Intro to ML

Evgeny Burnaev

Skoltech, Moscow, Russia

Skoltech
Skolkovo Institute of Science and Technology

- **Training data**: sample drawn i.i.d. from set X according to some distribution D

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \in X \times Y,$$

with $Y \subseteq \mathbb{R}$ is a measurable set, $X \subseteq \mathbb{R}^d$, $\mathbf{x}_i \in \mathbb{R}^{1 \times d}$

- **Loss function**: $L : Y \times Y \to \mathbb{R}_+$ a measure of closeness, e.g. $L(y, y') = (y - y')^2$ or $L(y, y') = |y - y'|^p$ for some $p \geq 1$

- **Problem**: find hypothesis $\widehat{f} : X \to \mathbb{R}$ in $\mathbb{H}$ with small generalization error w.r.t. target f

$$R_D(\widehat{f}) = \mathbb{E}_{\mathbf{x} \sim D}[L(\widehat{f}(\mathbf{x}), f(\mathbf{x}))]$$

- Empirical error:

$$\widehat{R}_D(h) = \frac{1}{m} \sum_{i=1}^{m} L(\widehat{f}(\mathbf{x}_i), y_i)$$

- Optimization problem statement
  $F(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \left(\mathbf{w} \cdot \mathbf{x}_i^\top + b - y_i\right)^2 \to \min_{\mathbf{w},b}$
- Rewrite objective function as $F(\mathbf{W}) = \frac{1}{m} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|^2$, where

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & 1 \\ \vdots & \vdots \\ \mathbf{x}_m & 1 \end{bmatrix} \in \mathbb{R}^{m \times (d+1)}, \ \mathbf{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \\ b \end{bmatrix}, \ \mathbf{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

- **Solution**:

$$\mathbf{W} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} \quad \text{if} \quad \mathbf{X}^\top \mathbf{X} \quad \text{invertible}$$

- **Ridge Regression**:

$$F(\mathbf{w}, b) = \sum_{i=1}^{m} (\mathbf{w} \cdot \mathbf{x}_i^\top + b - y_i)^2 + \lambda \|\mathbf{w}\|^2 \to \min_{\mathbf{w},b}$$

- **Solution**:

$$\mathbf{W} = \underbrace{(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}}_{\text{always invertible!}} \mathbf{X}^\top \mathbf{Y}$$

Direct Dual Solution

- We can easily prove that

$$(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top = \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}$$

- **Dual solution**: thus we get that

$$\mathbf{W} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{Y} = \mathbf{X}^\top \underbrace{(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}\mathbf{Y}}_{\text{new variable } \boldsymbol{\alpha}},$$

- With

$$\boldsymbol{\alpha} = (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}\mathbf{Y},$$

  we can represent $\mathbf{W}$ as

$$\mathbf{W} = \mathbf{X}^\top\boldsymbol{\alpha} = \sum_{i=1}^{m}\alpha_i\mathbf{x}_i^\top,$$

- We can use dual representation of the solution

$$\widehat{f}(\mathbf{x}) = \mathbf{x}\cdot\mathbf{W} = \sum_{i=1}^{m}\alpha_i(\mathbf{x}\cdot\mathbf{x}_i^\top)$$

PDS condition

- **Definition**: a kernel $K : X \times X \to \mathbb{R}$ is *positive definite symmetric* (PDS) is for any $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq X$ the matrix $\mathrm{K} = [K(\mathbf{x}_i, \mathbf{x}_j)]_{ij} \in \mathbb{R}^{m \times m}$ is symmetric positive semi-definite (SPSD)

- Matrix K SPSD if symmetric and one of the 2 equiv. cond.'s:
  - its eigenvalues are non-negative
  - for any $\mathbf{c} \in \mathbb{R}^{m \times 1}$, $\mathbf{c}^\top \mathrm{K} \mathbf{c} = \sum_{i,j=1}^{m} c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0$

- **Terminology**: PDS for kernels, SPDS for kernel matrices

- **Definition**: the *normalized kernel* $\widetilde{K}$ associated to a kernel K is defined by

$$\forall \mathbf{x}, \mathbf{x}' \in X,\ \widetilde{K}(\mathbf{x}, \mathbf{x}') = \begin{cases} 0, & \text{if } \ K(\mathbf{x}, \mathbf{x}) = 0 \ \text{ or } \ K(\mathbf{x}', \mathbf{x}') = 0 \\ \frac{K(\mathbf{x}, \mathbf{x}')}{\sqrt{K(\mathbf{x}, \mathbf{x}) K(\mathbf{x}', \mathbf{x}')}} \end{cases}$$

- **Gaussian kernels**:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right),\ \sigma \neq 0$$

Gaussian kernel is a normalized kernel of

$$(\mathbf{x}, \mathbf{x}') \to \exp\left(\frac{\mathbf{x} \cdot \mathbf{x}'}{\sigma^2}\right) = \sum_{n=0}^{\infty} \frac{(\mathbf{x} \cdot \mathbf{x}')^n}{\sigma^n n!}$$

Closure Properties of PDS kernels

- **Theorem**: Positive definite symmetric (PDS) kernels are closed under:
  - sum
  - product
  - tensor product
  - pointwise limit
  - composition with a power series
  - the following transformation with any function $f(\mathbf{x})$

$$f(\mathbf{x})K(\mathbf{x},\mathbf{x}')f(\mathbf{x}')$$

- Usual linear ridge regression in dual representation

$$\widehat{f}(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i (\mathbf{x} \cdot \mathbf{x}_i^\top)$$

with

$$\boldsymbol{\alpha} = (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I})^{-1} \mathbf{Y}$$

- Kernel ridge regression

$$\widehat{f}(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i (\boldsymbol{\Phi}(\mathbf{x}) \cdot \boldsymbol{\Phi}(\mathbf{x}_i)^\top) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}_i, \mathbf{x})$$

with

$$\boldsymbol{\alpha} = (\boldsymbol{\Phi}(\mathbf{X}) \cdot \boldsymbol{\Phi}(\mathbf{X})^\top + \lambda \mathbf{I})^{-1} \mathbf{Y} = (\mathrm{K} + \lambda \mathbf{I})^{-1} \mathbf{Y},$$

where

$$\mathrm{K} = \{\boldsymbol{\Phi}(\mathbf{x}_i) \cdot \boldsymbol{\Phi}(\mathbf{x}_j)^\top\}_{i,j=1}^{m} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^{m}$$

- **Ordinary Least Squares** (OLS):

$$F(\mathbf{w}, b) = \sum_{i=1}^{m} (\mathbf{w} \cdot \mathbf{x}_i^\top + b - y_i)^2 \to \min_{\mathbf{w}, b},$$

- **Ridge Regression**:

$$F(\mathbf{w}, b) = \lambda \|\mathbf{w}\|_2^2 + \sum_{i=1}^{m} (\mathbf{w} \cdot \mathbf{x}_i^\top + b - y_i)^2 t \to \min_{\mathbf{w}, b},$$

where $\lambda \geq 0$ is a regularization parameter

- **LASSO**:

$$F(\mathbf{w}, b) = \lambda \|\mathbf{w}\|_1 + \sum_{i=1}^{m} (\mathbf{w} \cdot \mathbf{x}_i^\top + b - y_i)^2 \to \min_{\mathbf{w}, b},$$

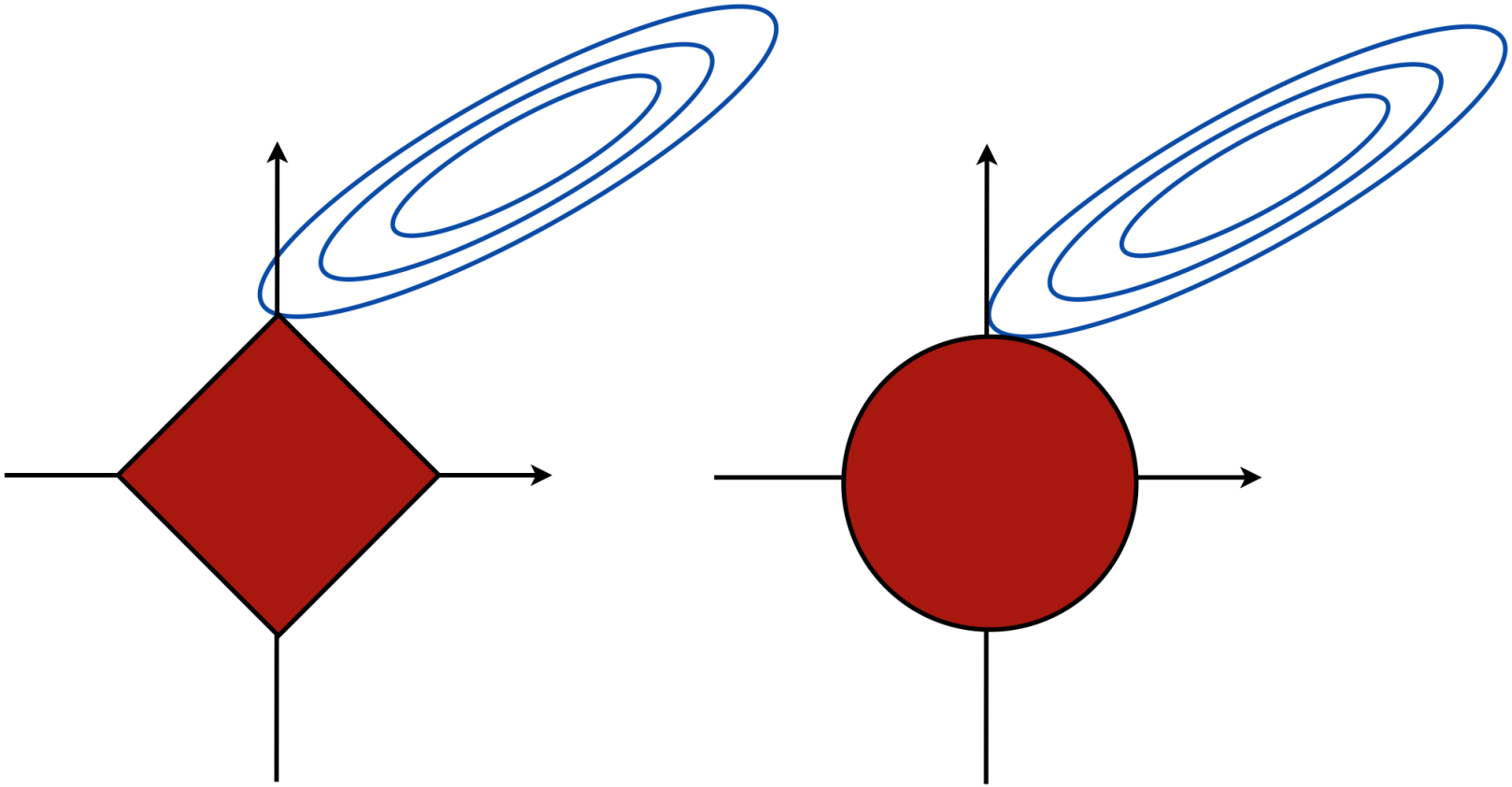
where $\lambda \geq 0$ is a regularization parameter

