ML → box 40%

Data → ?
↳ Pattern

Data Analysis

Statistics Prediction

Dimensionality Reduction

- Dimensionality reduction refers to techniques for reducing the number of input variables in training data.

x4

- When dealing with high dimensional data, it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the “essence” of the data. This is called dimensionality reduction.

multidimensio
→ Not increasing

dimensal $\leftarrow$ Data modelin

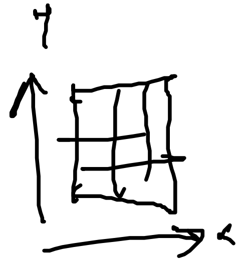
- Dimensionality reduction is a data preparation technique performed on data prior to modeling. It might be performed after data cleaning and data scaling and before training a predictive model.

Features $\rightarrow$

$\downarrow$

testing

bin 3 = 3



3^2 3^3

Techniques for Dimensionality Reduction

Extraction

already i here

↓

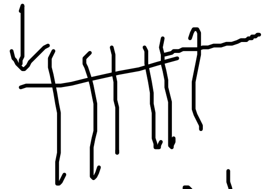
some mapping

↓

new feature

- **Feature Selection Methods**
- Perhaps the most common are so-called feature selection techniques that use scoring or statistical methods to select which features to keep and which features to delete.

↓



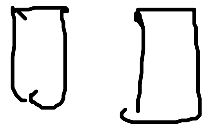
graduation

- perform feature selection, to remove “irrelevant” features that do not help much with the classification problem.

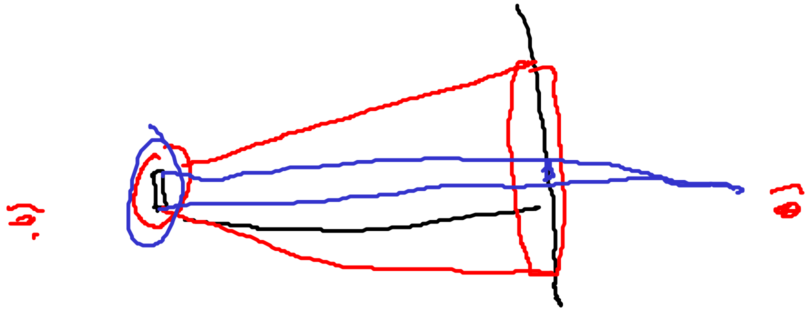
Name | DOB | Grade / reached | gender

- Two main classes of feature selection techniques include wrapper methods and filter methods.

- Wrapper methods, as the name suggests, wrap a machine learning model, fitting and evaluating the model with different subsets of input features and selecting the subset the results in the best model performance. RFE is an example of a wrapper feature selection method.
- Filter methods use scoring methods, like correlation between the feature and the target variable, to select a subset of input features that are most predictive. Examples include Pearson's correlation and Chi-Squared test.



- **Matrix Factorization**
- Techniques from linear algebra can be used for dimensionality reduction.
- Specifically, matrix factorization methods can be used to reduce a dataset matrix into its constituent parts.
- Examples include the eigendecomposition and singular value decomposition.
- The parts can then be ranked and a subset of those parts can be selected that best captures the salient structure of the matrix that can be used to represent the dataset.
- The most common method for ranking the components is principal components analysis, or PCA for short.



- **Manifold Learning**
- Techniques from high-dimensionality statistics can also be used for dimensionality reduction.
- In mathematics, a projection is a kind of function or mapping that transforms data in some way.
- These techniques are sometimes referred to as "*manifold learning*" and are used to create a low-dimensional projection of high-dimensional data, often for the purposes of data visualization.
- The projection is designed to both create a low-dimensional representation of the dataset whilst best preserving the salient structure or relationships in the data.

