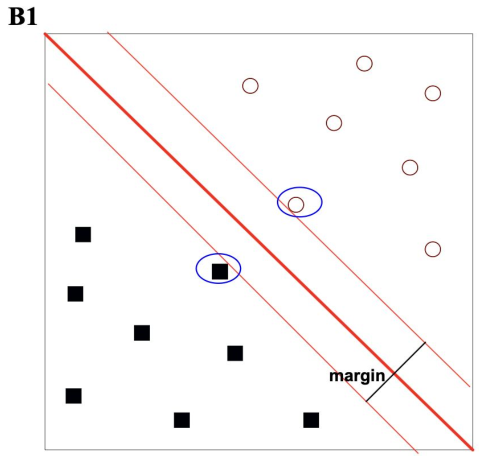
Support Vector Machine (SVM) & Ensembles

Exercise 1

Q1. The problem of finding a decision boundary in SVM can be formulated as an optimisation problem using Lagrange multipliers. What is the goal?

a) To maximize the margin of the decision boundary.
b) To minimize the margin of the decision boundary.

- **A decision boundary B1:**



Exercise 1

Q2. After SVM learning, each Lagrange multiplier *lambda_i* takes either zero or non-zero value. Which one is correct:

[✓ a) A non-zero *lambda_i* indicates that example *i* is a support vector.]
b) A zero *lambda_i* indicates that example *i* is a support vector.
c) A non-zero *lambda_i* indicates that the learning has not yet converged to a global minimum.
d) A zero *lambda_i* indicates that the learning process has identified support for example *i*.

Exercise 1

Q3. In linear SVM, during training we compute dot products between:

a) training vectors

b) training and testing vectors

c) support vectors

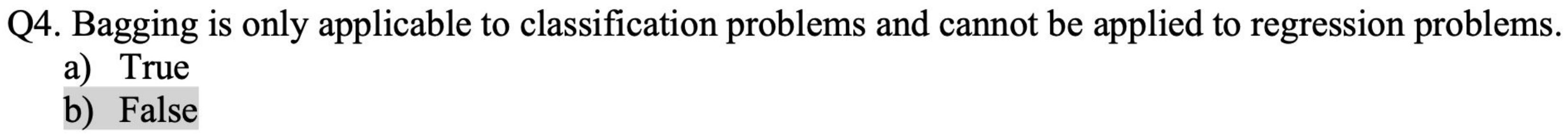
d) support vectors and Lagrange multiplayers

Explanation: During training, we compute dot products between pairs of training vectors; during testing – between the testing example and the support vectors. Note that <u>d) can be immediately ruled out as it is a product of a vector and coefficient; dot product is a product of 2 vectors.</u>

$$\max \mathbf{w}(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{N} \lambda_i \lambda_j \underbrace{y_i y_j}_{\text{Target value (class value) of the training vectors}} \overbrace{\mathbf{x}_i \cdot \mathbf{x}_j}^{\text{Dot product of pairs of training vectors}}$$

$$subject\ to\ \lambda_i \geq 0, \sum_{i=1}^{N} \lambda_i y_i = 0$$

Exercise 1



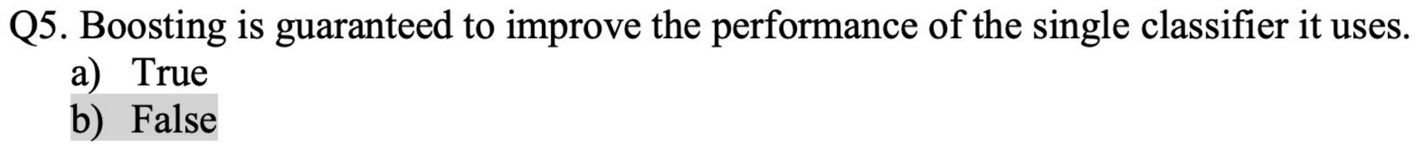
Q4. Bagging is only applicable to classification problems and cannot be applied to regression problems.

a) True

[✓ b) False]

Explanation: **Bagging can be applied to both classification and regression problems. In regression problems, the individual predictions will be averaged.**

Exercise 1



Q5. Boosting is guaranteed to improve the performance of the single classifier it uses.