- **Elastic Net**:

$$F(\mathbf{w}, b) = \lambda \left((1 - \alpha) \|\mathbf{w}\|_1 + \alpha \|\mathbf{w}\|_2^2 \right) + \sum_{i=1}^{m} (\mathbf{w} \cdot \mathbf{x}_i^\top + b - y_i)^2 \to \min_{\mathbf{w}, b},$$

where $\lambda \geq 0$ is a regularization parameter, $\alpha \in [0, 1]$

Sparsity of L_1 regularization



L1 regularization

L2 regularization

Binary Classification Errors

- Binary classification, $y_i \in \{-1, +1\}$
- Classifier $f(\mathbf{x}_i) \in \{-1, +1\}$

Error type	Classifier	True label
TP, True Positive	$f(\mathbf{x}_i) = +1$	$y_i = +1$
TN, True Negative	$f(\mathbf{x}_i) = -1$	$y_i = -1$
FP, False Positive	$f(\mathbf{x}_i) = +1$	$y_i = -1$
FN, False Negative	$f(\mathbf{x}_i) = -1$	$y_i = +1$

- Proportion of classification errors

$$\text{Accuracy} = \frac{1}{m}\sum_{i=1}^{m} 1_{\{f(\mathbf{x}_i) \neq y_i\}} = \frac{FP + FN}{FP + FN + TP + TN}$$

- **Shortcomings**: Accuracy does not take into account neither disbalance of classes nor different error costs for different classes

- Let us define ROC (receiver operating characteristic) curve
- Each point of the curve corresponds to some

$$f(\mathbf{x}; \mathbf{w}, w_0) = \text{sign}(h(\mathbf{x}; \mathbf{w}) - w_0)$$

- **Abscissa**: we depict values of FPR (false positive rate) as a function of w_0

$$FPR(w_0) = \frac{\sum_{i=1}^{m} 1_{\{y_i=-1\}} 1_{\{f(\mathbf{x}_i;\mathbf{w},w_0)=+1\}}}{\sum_{i=1}^{m} 1_{\{y_i=-1\}}}$$

- **Ordinate**: we depict values of TPR (true positive rate):

$$TPR(w_0) = \frac{\sum_{i=1}^{m} 1_{\{y_i=+1\}} 1_{\{f(\mathbf{x}_i;\mathbf{w},w_0)=+1\}}}{\sum_{i=1}^{m} 1_{\{y_i=+1\}}}$$

- $(1 - FPR)$ is called specificity, TPR is called sensitivity

Precision and Recall in case of Binary Classification

- TP — true positives
- FP — false positives
- FN — false negatives
- Precision and Recall

$$\text{Precision} : P = \frac{TP}{TP + FP},$$

$$\text{Recall} : R = \frac{TP}{TP + FN}$$

In information retrieval

- P is a fraction of relevant objects among retrieved
- R is a fraction of retrieved objects among relevant

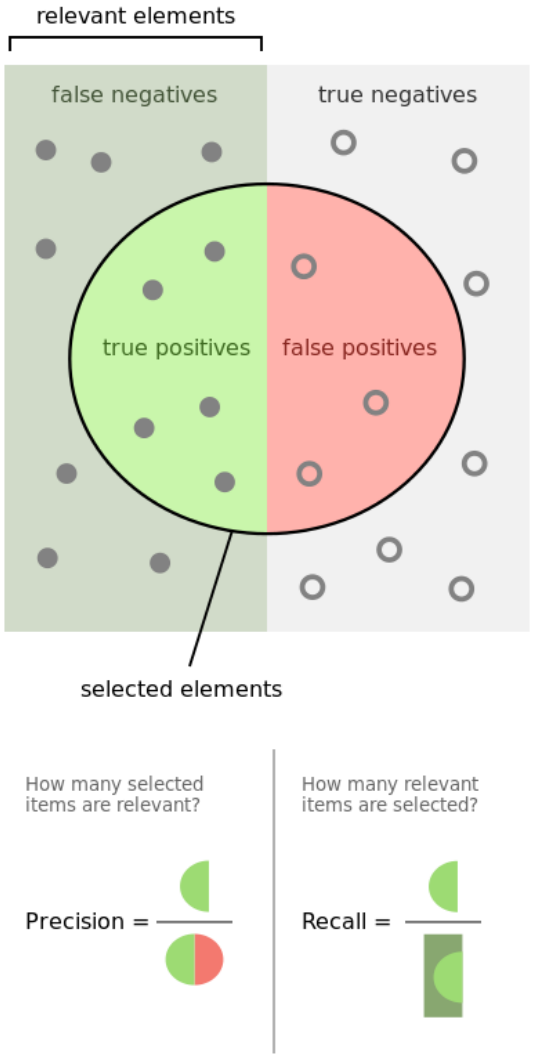


Figure – Graphical illustration [Wikipedia]

Accuracy Measures: comments

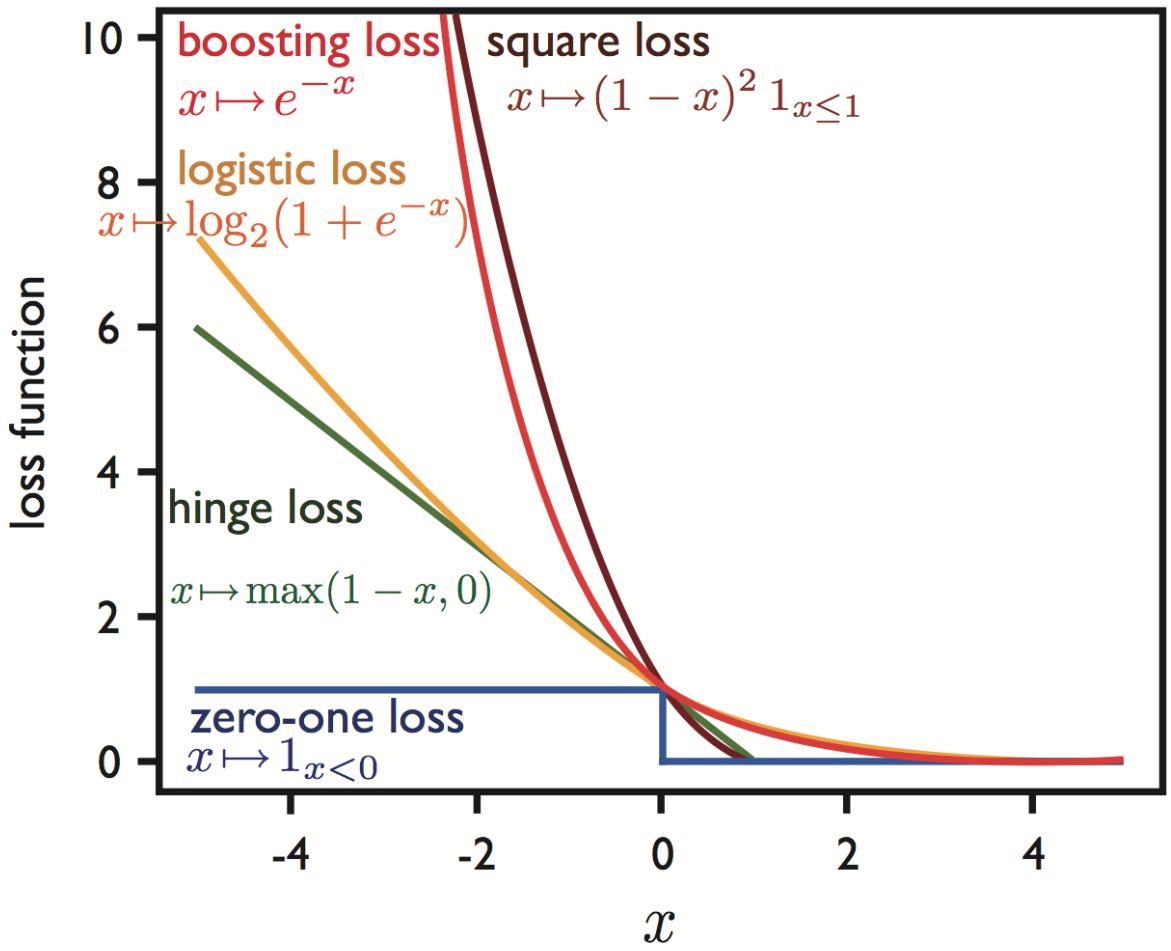
- Sensitivity and specificity are better suited for problems with unbalanced classes
- Precision and Recall are better suited for problems when a class of interest is rare
- Aggregated errors:
  - AUC is better suited when ratio of errors costs for different classes is not fixed
  - AUC-PR — area under precision-recall curve
  - F-measure $F_1 = \frac{2PR}{P+R}$
  - F_β-measure $F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P+R}$ (the bigger β is the more R is important)

Convex upper bound for an indicator function

$$1_{\{x<0\}} \leq L(x),$$

i.e.

$$1_{\{g(\mathbf{x}_i;\mathbf{w})y_i<0\}} \leq L(g(\mathbf{x}_i;\mathbf{w})y_i),$$



Surrogate Loss ERM

- Learning sample $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$
- Classifier

$$f(\mathbf{x}; \mathbf{w}) = \text{sign}(g(\mathbf{x}; \mathbf{w}))$$

- Example of Linear classifier: $g(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$ (we assume x_1 to account for a constant $-w_0$)
- Binary Loss function and its convex upper bound

$$1_{\{g(\mathbf{x}_i;\mathbf{w})y_i<0\}} \leq L(g(\mathbf{x}_i; \mathbf{w})y_i)$$

- Learning $\equiv$ ERM

$$\widehat{R}(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^m 1_{\{g(\mathbf{x}_i;\mathbf{w})y_i<0\}} \leq \frac{1}{m}\sum_{i=1}^m L(g(\mathbf{x}_i; \mathbf{w})y_i) \to \min_{\mathbf{w}}$$

- Estimate a test error using a separate test sample $S'_{m'} = \{(\mathbf{x}'_j, y'_j)\}_{j=1}^{m'}$

$$\frac{1}{m'}\sum_{j=1}^{m'} 1_{\{g(\mathbf{x}'_j;\mathbf{w})y'_j<0\}}$$

Log-loss

- Probabilistic classifier model: a parametric model for

$$\mathbb{P}(y|\mathbf{x}) = p(y|\mathbf{x};\mathbf{w})$$

- MLE for $\mathbf{w}$

$$\prod_{i=1}^{m} p(y_i|\mathbf{x}_i;\mathbf{w}) \to \max_{\mathbf{w}}$$

- Log-likelihood

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{m} \log p(y_i|\mathbf{x}_i;\mathbf{w}) \to \max_{\mathbf{w}}$$