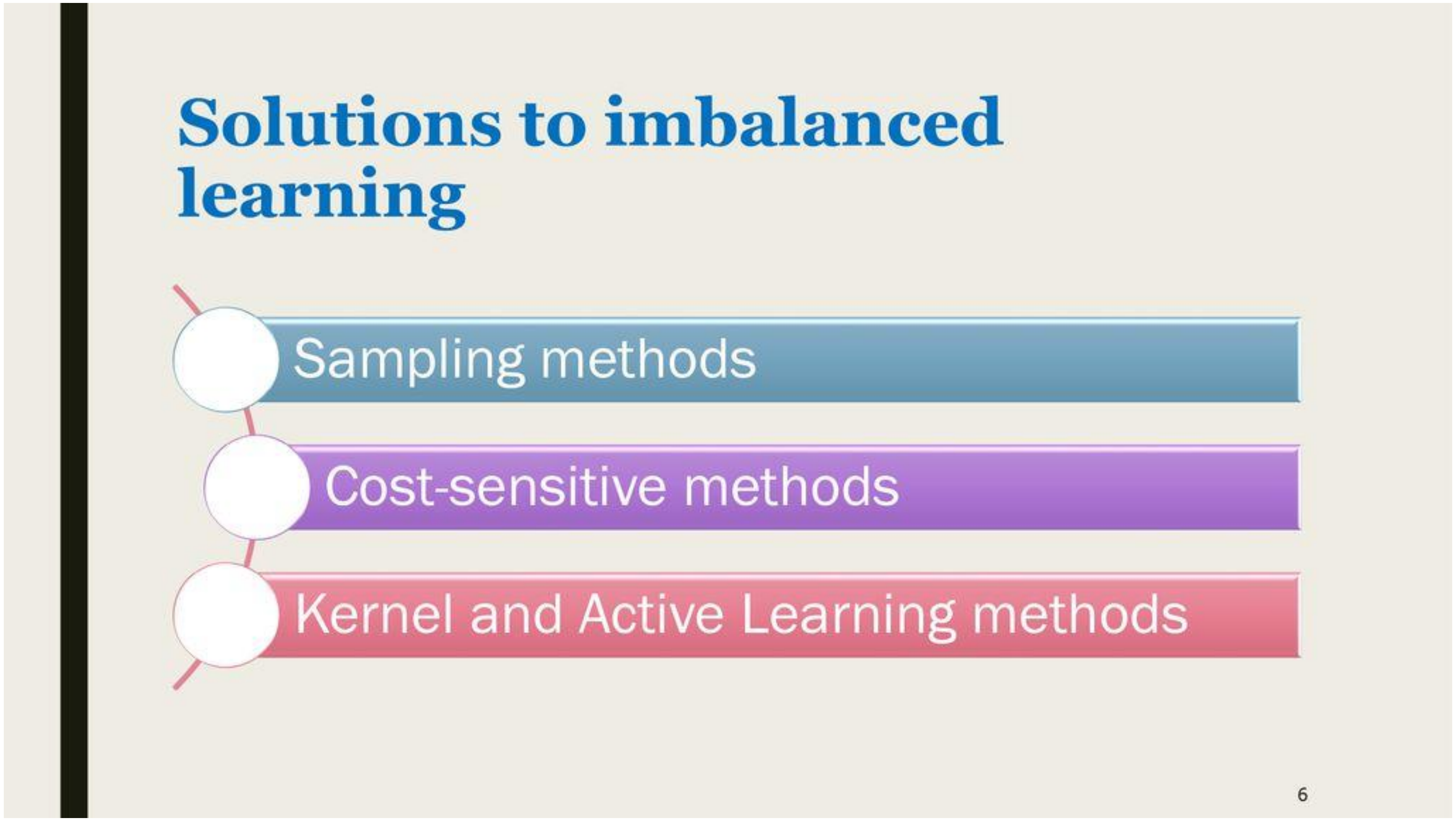
- **Autoencoder Methods** $\nearrow$
- Deep learning neural networks can be constructed to perform dimensionality reduction.
- A popular approach is called autoencoders. This involves framing a self-supervised learning problem where a model must reproduce the input correctly.

- An auto-encoder is a kind of unsupervised neural network that is used for dimensionality reduction and feature discovery. More precisely, an auto-encoder is a feedforward neural network that is trained to predict the input itself.