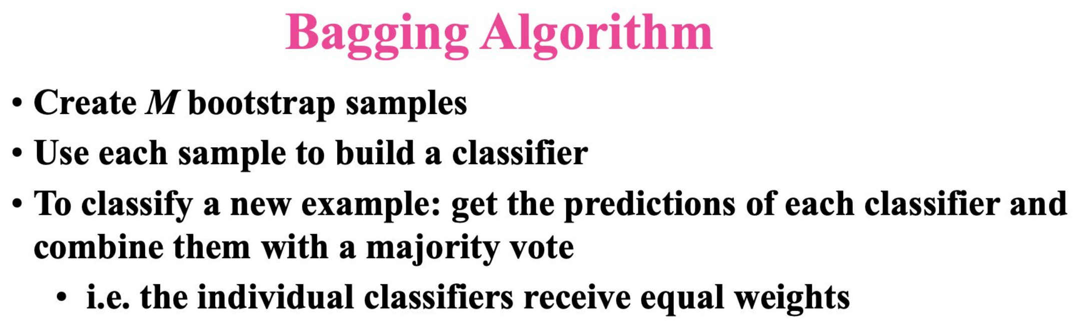
a) True

[✓ b) False]

Explanation: There is no guarantee that an ensemble of classifiers (including Boosting) will produce better accuracy than the single classifier it uses. In practice, ensembles often perform better but there is no guarantee.

Bagging

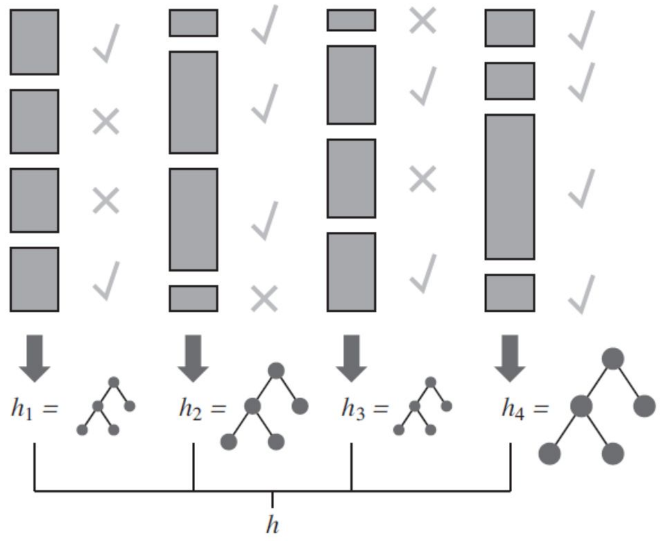
- Bagging stands for **“bootstrap aggregation”**
  - Bootstrap is “sample with replacement”



Bagging Algorithm

- **Create *M* bootstrap samples**
- **Use each sample to build a classifier**
- **To classify a new example: get the predictions of each classifier and combine them with a majority vote**
  - **i.e. the individual classifiers receive equal weights**

Boosting



- 1 rectangle corresponds to 1 example
- The height of the rectangle corresponds to the weight of the example
- √ and X show how the example was classified by the current hypothesis (classifier)
- The size of the DT corresponds to the weight of that hypothesis in the final ensemble

Exercise 2

Exercise 2

a) What are the similarities and differences between Bagging and Boosting?

Exercise 2

a) What are the similarities and differences between Bagging and Boosting? (see Lecture Slide 31)

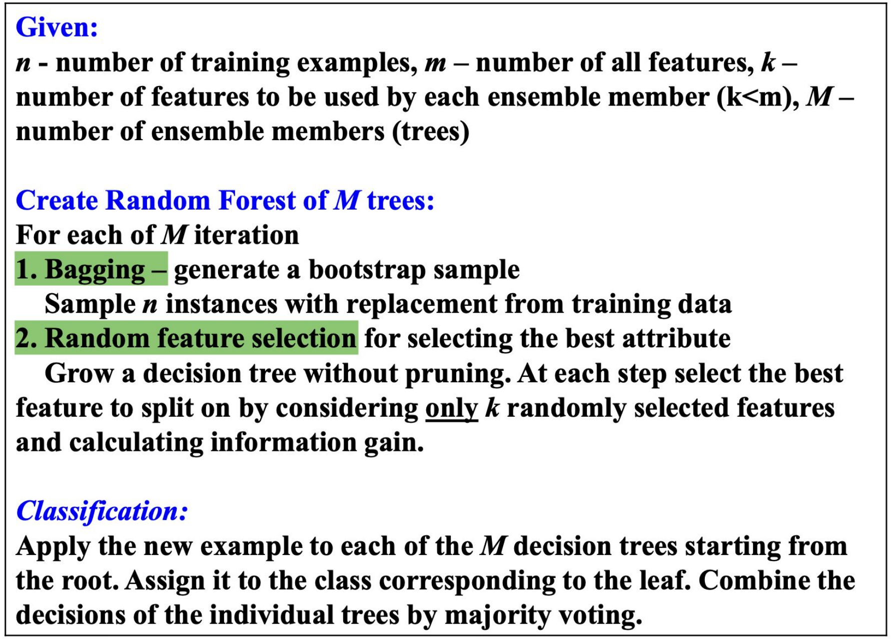
- **Similarities**
  - **Use voting (for classification) and averaging (for prediction) to combine the outputs of the individual learners**
  - **Combine classifiers of the same type, e.g. DTs**
- **Differences**
  - **In bagging the ensemble members are built separately; in boosting they are built iteratively – the new ensemble members are influenced by the performance of the previous ones**
    - **A new ensemble member is encouraged to become an expert for the examples classified incorrectly by the previous ensemble member**
    - **Intuitive justification: ensemble members should be experts that complement each other**
  - **Combing the opinions of the individual ensemble members:**
    - **Bagging – equal weighting (all experts are equally influential)**
    - **Boosting – weighed based on performance (i.e. some experts are more influential)**