- If we set

$$p(y|\mathbf{x};\mathbf{w}) = e^{-L(g(\mathbf{x};\mathbf{w})y)},$$

then we get the surrogate loss ERM problem

$$\sum_{i=1}^{m} L(g(\mathbf{x}_i;\mathbf{w})y_i) \to \min_{\mathbf{w}},$$

i.e. the surrogate loss function $L(\cdot)$ and $g(\mathbf{x};\mathbf{w})$ define the probabilistic classifier model

- Linear Classifier in case of $Y = \{-1, +1\}$

$$f(\mathbf{x}; \mathbf{w}) = \text{sign}(g(\mathbf{x}; \mathbf{w})) = \text{sign}(\mathbf{w}^\top \mathbf{x})$$

- Logarithmic loss function

$$L(t) = \log(1 + e^{-t})$$

- A model for conditional probability

$$p(y|\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}$$

- Regularized logistic regression

$$\overline{L}(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \log(1 + \exp(-(\mathbf{w}^\top \mathbf{x}_i) y_i)) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \to \min_{\mathbf{w}}$$

Naive Bayes Classifier

- Naive Bayes (NB) is a conditional probability model
- Given an object to be classified, represented by features $\mathbf{x} = (x_1, \ldots, x_d)$, Bayes classifier assigns probabilities

$$p(y|x_1, \ldots, x_d)$$

for each of K possible classes $y \in Y = \{1, \ldots, K\}$

- The conditional probability can be decomposed as

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})} \sim p(y)p(\mathbf{x}|y),\ y \in Y$$

- The maximum a posteriori (MAP) decision rule

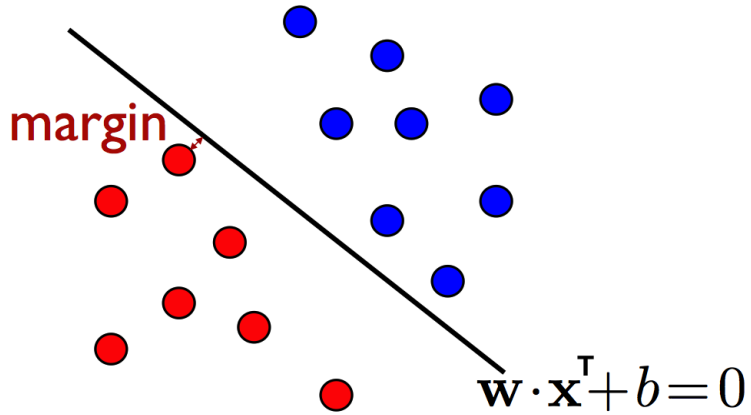
$$\widehat{y} = \arg \max_{y \in \{1,\ldots,K\}} p(y)p(\mathbf{x}|y)$$

- We model $p(\mathbf{x}|y)$ as $\prod_{j=1}^{d} p(x_j|y)$, thus

$$p(y|\mathbf{x}) \sim p(y) \prod_{j=1}^{d} p(x_j|y),\ y \in Y,$$

and use individual parametric models for $p(x_j|y)$ with paramerters, estimated from data

SVM: Linear separable case



- classifiers: $H = \{\mathbf{x} \to \text{sgn}(\mathbf{w} \cdot \mathbf{x}^\top + b), \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$
- Distance from i-th point to the hyperplane

$$\gamma^{(i)} = \left| \frac{\mathbf{w}}{\|\mathbf{w}\|} \mathbf{x}_i^\top + \frac{b}{\|\mathbf{w}\|} \right| = \frac{|\mathbf{w} \cdot \mathbf{x}_i^\top + b|}{\|\mathbf{w}\|} \to \min_{i \in [1,m]} \text{(worst case!)}$$

- The margin is

$$\rho = \max_{\mathbf{w},b:\, y_i(\mathbf{w}\cdot\mathbf{x}_i^\top + b) \geq 0} \left[\min_{i \in [1,m]} \frac{|\mathbf{w} \cdot \mathbf{x}_i^\top + b|}{\|\mathbf{w}\|} \right] = \max_{\mathbf{w},b:\, y_i(\mathbf{w}\cdot\mathbf{x}_i^\top + b) \geq 1} \left[\frac{1}{\|\mathbf{w}\|} \right]$$

- Optimization problem

$$\max_{\mathbf{w},b:\, y_i(\mathbf{w}\cdot\mathbf{x}_i^\top+b)\geq 1} \left[\frac{1}{\|\mathbf{w}\|}\right]$$

- **Constrained Optimization**:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t. } y_i(\mathbf{w}\cdot\mathbf{x}_i^\top + b) \geq 1,\ i \in [1,m]$$

- **Lagrangian**: for all $\mathbf{w}, b, \alpha_i \geq 0$

$$L(\mathbf{w},b,\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i[y_i(\mathbf{w}\cdot\mathbf{x}_i^\top + b) - 1]$$

- **Constrained Optimization**:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j^\top)$$

$$\text{s.t. } \alpha_i \geq 0 \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0,\ i \in [1, m]$$

- Optimal

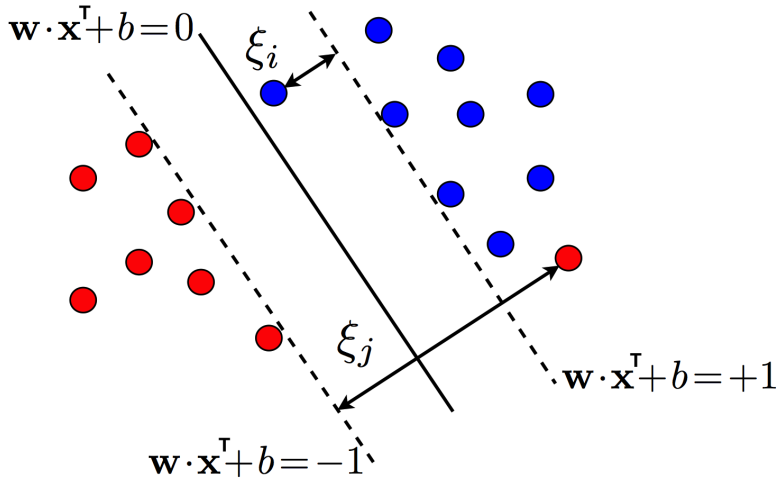
$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

- **Solution**: separating hyperplane $\mathbf{w} \cdot \mathbf{x}^\top + b$ has the form

$$h(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}^\top) + b \right),$$

with $b = y_i - \sum_{j=1}^{m} \alpha_j y_j (\mathbf{x}_j \cdot \mathbf{x}_i^\top)$ for any SV $\mathbf{x}_i$. Here $\alpha_i > 0$ only for SVs $\mathbf{x}_i$

SVM: Non-separable case



- Relax constraints using slack variables $\xi_i \geq 0$

$$y_i[\mathbf{w}\cdot\mathbf{x}_i^{\top}+b] \geq 1-\xi_i$$

- **Constrained Optimization**:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i$$

$$\text{s.t. } y_i(\mathbf{w}\cdot\mathbf{x}_i^{\top}+b) \geq 1-\xi_i \text{ and } \xi_i \geq 0,\ i \in [1,m]$$

- **Constrained Optimization**:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j^\top)$$

$$\text{s.t. } 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0,\ i \in [1, m]$$

- **Solution**: separating hyperplane $\mathbf{w} \cdot \mathbf{x}^\top + b = 0$

$$h(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}^\top) + b \right),$$

with $b = y_i - \sum_{j=1}^{m} \alpha_j y_j (\mathbf{x}_j \cdot \mathbf{x}_i^\top)$ for any SV $\mathbf{x}_i$ with $0 < \alpha_i < C$

- **Constrained Dual Optimization** problem:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \underbrace{\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)^\top}_{K(\mathbf{x}_i, \mathbf{x}_j)}$$

$$\text{s.t. } 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0,\ i \in [1, m]$$

- **Decision function** $h(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \Phi(\mathbf{x})^\top + b)$ has the form

$$h(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i \underbrace{\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})^\top}_{K(\mathbf{x}_i, \mathbf{x})} + b \right),$$

with

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \Phi(\mathbf{x}_i),\ b = y_i - \sum_{j=1}^{m} \alpha_j y_j \underbrace{\Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i)^\top}_{K(\mathbf{x}_j, \mathbf{x}_i)}$$

for any SV $\mathbf{x}_i$ with $0 < \alpha_i < C$

- Hypothesis set

$$\{x \to \mathbf{w} \cdot \Phi(\mathbf{x})^\top + b : \mathbf{w} \in \mathbb{R}^p, b \in \mathbb{R}\}$$

- Loss function: ϵ-insensitive loss

$$L(y, y') = |y - y'|_\epsilon = \max(0, |y' - y| - \epsilon)$$

- **Optimization problem**: similar to that of SVM

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} |y_i - (\mathbf{w} \cdot \Phi(\mathbf{x}_i)^\top + b)|_\epsilon \to \min_{\mathbf{w},b}$$

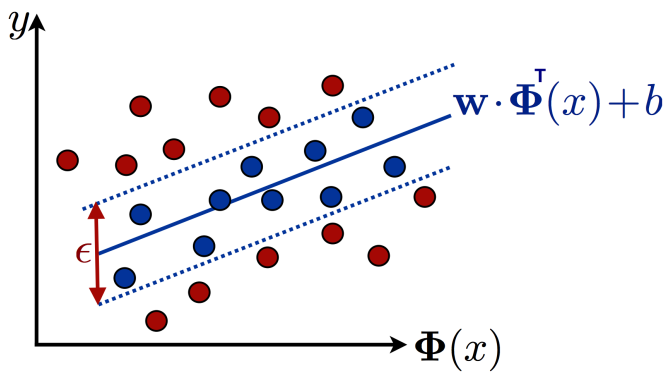
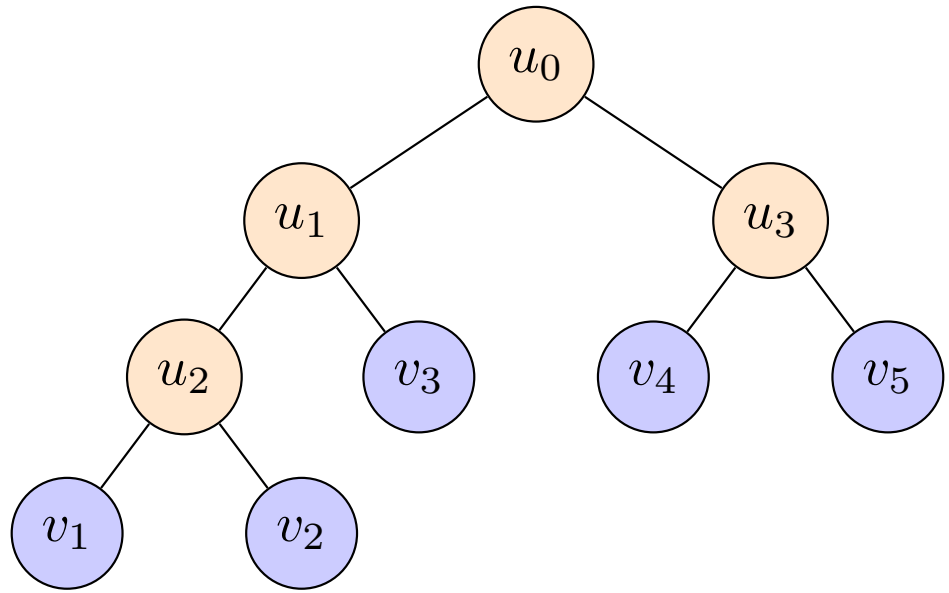