PCA
LDA

Sampling

Data

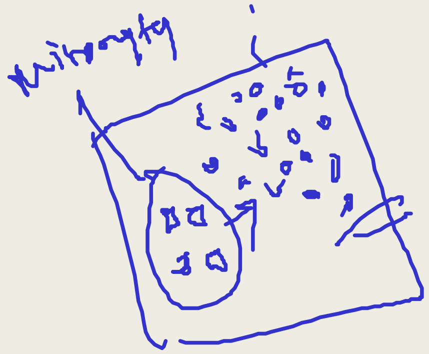


Solutions to imbalanced learning

- Sampling methods
- Cost-sensitive methods
- Kernel and Active Learning methods

6

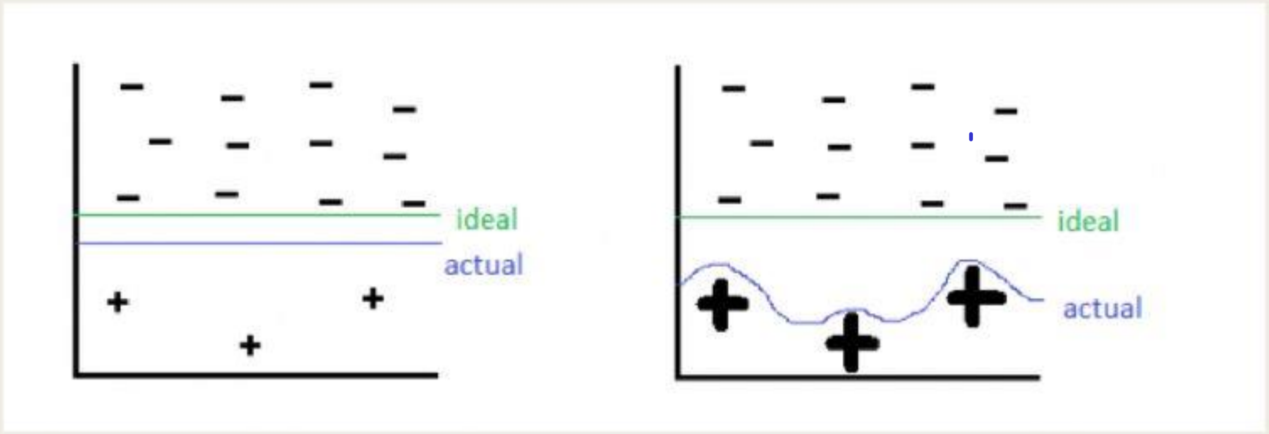
Sampling **methods**



- Oversampling
  - *by adding more of the minority class so it has more effect on the machine learning algorithm*
- Under-sampling
  - *by removing some of the majority class so it has less effect on the machine learning algorithm*

Oversampling

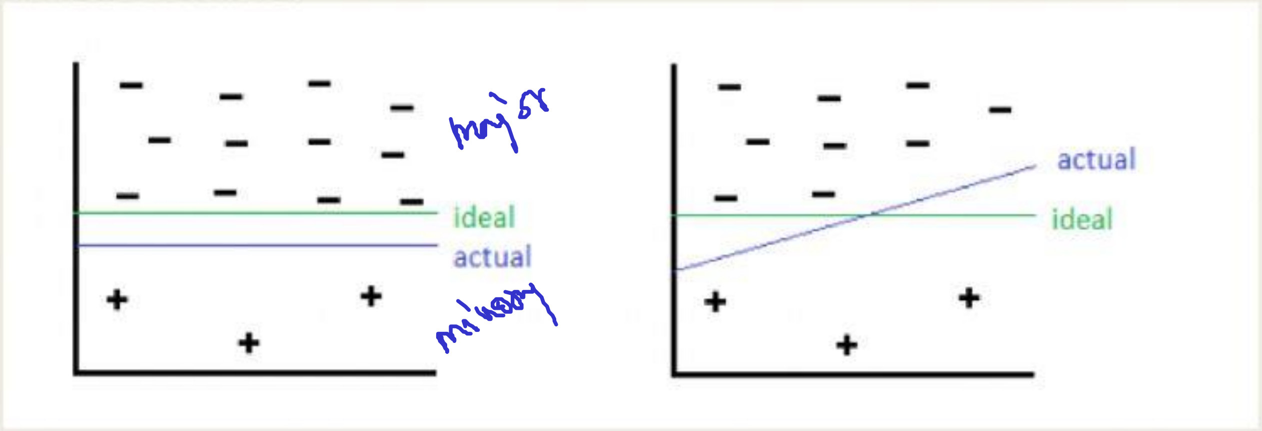
- By oversampling, just duplicating the minority classes could lead the classifier to overfitting to a few examples, which can be illustrated below:



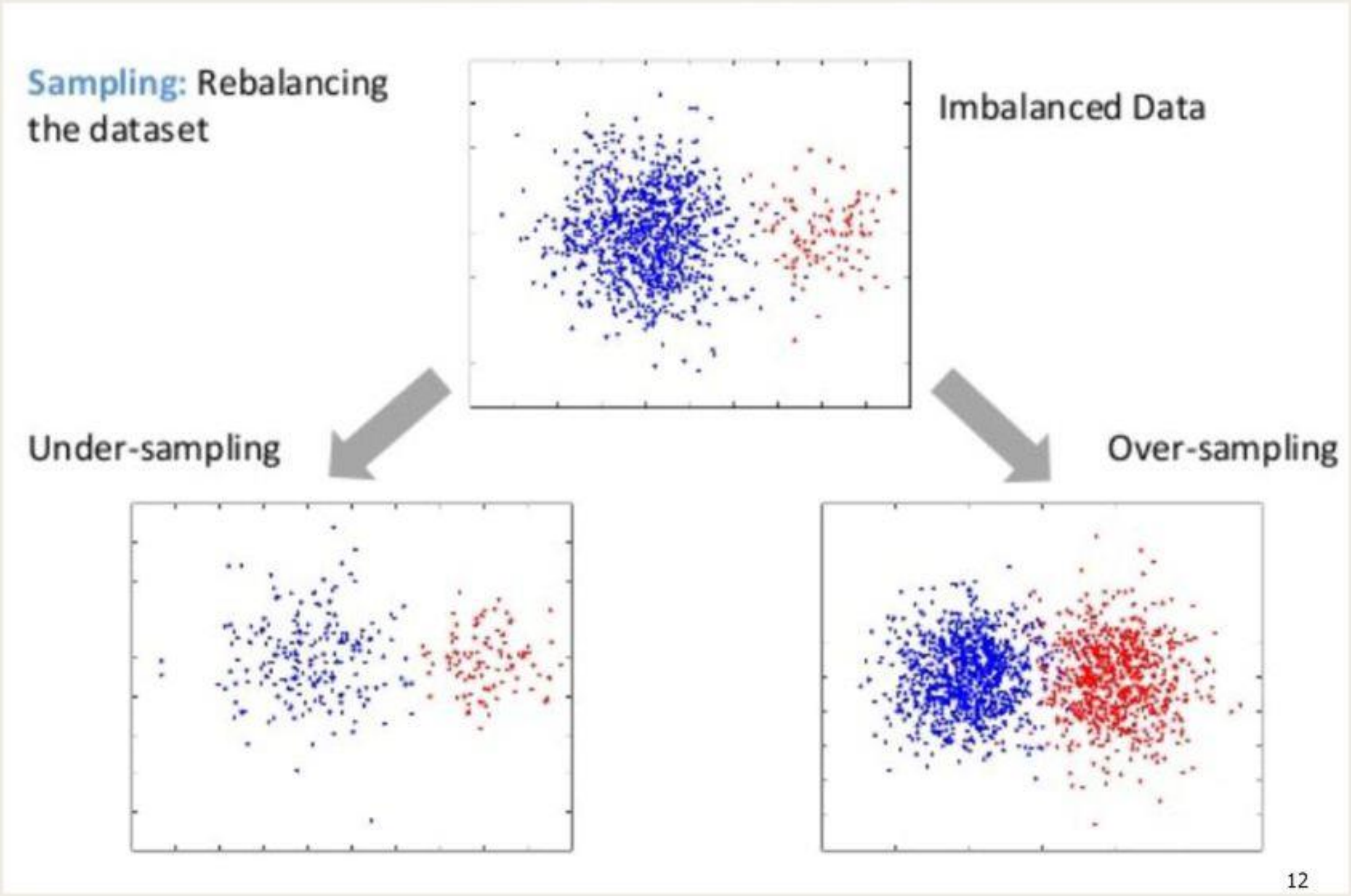
- On the left hand side is before oversampling, where as on the right hand side is oversampling has been applied. On the right side, The thick positive signs indicate there are multiple repeated copies of that data instance. The machine learning algorithm then sees these cases many times and thus designs to overfit to these examples specifically, resulting in a blue line boundary as above.

Under-sampling

- By undersampling, we could risk removing some of the majority class instances which is more representative, thus discarding useful information. This can be illustrated as follows:



- Here the green line is the ideal decision boundary we would like to have, and blue is the actual result. On the left side is the result of just applying a general machine learning algorithm without using undersampling. On the right, we undersampled the negative class but removed some informative negative class, and caused the blue decision boundary to be slanted, causing some negative class to be classified as positive class wrongly.