Exercise 2

b) What are the 2 main ideas that are combined in Random Forest?

Exercise 2

b) What are the 2 main ideas that are combined in Random Forest?



Given:
n **- number of training examples,** ***m*** **– number of all features,** ***k*** **– number of features to be used by each ensemble member (k<m),** ***M*** **– number of ensemble members (trees)**

Create Random Forest of ***M*** **trees:**
For each of ***M*** **iteration**
1. Bagging – generate a bootstrap sample
Sample ***n*** **instances with replacement from training data**
2. Random feature selection for selecting the best attribute
Grow a decision tree without pruning. At each step select the best feature to split on by considering only ***k*** **randomly selected features and calculating information gain.**

Classification:
Apply the new example to each of the ***M*** **decision trees starting from the root. Assign it to the class corresponding to the leaf. Combine the decisions of the individual trees by majority voting.**

Exercise 3

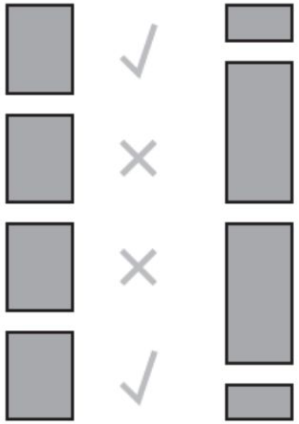
Adaboost

***Exercise 3* Boosting**

Continue the example from slides 24-25. The current probabilities of the training examples to be used by AdaBoost are given below:

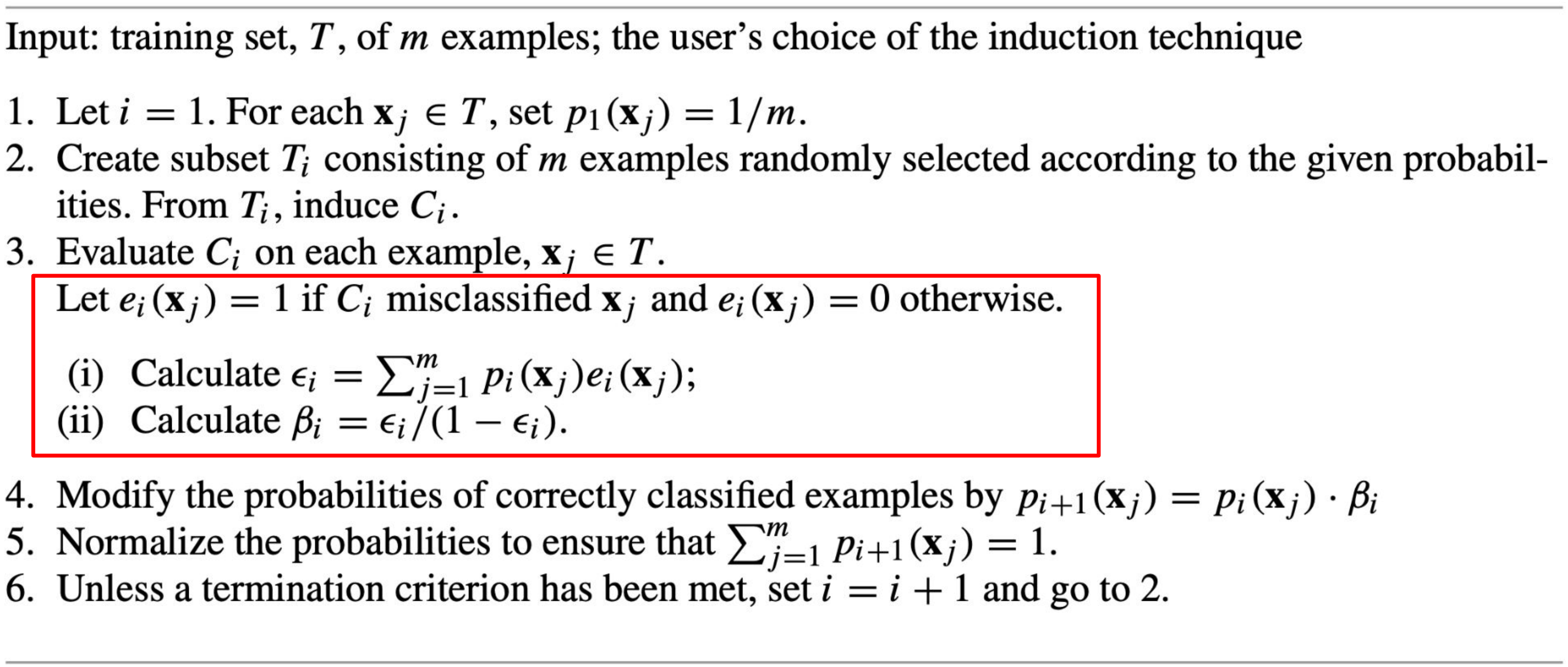
$p(x_1)$	$p(x_2)$	$p(x_3)$	$p(x_4)$	$p(x_5)$	$p(x_6)$	$p(x_7)$	$p(x_8)$	$p(x_9)$	$p(x_{10})$
0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.17	0.17	0.17

From these, a training set T_2 has been created and using T_2 a classifier C_2 has been generated. Suppose that C_2 misclassifies examples x_2 and x_9. Show how the probabilities of all training examples are recalculated and normalized.



Pseudo-code of Adaboost

From Miroslav Kubat, An Introduction to ML



```
Input: training set, $T$, of $m$ examples; the user's choice of the induction technique

1. Let $i = 1$. For each $\mathbf{x}_j \in T$, set $p_1(\mathbf{x}_j) = 1/m$.
2. Create subset $T_i$ consisting of $m$ examples randomly selected according to the given probabil-
   ities. From $T_i$, induce $C_i$.
3. Evaluate $C_i$ on each example, $\mathbf{x}_j \in T$.
   Let $e_i(\mathbf{x}_j) = 1$ if $C_i$ misclassified $\mathbf{x}_j$ and $e_i(\mathbf{x}_j) = 0$ otherwise.

    (i) Calculate $\epsilon_i = \sum_{j=1}^{m} p_i(\mathbf{x}_j) e_i(\mathbf{x}_j)$;
   (ii) Calculate $\beta_i = \epsilon_i/(1 - \epsilon_i)$.

4. Modify the probabilities of correctly classified examples by $p_{i+1}(\mathbf{x}_j) = p_i(\mathbf{x}_j) \cdot \beta_i$
5. Normalize the probabilities to ensure that $\sum_{j=1}^{m} p_{i+1}(\mathbf{x}_j) = 1$.
6. Unless a termination criterion has been met, set $i = i + 1$ and go to 2.
```

***Exercise 3* Boosting**

Continue the example from slides 24-25. The current probabilities of the training examples to be used by AdaBoost are given below:

	$p(x_1)$	$p(x_2)$	$p(x_3)$	$p(x_4)$	$p(x_5)$	$p(x_6)$	$p(x_7)$	$p(x_8)$	$p(x_9)$	$p(x_{10})$
	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.17	0.17	0.17
$e_2(x_j)$	0	1	0	0	0	0	0	0	1	0

From these, a training set T_2 has been created and using T_2 a classifier C_2 has been generated. Suppose that C_2 misclassifies examples x_2 and x_9. Show how the probabilities of all training examples are recalculated and normalized.

Let $e_i(\mathbf{x}_j) = 1$ if C_i misclassified $\mathbf{x}_j$ and $e_i(\mathbf{x}_j) = 0$ otherwise.

(i) Calculate $\epsilon_i = \sum_{j=1}^{m} p_i(\mathbf{x}_j)e_i(\mathbf{x}_j)$;
(ii) Calculate $\beta_i = \epsilon_i/(1-\epsilon_i)$.