Figure – Fit "tube" with width ϵ to data

Binary Decision Tree

- Decision tree is often a binary tree DT
- Internal nodes $u \in DT$: *predicates* $\beta_u : X \to \{0, 1\}$
- Leafs $v \in DT$: *predictions* y
- Algorithm $h(\mathbf{x})$ starts at $u = u_0$
  - Compute $b = \beta_u(\mathbf{x})$
  - If $b = 0$, $u \leftarrow \text{LeftChild}(u)$
  - If $b = 1$, $u \leftarrow \text{RightChild}(u)$
  - If u is a leaf, return some y
- In practice for a real variable: $\beta_u(\mathbf{x}; j, t) = 1[x_j < t]$



Greedy binary tree learning for classification

- Input: training set $S_m = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{m}$
  1. Greedily split S_m into S_1 and S_2:

     $S_1(j,t) = \{(\mathbf{x}, y) \in S_m | x_j \leq t\}, \qquad S_2(j,t) = \{(\mathbf{x}, y) \in S_m | x_j > t\}$

     optimizing a given loss: $Q(S_m, j, t) \to \min\limits_{(j,t)}$
  2. Create internal node u corresponding to the predicate $1[x_j < t]$
  3. If a stopping criterion is satisfied for u,
     declare it a leaf, setting some $c_u \in Y$ as leaf prediction
  4. If not, repeat 1–2 for $S_1(j,t)$ and $S_2(j,t)$
- Output: a decision tree DT

Formal definition: Loss function $Q(S, j)$ for a binary tree

- S_t: the subset of S at step t
- With the current split, let $S_l \subseteq S_t$ go left and $S_r \subseteq S_t$ go right
- Choose predicate to optimize

$$Q(S_t, j) = E(S_t) - \frac{|S_l|}{|S_t|} E(S_l) - \frac{|S_r|}{|S_t|} E(S_r) \to \max$$

- $E(S)$: impurity criterion
- Generally

$$E(S) = \min_{c \in Y} \frac{1}{|S|} \sum_{(\mathbf{x}_i, y_i) \in S} L(y_i, c)$$

- **Regression:**
  - Impurity
$$E(S) = \min_{c \in Y} \frac{1}{|S|} \sum_{(\mathbf{x}_i, y_i) \in S} (y_i - c)^2$$
  - Sum of squared residuals minimized by
$$c = \frac{1}{|S|} \sum_{(\mathbf{x}_i, y_i) \in S} y_i$$
  - Impurity $\equiv$ variance of the target
- **Classification:**
  - Let (share of y_i's equal to k)
$$p_k = \frac{1}{|S|} \sum_{(\mathbf{x}_i, y_i) \in S} [y_i = k]$$
  - Miss rate:
$$E(S) = \min_{c \in \mathbb{Y}} \frac{1}{|S|} \sum_{(\mathbf{x}_i, y_i) \in S} [y_i \neq c]$$

    Impurity criterion is defined as

    Minimizing miss rate $E(S)$

The Random Forest algorithm

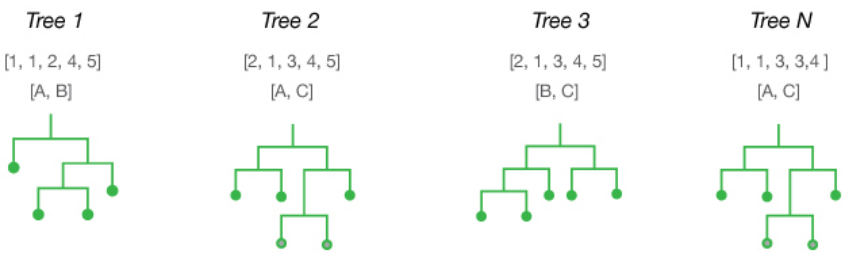
- Bagging over (weak) decision trees
- Reduce error via averaging over instances and features
- Input: a sample $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^d, y_i \in Y$
- The algorithm iterates for $i = 1, \ldots, N$:
  1. Pick p random features out of d
  2. Bootstrap a sample $S_m^i = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^p, y_i \in Y$
  3. Learn a decision tree $h_i(\mathbf{x})$ using bootstrapped S_m^i
  4. Stop when leafs in h_i contain less that $n_{\min}$ instances

$\mathbf{x}_i \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$

$S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^5$

Bootstrap $S_m^i, i \in \{1, 2, 3, 4\}$

Learn $\mathrm{Tree}_i(\mathbf{x})$ using S_m^i



Picture credit: http://www.thefactmachine.com/random-forests

Resampling method r:

1. Takes input:
   - dataset S_m;
   - resampling multiplier $m > 1$ for resulting imbalance ratio $IR(r(S_m)) = m \cdot IR(S_m)$;
   - additional parameters, specific for the method
2. Add synthesized objects to the minor class (oversampling), or drop objects from the major class (undersampling), or both
3. Outputs resampled dataset $r(S_m)$ with imbalance ratio $IR(r(S_m)) = m \cdot IR(S_m)$

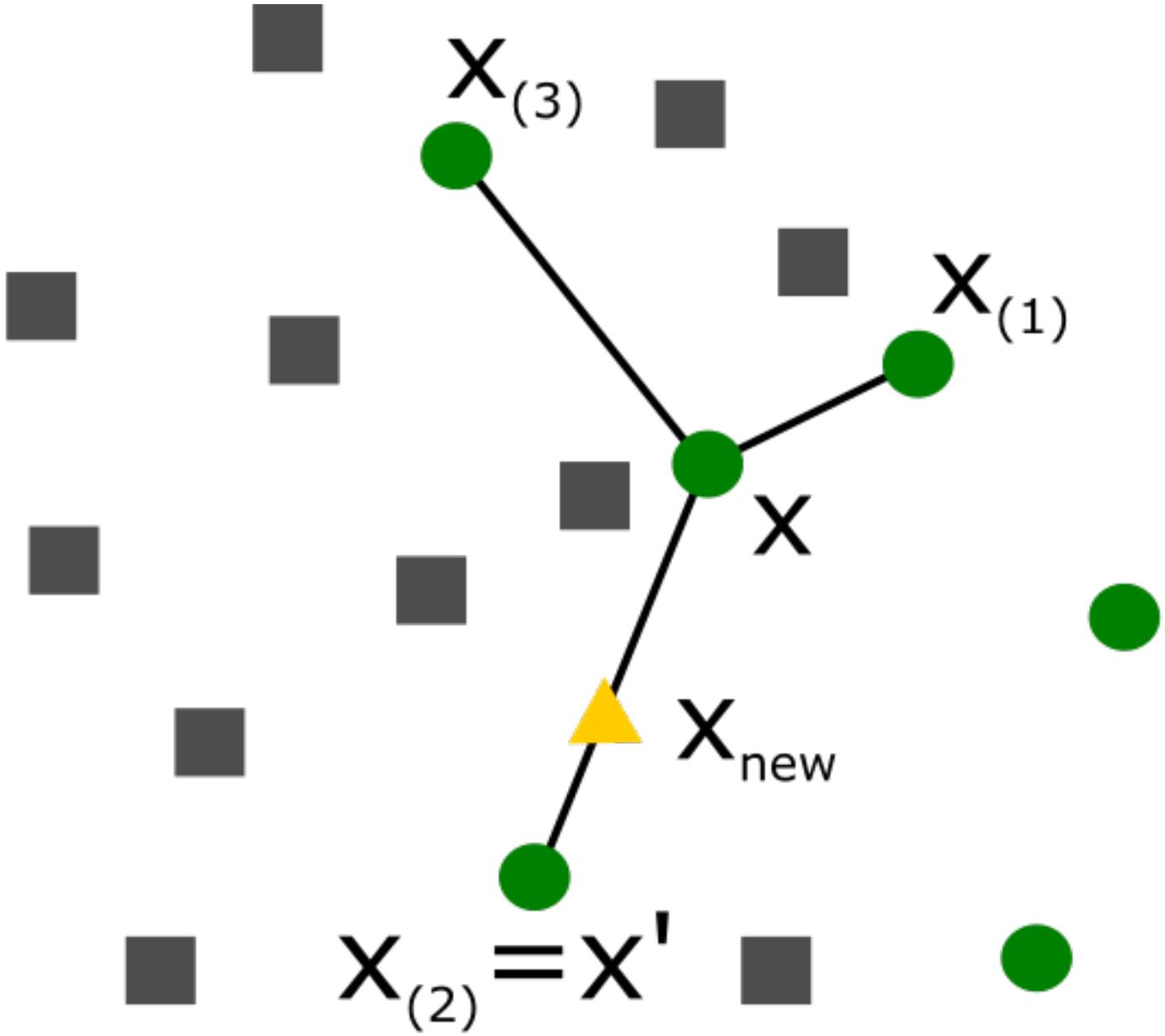
Random Oversampling (ROS)

- ROS, also known as bootstrap oversampling
- No additional input parameters
- It adds to the minor class new $(m-1)|C_{-1}(S_m)|$ objects
- Each of objects is drawn from uniform distribution on $C_{-1}(S_m)$

Random Undersampling (RUS)

- No additional input parameters
- It chooses random subset of $C_{+1}(S_m)$ with $|C_{+1}(S_m)|\frac{m-1}{m}$ elements and drops it from the dataset
- All subsets of $C_{+1}(S_m)$ have equal probabilities to be chosen

Synthetic Minority Oversampling Technique (SMOTE)

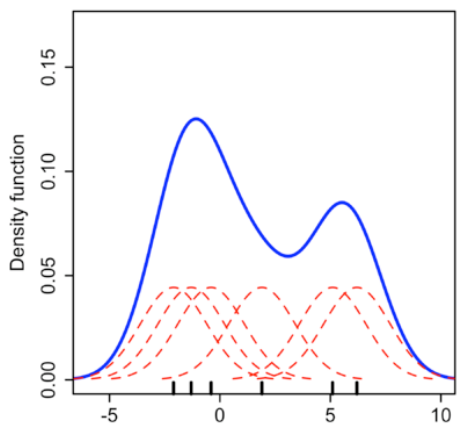


Kernel Density Estimation: Definition

- KDE has the form $\widehat{p}_m(\mathbf{x}) = \frac{1}{mh}\sum_{i=1}^{m} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$, h is a kernel width
- Kernel is a function K:

$$K(\mathbf{x}) \geq 0, \int_{\mathbb{R}} K(\mathbf{x})d\mathbf{x} = 1, \int_{\mathbb{R}} \mathbf{x}K(\mathbf{x})d\mathbf{x} = 0, \sigma_K^2 \equiv \int_{\mathbb{R}} \mathbf{x}^2 K(\mathbf{x})d\mathbf{x} < \infty$$