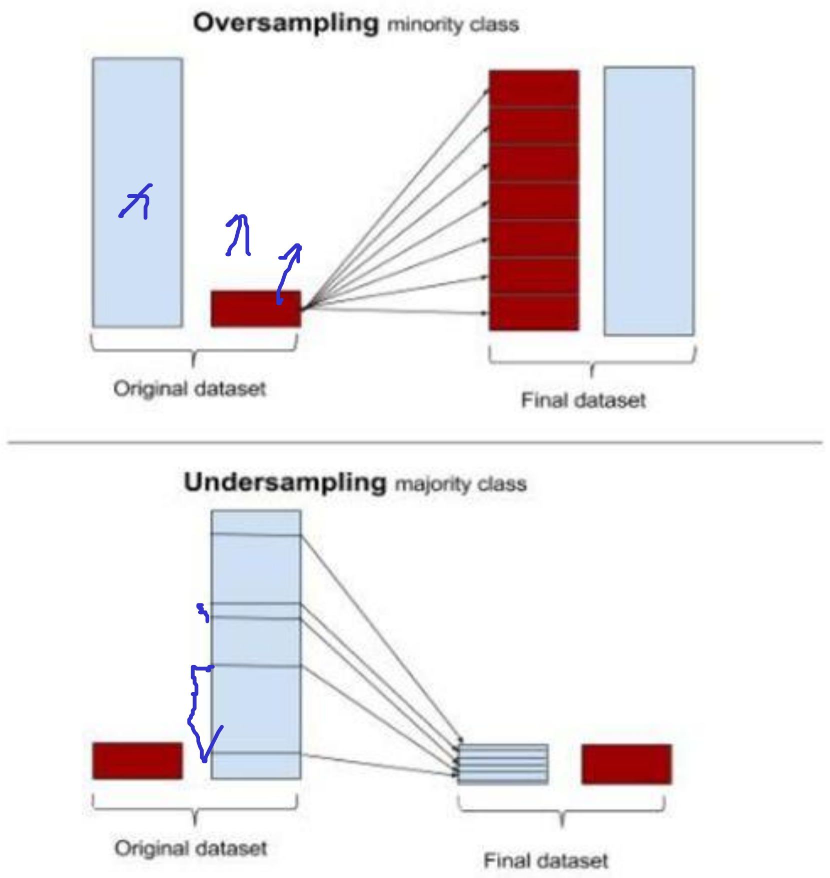


Sampling: Rebalancing the dataset

Imbalanced Data

Under-sampling

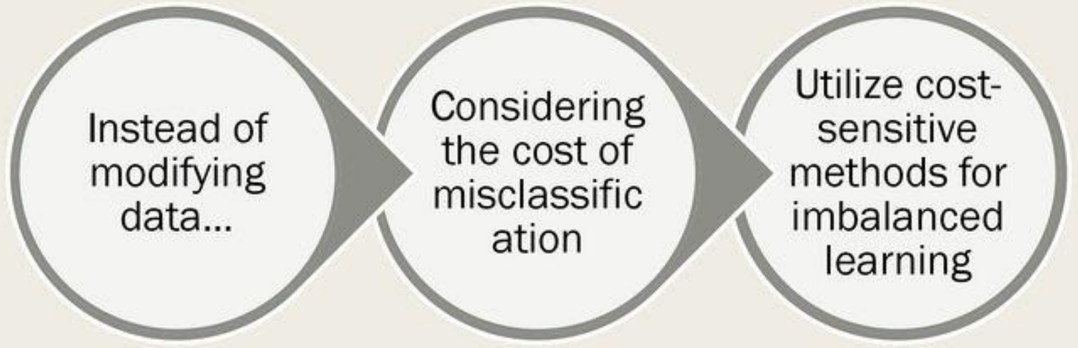
Over-sampling


Oversampling minority class
Original dataset
Final dataset
Undersampling majority class
Original dataset
Final dataset

methods require a performance measure to be specified a priori before learning.

- An alternative is to use a so-called threshold-moving method that a posteriori changes the decision threshold of a model to counteract the imbalance, thus has a potential to adapt to the performance measure of interest.
- Surprisingly, little attention has been paid to the potential of combining bagging ensemble with threshold-moving.
- our method preserves the natural class distribution of the data resulting in well calibrated posterior probabilities.

