors and singular value decomposition:
    - \( A = U\Sigma V^T \)
Linear Algebra

- Using matrices to find solutions of linear equations: \( \mathbf{w} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y} \), residual orthogonal to optimal weight vector
- Properties of matrices and vector spaces: \( \mathbf{x} = \sum_i \alpha_i \mathbf{u}_i \) for \( \{\mathbf{u}_i\} \) basis.
- Eigenvalues, eigenvectors and singular value decomposition:
  - \( \mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T \)
  - \( \mathbf{A}^T \mathbf{A} = \mathbf{V} \Sigma^2 \mathbf{V}^T \) or \( (\mathbf{A}^T \mathbf{A}) \mathbf{v}_k = \sigma_k^2 \mathbf{v}_k \).
Linear Algebra

- Using matrices to find solutions of linear equations: \( \mathbf{w} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y} \), residual orthogonal to optimal weight vector
- Properties of matrices and vector spaces: \( \mathbf{x} = \sum_i \alpha_i \mathbf{u}_i \) for \( \{\mathbf{u}_i\} \) basis.
- Eigenvalues, eigenvectors and singular value decomposition:
  - \( \mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T \)
  - \( \mathbf{A}^T \mathbf{A} = \mathbf{V} \mathbf{\Sigma}^2 \mathbf{V}^T \) or \( (\mathbf{A}^T \mathbf{A}) \mathbf{v}_k = \sigma_k^2 \mathbf{v}_k \).
  - \( \mathbf{A} \mathbf{A}^T = \mathbf{U} \mathbf{\Sigma}^2 \mathbf{U}^T \) or \( (\mathbf{A} \mathbf{A}^T) \mathbf{u}_k = \sigma_k^2 \mathbf{u}_k \).
Convexity: definition of convex function in the context of KL divergence

I-D minimisation: calculus

Gradient methods in higher dimensions (co-ordinate descent):
\[ C((x_1^*, \ldots, x_p^*) + \epsilon x_i) = C((x_1^*, \ldots, x_p^*)) + \epsilon \frac{\partial}{\partial x_i} C((x_1^*, \ldots, x_p^*)). \]

Constrained optimisation: regularised regression, maximum likelihood estimation: eg. set to 0 partial derivatives with respect to \( \theta \) and \( \lambda \):
\[
\sum_n \log(p(x^n; \theta)) + \lambda \left( \sum_n p(x^n; \theta) - 1 \right)
\]