- Examples
  - $K(x) = \frac{1}{2}\mathbb{I}\{|x| < 1\}$ — rectangular kernel
  - $K(x) = (1 - |x|)\mathbb{I}\{|x| < 1\}$ — triangle kernel
  - $K(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$ — Gaussian kernel
  - $K(x) = \frac{3}{4}(1 - x^2)\mathbb{I}\{|x| < 1\}$ — Epanechnikov kernel



Nonparametric regression: Definition

- Let us consider m observations: $S_m = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, generated from a joint density $p(\mathbf{x}, y)$
- These observations are generated by the model

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$

where ε_i is an i.i.d white noise, $\mathbb{E}\varepsilon_i = 0$, $\mathbb{V}(\varepsilon_i) = \sigma^2$

- We should estimate a regression function

$$f(\mathbf{x}) = \mathbb{E}(y|\mathbf{x}) = \int_{\mathbb{R}} yp(y|\mathbf{x})dy = \frac{\int_{\mathbb{R}} yp(\mathbf{x}, y)dy}{\int_{\mathbb{R}} p(\mathbf{x}, y)dy} = \frac{\int_{\mathbb{R}} yp(\mathbf{x}, y)dy}{p(\mathbf{x})}$$

- Nadaraya-Watson estimate has the form

$$\widehat{f}_m^{NW}(\mathbf{x}) = \sum_{i=1}^{m} w_i(\mathbf{x}) y_i,$$

where

$$w_i = \frac{K(\frac{\mathbf{x}-\mathbf{x}_i}{h})}{\sum_{j=1}^{m} K(\frac{\mathbf{x}-\mathbf{x}_j}{h})}$$

What is Model Selection?

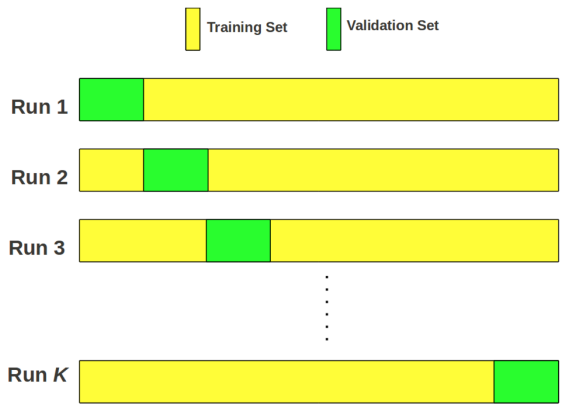
- Given a set of models $F = \{F_1, \ldots, F_K\}$, choose the model **expected to do the best on the test data**
- F may consist of
  1. Same learning model with different complexities or hyperparameters
     - Nonlinear regression: polynomials with different degress
     - k-Nearest Neighbors: Different choices of k
     - Decision Trees: different choices of the number of levels/leaves
     - SVM: different choices of the misclassification penalty hyperparameter C
     - Regularized Models: different choices of the regularization parameter
     - Kernel based Methods: different choices of kernels, etc.
     - Different subsets of features $(x_j, j \in J)$, $J \subseteq \{1, 2, \ldots, d\}$
  2. Different learning models (e.g., SVM, KNN, DT, etc.)
- **Note**: usually considered in supervised learning contexts but unsupervised learning also faces this issue (e.g., "how many clusters" when doing clustering)

Model Selection

- Occam's razor: among competing hypotheses, the one with the fewest assumptions should be selected
- Too much variables/parameters $\Rightarrow$ significant prediction variance and small bias on the training sample, and vice versa
- We have **two interrelated problems**
  - **Problem 1**. To estimate value of a target function, characterizing generalization ability of the considered model
  - **Problem 2**. To computationally efficiently select an optimal model w.r.t. to the constructed accuracy criterion

M-fold Cross-Validation

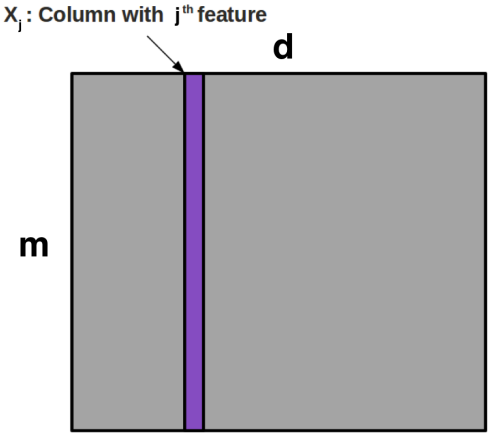
- Create M equal sized partitions of the training data
- Each partition has m/M examples
- Train using $M-1$ partitions, validate on the remaining partitions
- Repeat the same M times, each with a different validation partition



- Finally, choose the model with smallest average validation error
- Usually M is chosen as $4-10$

Filter Feature Selection

- Uses heuristics but is much faster than wrapper methods



- Correlation Criteria: rank features in order of their correlation with the labels

$$r(x_j, y) = \frac{\text{cov}(x_j, y)}{\sqrt{\text{var}(x_j)\text{var}(y)}}$$

- Mutual Information Criteria:

$$MI(x_j, y) = \sum_{x_j} \sum_{y} p(x_j, y) \log \frac{p(x_j, y)}{p(x_j)p(y)}$$

  - High mutual information mean high relevance of that feature
  - Note: these probabilities can be easily estimated from the data

Wrapper Feature Selection

Two types: Forward Search and Backward Search

- Forward Search
- Backward Search
- Inclusion/Removal criteria uses cross-validation

- Regularization penalizes complex models F

$$\widehat{R}_{\text{pen}}(f; S_m) = \widehat{R}(f; S_m) + \text{pen}(F)$$

- Let us consider linear models
  - $F = \{f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x}^\top)\}$ (classification) or
  - $F = \{f(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}^\top)\}$ (regression)
- Then
  a) L_2-regularization $\text{pen}(F) = \lambda \sum_{j=1}^{d} w_j^2$
  b) L_1-regularization $\text{pen}(F) = \lambda \sum_{j=1}^{d} |w_j|$ (Embedded Feature Selection!)
  c) L_0-regularization $\text{pen}(F) = \lambda \sum_{j=1}^{d} 1_{w_j \neq 0}$

AIC & BIC

- AIC (Akaike Information Criterion) provides estimate of the risk in case of more general models. It has the form

$$\mathcal{L}_J - |J| \to \max_{\mathbf{w}_J, J},$$

  where
  - $\mathcal{L}_J$ is a model log-likelihood
  - $|J|$ is a number of free model parameters
- BIC (Bayesian Information Criterion) is equal to

$$\mathcal{L}_J - |J| \log m \to \max_{\mathbf{w}_J, J}$$

- We consider a binary classification with $Y = \{-1, +1\}$
- As a strong classifier we consider weighted voting scheme, i.e.

$$\widehat{f}_T(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

- As a risk we consider accuracy, i.e.

$$\widehat{R}(f_T) = \frac{1}{m}\sum_{i=1}^{m} 1\left\{y_i \cdot \left[\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)\right] \leq 0\right\}$$

- Two main heuristics, underlying boosting
  - We fix $\alpha_1 h_1(\mathbf{x}), \ldots, \alpha_{t-1} h_{t-1}(\mathbf{x})$ when adding $\alpha_t h_t(\mathbf{x})$
  - We use continuous upper bound for accuracy

- Since $1_{z \leq 0} \leq e^{-z}$, we get that

$$1_{\left\{y_i \cdot \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i) \leq 0\right\}} \leq \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)\right)$$

- Let us consider an upper bound for $\widehat{R}(f)$

$$\widehat{R}(f_T) = \frac{1}{m} \sum_{i=1}^{m} 1_{\left\{y_i \cdot \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i) \leq 0\right\}} \leq$$

$$\leq \widetilde{R}(f_T) = \frac{1}{m} \sum_{i=1}^{m} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)\right)$$

- We will optimize $\widehat{R}(f_T)$ w.r.t. a new weak classifier $h_T(\mathbf{x})$ and its weight α_T given that $\{\alpha_t, h_t(\mathbf{x})\}_{t=1}^{T-1}$ are fixed

$\mathrm{AdaBoost}(S_m = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\})$

1. **for** $i \leftarrow 1$ **to** m **do**
2. $\quad w_{i,1} \leftarrow \frac{1}{m}$
3. **for** $t \leftarrow 1$ **to** T **do**
4. $\quad$ Learn a based classifier:
   $h_t \leftarrow$ base classif. with small $N(h_t, \widetilde{\mathbf{w}}_t) = \sum_{i=1}^{m} \widetilde{w}_{i,t} 1_{\{y_i \cdot h_t(\mathbf{x}_i) \leq 0\}}$
5. $\quad \alpha_t \leftarrow \frac{1}{2} \log \frac{1 - N(h_t, \widetilde{\mathbf{w}}_t)}{N(h_t, \widetilde{\mathbf{w}}_t)}$
7. $\quad$ **for** $i \leftarrow 1$ **to** m **do**
8. $\quad\quad w_{i,t+1} \leftarrow w_{i,t} \exp(-\alpha_t y_t h_t(\mathbf{x}_i))$
9. $\quad\quad \widetilde{w}_{i,t+1} \leftarrow \frac{w_{i,t+1}}{\sum_{j=1}^{m} w_{j,t+1}}$
10. $\quad f_t \leftarrow \sum_{s=1}^{t} \alpha_s h_s$
10. **return** $\widehat{f}_T = \mathrm{sign}(f_T)$

- With $f_{T-1}(\mathbf{x})$ already built, how to find the next α_T and h_T if

$$\sum_{i=1}^{m} L(y_i, f_{T-1}(\mathbf{x}_i) + \alpha_T h_T(\mathbf{x}_i)) \to \min_{\alpha_T, h_T}$$

- Recall: functions decrease in the direction of negative gradient
- View $L(y, z)$ as a function of z $(= f_T(\cdot))$, execute gradient descent on z
- Search for such $s_1, \ldots, s_m$ that

$$\sum_{i=1}^{m} L(y_i, f_{T-1}(\mathbf{x}_i) + \alpha \cdot s_i) \to \min_{\{s_1,\ldots,s_m\},\alpha}$$

- Choose $s_i = -\left.\frac{\partial L(y_i,z)}{\partial z}\right|_{z=f_{T-1}(\mathbf{x}_i)}$
- Approximate s_i's by $h_T(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}_i$ and tune α