Cost-Sensitive Methods


Instead of modifying data…
Considering the cost of misclassific ation
Utilize cost-sensitive methods for imbalanced learning

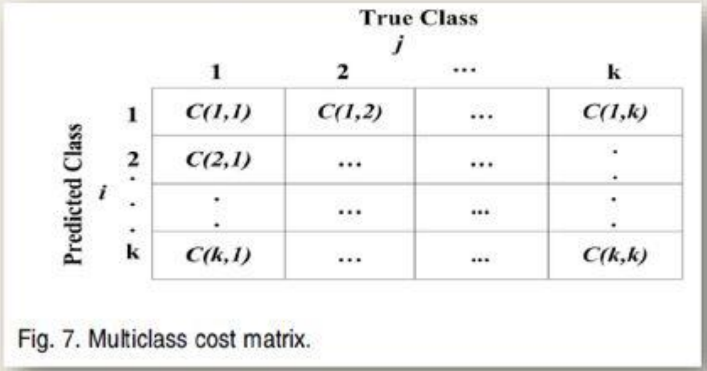
Cost-Sensitive Learning Framework

- Define the cost of misclassifying a majority to a minority as $C(Min, Maj)$
- Typically $C(Maj, Min) > C(Min, Maj)$
- Minimize the overall cost - usually the *Bayes conditional risk* - on the training data set

$$R(i|x) = \sum_j P(j|x)C(i,j)$$

Bayes

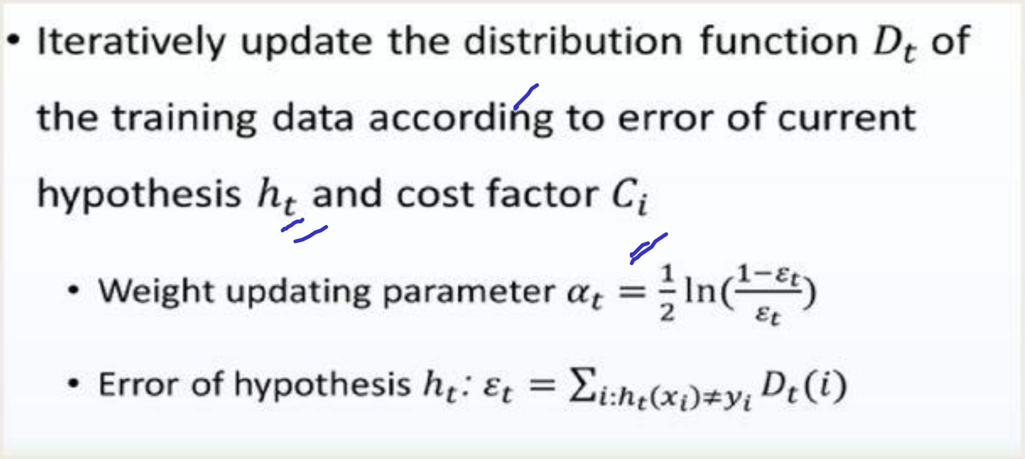
Bayesian Statistics



		True Class *j*			
		1	2	...	k
Predicted Class *i*	1	*C(1,1)*	*C(1,2)*	...	*C(1,k)*
	2	*C(2,1)*	...	...	. .
	. . .	. .	...	...	. .
	k	*C(k,1)*	...	...	*C(k,k)*

Fig. 7. Multiclass cost matrix.

Cost-Sensitive Dataspace Weighting with Adaptive Boosting



- Iteratively update the distribution function D_t of the training data according to error of current hypothesis h_t and cost factor C_i
  - Weight updating parameter $\alpha_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$
  - Error of hypothesis h_t: $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$

hypothesis

Cost-Sensitive Dataspace Weighting with Adaptive Boosting

- Given D_t, h_t, C_i, α_t, and ε_t

1. AdaC1: $D_{t+1}(i) = \frac{D_t(i)\, exp(-\alpha_t C_i h_t(x_i) y_i)}{Z_t}$
2. AdaC2: $D_{t+1}(i) = \frac{C_i D_t(i)\, exp(-\alpha_t h_t(x_i) y_i)}{Z_t}$
3. AdaC3: $D_{t+1}(i) = \frac{C_i D_t(i)\, exp(-\alpha_t C_i h_t(x_i) y_i)}{Z_t}$
4. AdaCost: $D_{t+1}(i) = \frac{D_t(i)\, exp(-\alpha_t h_t(x_i) y_i \beta_i)}{Z_t}$,

   $\beta_i = \beta(sign(y_i, h_t(x_i)), C_i)$