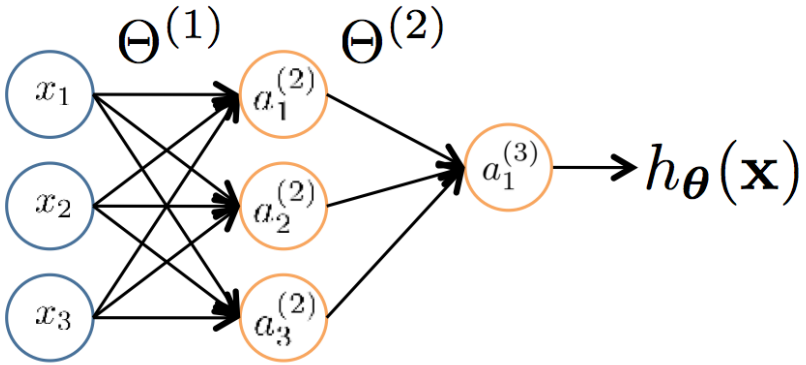
Perceptron Algorithm

Perceptron($\mathbf{w}_0$)

1. $\mathbf{w}_1 \leftarrow \mathbf{w}_0$ (typically $\mathbf{w}_0 = 0$)
2. **for** $t \leftarrow 1$ **to** T **do**
3. Receive($\mathbf{x}_t$)
4. $\widehat{y}_t \leftarrow \text{sgn}(\mathbf{w}_t \cdot \mathbf{x}_t)$
5. Receive(y_t)
6. **if** $(\widehat{y}_t \neq y_t)$ **then**
7. $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t \mathbf{x}_t$ (or $\leftarrow \mathbf{w}_t + \eta y_t \mathbf{x}_t$, $\eta > 0$)
8. **else** $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
9. **return** $\mathbf{w}_{T+1}$

- A margin-based upper bound on the number of mistakes or updates made by the Perceptron algorithm when processing a sequence of T points that can be linearly separated by a hyperplane with margin $\rho > 0$
- **Theorem**:
  - Let $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^d$ be a sequence of T points with $\|\mathbf{x}_t\| \leq R$ for all $t \in [1, T]$, for some $R > 0$.
  - Assume that there exists $\rho > 0$ and $\mathbf{v} \in \mathbb{R}^d$ such that for all $t \in [1, T]$, $\rho \leq \frac{y_t(\mathbf{v} \cdot \mathbf{x}_t)}{\|\mathbf{v}\|}$.
  - Then, the number of updates (= the number of mistakes) made by the Perceptron algorithm when processing $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^d$ is bounded by $\frac{R^2}{\rho^2}$.
- **Proof**: Let I be the set of t-s at which there is an update and let M be the total number of updates

Neural Network: Equations



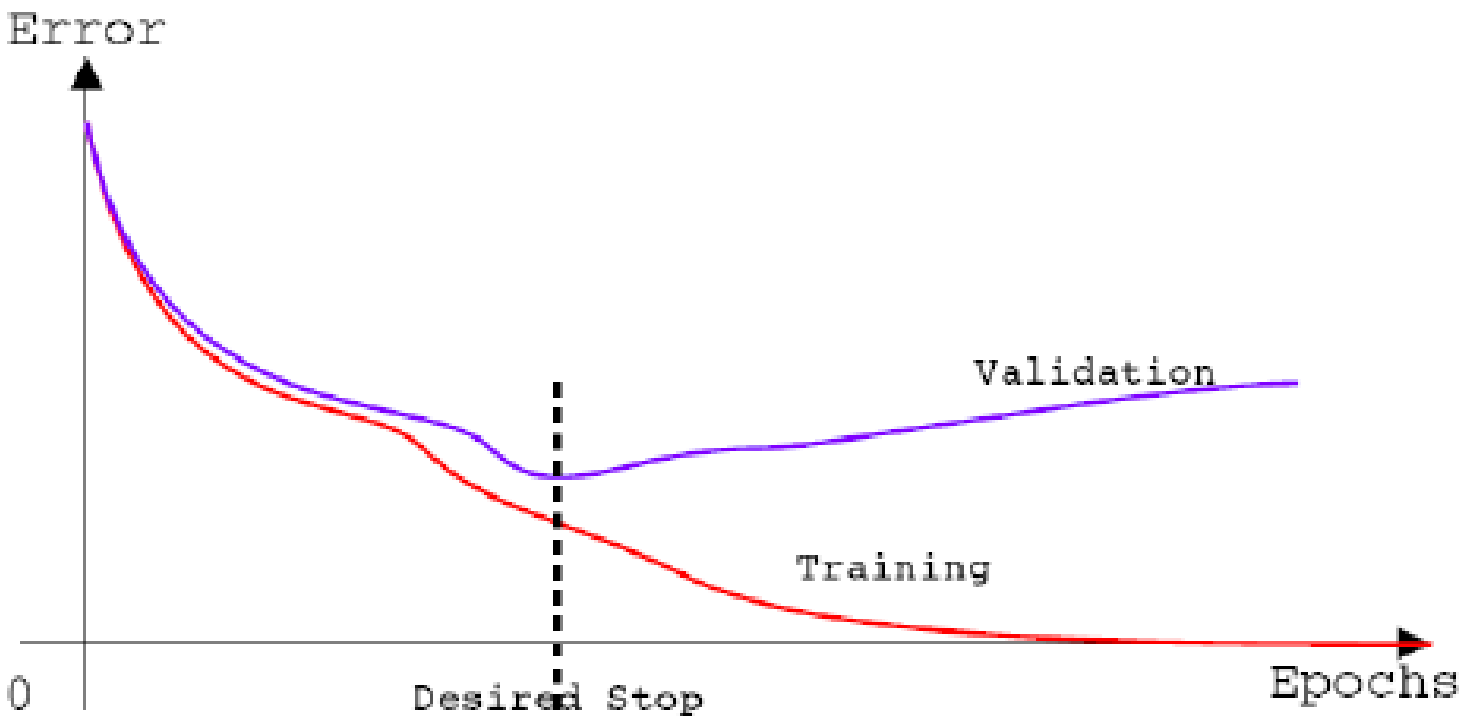
- $a_i^{(j)}$ = "activation" of unit i in layer j
- $\Theta^{(j)}$ = weight matrix controlling function mapping from layer j to layer $j+1$

$$a_1^{(2)} = g\left(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3\right)$$
$$a_2^{(2)} = g\left(\theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3\right)$$
$$a_3^{(2)} = g\left(\theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3\right)$$
$$h_\theta(\mathbf{x}) = a_1^{(3)} = g\left(\theta_{10}^{(2)}a_0^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)} + \theta_{13}^{(2)}a_3^{(2)}\right)$$

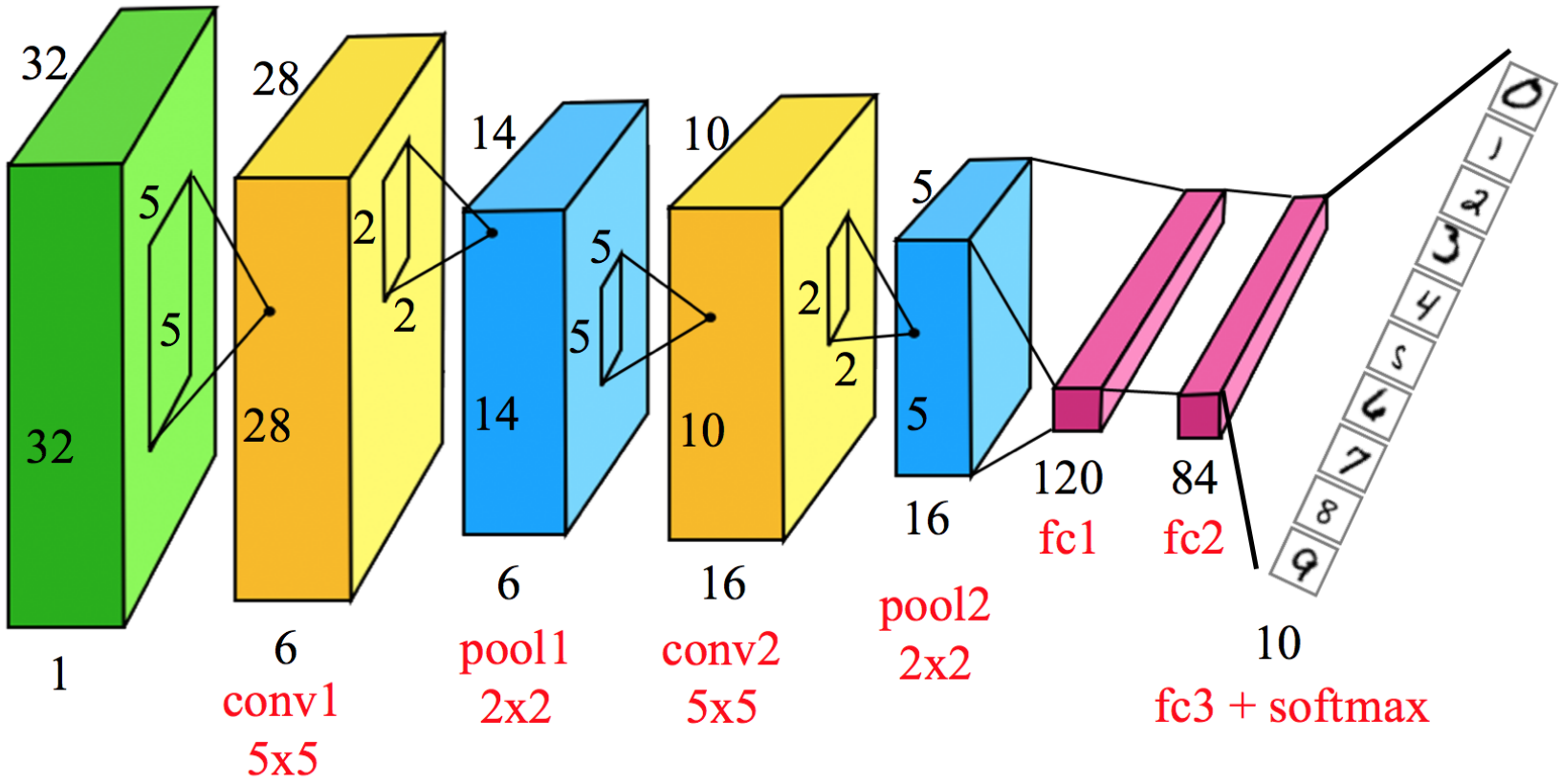
If network has s_j units in layer j and s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ has dimension $s_{j+1} \times (s_j + 1)$: $\Theta^{(1)} \in \mathbb{R}^{3\times 4}$, $\Theta^{(2)} \in \mathbb{R}^{1\times 4}$

Backprop issues

- Local minima
- Random initialization
- Random division into train and validation sets
- Second order optimization methods
- Architecture selection
- Early stopping

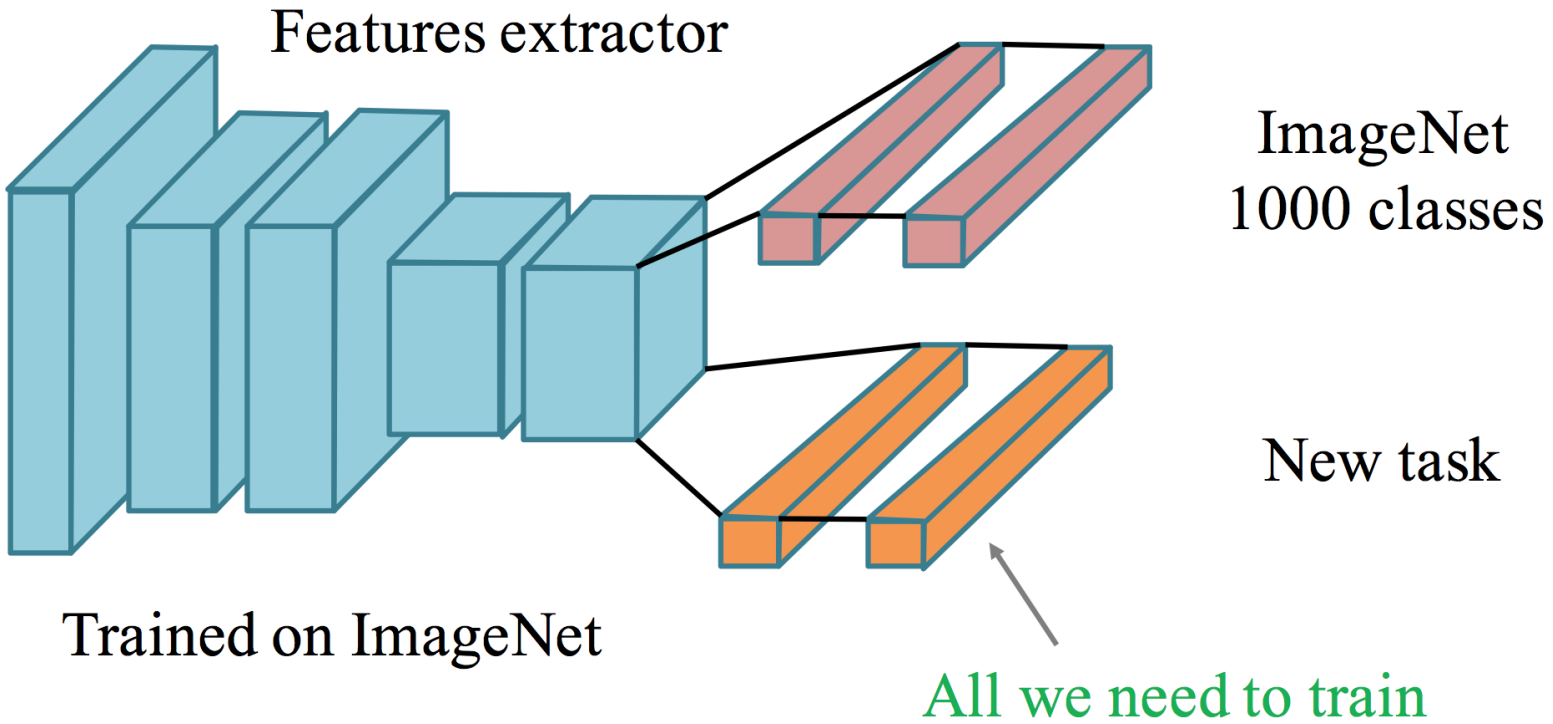


Convolutional network

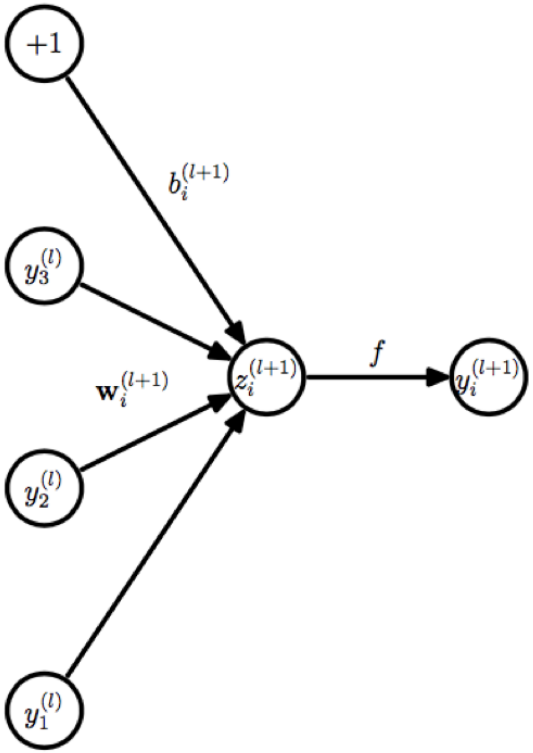


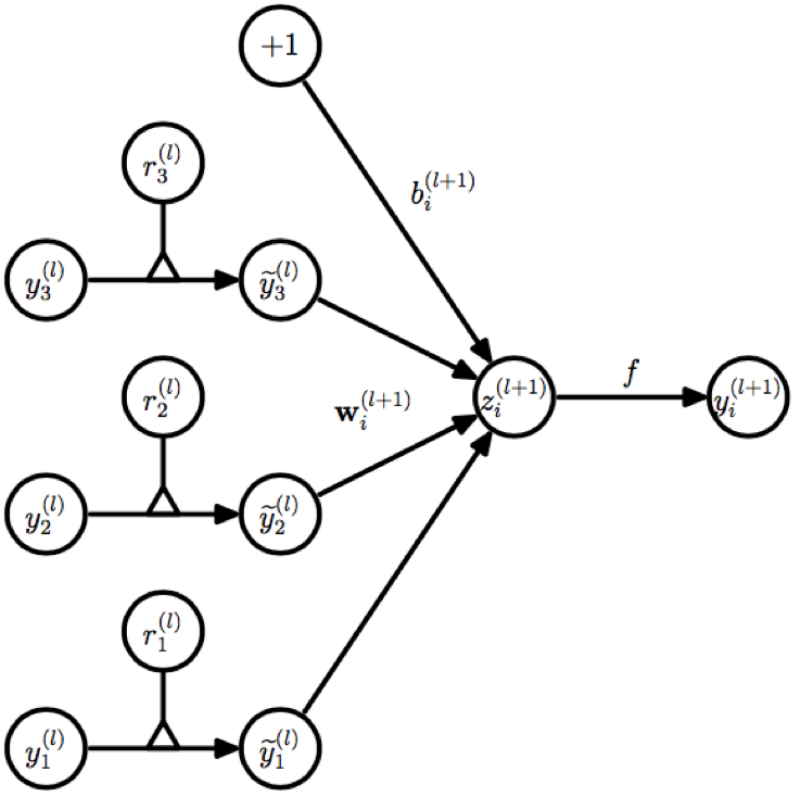
http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

- Deep networks learn complex features extractor, but we need lots of data to train it from scratch!
- What if we can reuse an existing features extractor for a new task?



Dropout


+1
$b_i^{(l+1)}$
$y_3^{(l)}$
$\mathbf{w}_i^{(l+1)}$
$z_i^{(l+1)}$
f
$y_i^{(l+1)}$
$y_2^{(l)}$
$y_1^{(l)}$


+1
$r_3^{(l)}$
$b_i^{(l+1)}$
$y_3^{(l)}$
$\tilde{y}_3^{(l)}$
$r_2^{(l)}$
$\mathbf{w}_i^{(l+1)}$
$z_i^{(l+1)}$
f
$y_i^{(l+1)}$
$y_2^{(l)}$
$\tilde{y}_2^{(l)}$
$r_1^{(l)}$
$y_1^{(l)}$
$\tilde{y}_1^{(l)}$

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$

- The prior is

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- The noise model, or likelihood is

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \sigma^2 \mathrm{I}_m\right)$$

- Integrating over the function variables $\mathbf{f}$ we get the marginal likelihood

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) d\mathbf{f} = \mathcal{N}(\mathbf{0},\, \mathbf{K} + \sigma^2 \mathrm{I}_m)$$

- Let us denote input test point as $\mathbf{x}_*$, and output

$$y_* = f_* + \varepsilon_*, \;\; f_* = f(\mathbf{x}_*)$$

- Consider joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^\top & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$
- What we know about noiseless value $f(\mathbf{x}_*)$?

Prediction

- Joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^\top & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$
- Condition on training outputs we get

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- In fact μ_* has the form

$$\mu_* = \sum_{i=1}^m \alpha_i K(\mathbf{x}_*, \mathbf{x}_i), \quad \boldsymbol{\alpha} = [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y}$$

- Computational complexity: $O(m^3)$ for inversion in $\boldsymbol{\alpha}$, $O(m)$ for μ_* and $O(m^2)$ for σ_*^2

Methodology to perform global optimization of multimodal black-box functions

1. Choose some prior measure over the space of possible objectives f
2. Combine prior and the likelihood to get a posterior over the objective given some observations
3. Use the posterior to decide where to take the next evaluation according to some acquisition function
4. Augment the data set
5. Iterate between 2 and 4 until the evaluation budget is over

Comment: BO can be theoretically formalized in the framework of dynamic programming principle

- Use GP $\mathcal{GP}(\cdot|\mu(\mathbf{x}), K(\mathbf{x}, \mathbf{x}'))$ as a prior for $f(\cdot)$
- GP has marginal closed-form for the posterior mean $\mu_*(\mathbf{x})$ and variance $\sigma_*^2(\mathbf{x}) \Rightarrow$ efficient calculation of acquisition function
  - **Exploration**: Evaluate in places where the variance is large
  - **Exploitation**: Evaluate in places where the mean is low
- **Acquisition functions balance these two factors to determine where to evaluate next**
- E.g. we can use GP Upper (lower) Confidence Band

$$\alpha_{LCB}(\mathbf{x}) = -\mu_*(\mathbf{x}) + \zeta \cdot \sigma_*(\mathbf{x})$$

The procedure of choosing new point can be formulated as maximization of some criterion:

$$\mathbf{x}_{m+1} = \arg\max_{\mathbf{x}\in\mathbb{X}} \mathcal{I}(\mathbf{x}|S_m, \widehat{f}_m, \widehat{\sigma}^2_m),$$

where $\mathcal{I}(\mathbf{x}|S, \widehat{f}, \widehat{\sigma}^2)$ is a criterion of point selection based on

- current training set S,
- current approximation model $\widehat{f}$ and
- current error of approximation $\widehat{\sigma}^2$

- Maximum variance criterion:

$$\mathcal{I}_{MV}(\mathbf{x}) = \widehat{\sigma}^2(\mathbf{x}|S),$$

  where $\widehat{\sigma}^2(\mathbf{x}|S)$ is an error at point $\mathbf{x}$ of the model trained on $S = (\mathbf{X}, \mathbf{y})$
- As $\widehat{\sigma}^2(\mathbf{x}|S)$ we can use GP-based posterior variance $\sigma^2_*(\mathbf{x})$

Problem: find $\mathbb{R}^p$ an affine subspace

$$L(q) = \left\{\mathbf{x} \in \mathbb{R}^p : \mathbf{x} = \mathbf{x}_0 + \sum_{j=1}^{q} z_j \times \mathbf{e}_j,\ (z_1, z_2, \ldots, z_q) \in \mathbb{R}^q\right\}$$

of dimension $q < p$, **which best approximates the set of points**

$$\mathbf{X}_m = \left\{\mathbf{x}_i, i = 1, 2, \ldots, m\right\} \subset \mathbb{R}^p$$

in PCA: "the best" = minimize w.r.t. $\mathbf{x}_0, \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_q\} \subset \mathbb{R}^p$

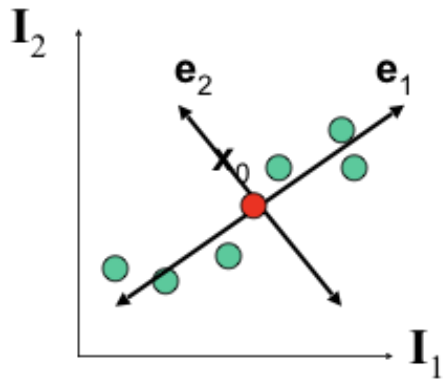
$$\frac{1}{m}\sum_{j=1}^{m} \left\| \mathbf{x}_j - P_{L(q)}\mathbf{x}_j \right\|^2$$

$$\mathrm{Pr}_{L(q)}(\mathbf{x}) = \mathbf{x}_0 + \sum_{j=1}^{q} z_j(\mathbf{x}) \times \mathbf{e}_j, \quad z_j(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_0, \mathbf{e}_j)$$

$\mathbf{x}_{\text{mean}}$ — empirical mean of $\{\mathbf{x}_i, i = 1, 2, \ldots, m\}$
$\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p\}$ — eigenvalues of an empirical covariance $(p \times p)$-matrix

$$\Sigma = \frac{1}{m} \sum_{j=1}^{m} (\mathbf{x}_j - \mathbf{x}_{\text{mean}}) \times (\mathbf{x}_j - \mathbf{x}_{\text{mean}})^\top$$

providing orthonormal basis in $\mathbb{R}^p$



$$L(1) = \{\mathbf{x} \in R^2 : \mathbf{x} = \mathbf{x}_0 + z_1 \times \mathbf{e}_1, z_1 \in R^1\},\ p = 2,\ q = 1$$

Solution: $\mathbf{x}_0 = \mathbf{x}_{\text{mean}},\ L(q) = \mathbf{x}_0 \bigoplus \text{Span}(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_q)$,

where orthonormal eigenvectors $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_q\}$ of Σ correspond to q highest eigenvalues of this matrix

- Build a sparse graph with k-nearest neighbours

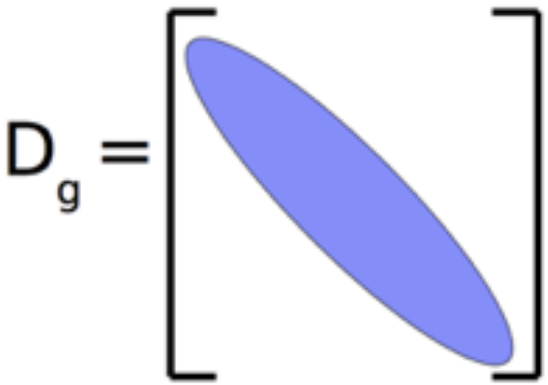
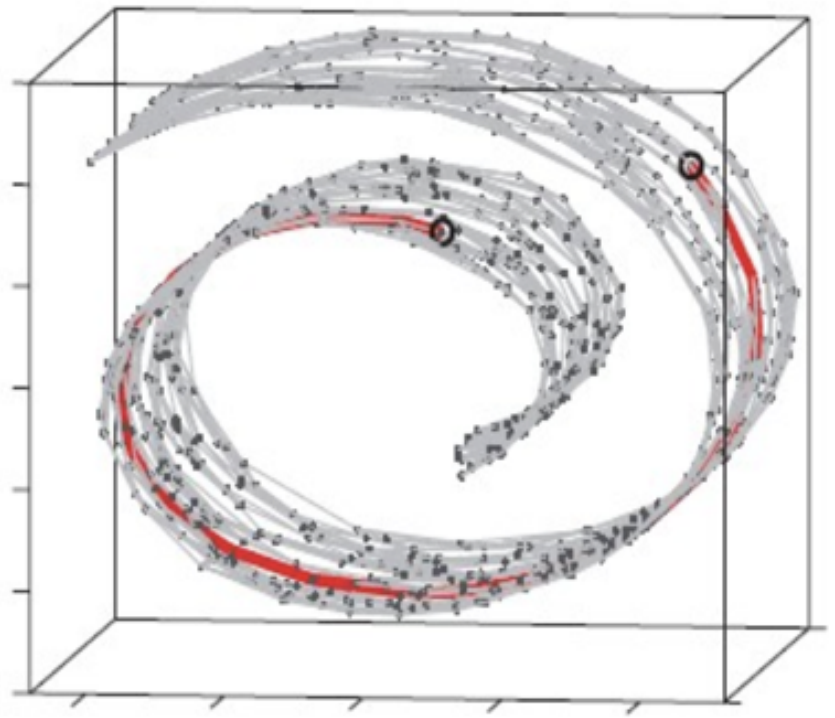
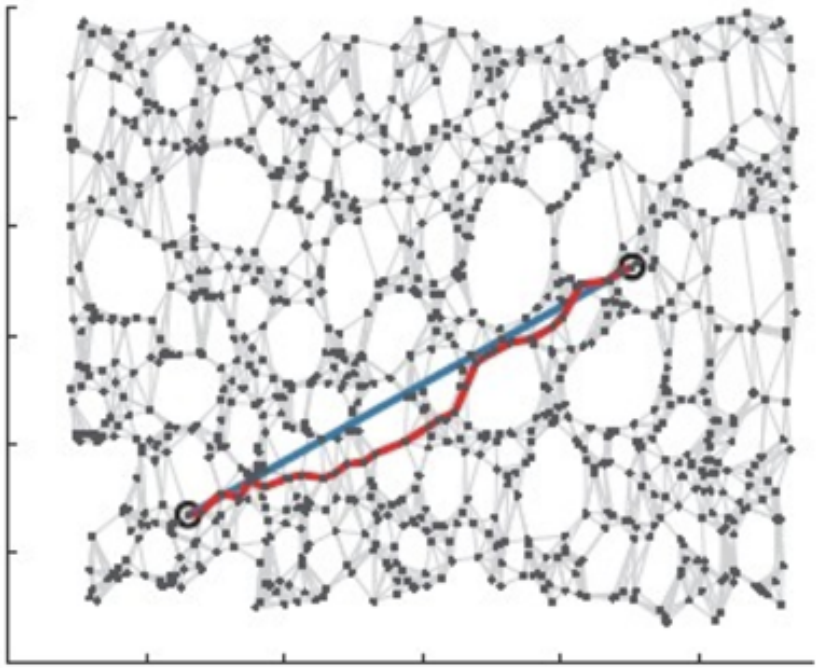


Figure – Distance matrix is sparse



- Infer other interpoint distances by finding shortest paths on the graph (Dijkstra's algorithm).

$$D_g = \left[\right]$$



Sammon mapping for ISOMAP

Common useful idea: local distances are often more important to preserve

$$\sum_{i,j} \frac{\big(\rho(O_i, O_j) - \|\mathbf{z}_i - \mathbf{z}_j\|\big)^2}{\big(\rho(O_i, O_j)\big)^2} \to \min_{\mathbf{x}_1, \ldots, \mathbf{x}_m}$$

Stochastic Neighbour Embedding

- "Stochastic" similarity in the initial space

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}$$

- Similarity in the compressed space

$$q_{j|i} = \frac{\exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2)}{\sum_{k\neq i}\exp(-\|\mathbf{z}_i - \mathbf{z}_k\|^2)}$$

- Cost function

$$\text{Cost} = \sum_i KL(P_i \parallel Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

- Randomly **initialize** k centers:

$$\mu^0 = (\mu_1^0, \dots, \mu_k^0).$$

- **Classify:** Assign each point $j \in \{1, \dots, m\}$ to nearest center:

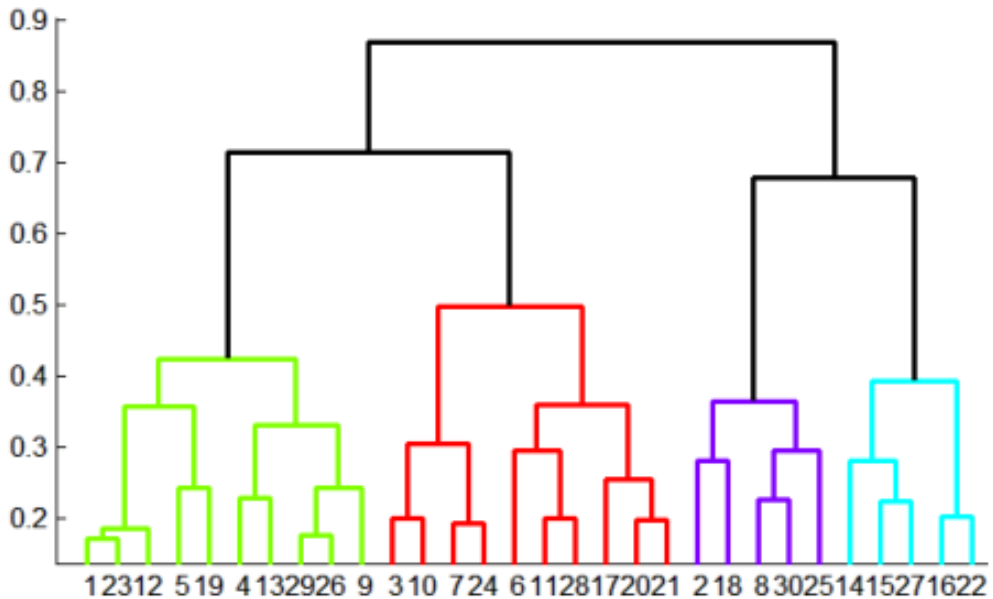
$$z^j = \arg\min_i \|\mathbf{x}_j - \mu_i^t\|_2^2.$$

- **Recenter:** μ_i becomes centroid of its points:

$$\mu_i^{t+1} = \arg\min_\mu \sum_{j:z^j=i} \|\mathbf{x}_j - \mu\|_2^2.$$

  Equivalent to μ_i average of its points!

- **Agglomerative (bottom up)**:
  - Initially, each point is a cluster;
  - Repeatedly combine the two "nearest" clusters into one.
- **Divisive (top down)**:
  - Start with one cluster and recursively split it.



- Mixture distribution:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}|z=k).$$

- Each component is a Gaussian distribution:

$$p(\mathbf{x}|z=k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$