Exercise 3 Boosting

Continue the example from slides 24-25. The current probabilities of the training examples to be used by AdaBoost are given below:

	$p(x_1)$	$p(x_2)$	$p(x_3)$	$p(x_4)$	$p(x_5)$	$p(x_6)$	$p(x_7)$	$p(x_8)$	$p(x_9)$	$p(x_{10})$
	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.17	0.17	0.17
$e_2(x_j)$	0	1	0	0	0	0	0	0	1	0

From these, a training set T_2 has been created and using T_2 a classifier C_2 has been generated. Suppose that C_2 misclassifies examples x_2 and x_9. Show how the probabilities of all training examples are recalculated and normalized.

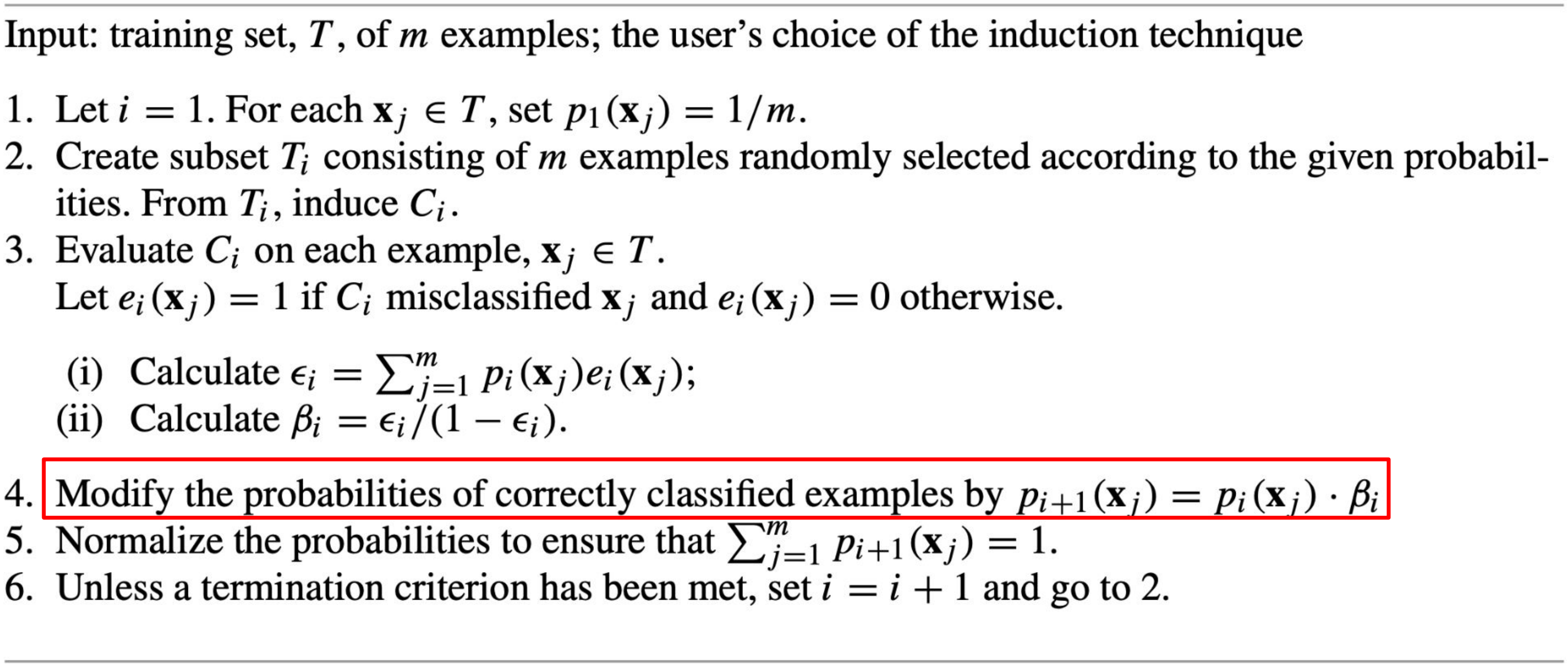
Let $e_i(\mathbf{x}_j) = 1$ if C_i misclassified $\mathbf{x}_j$ and $e_i(\mathbf{x}_j) = 0$ otherwise.

(i) Calculate $\epsilon_i = \sum_{j=1}^{m} p_i(\mathbf{x}_j)e_i(\mathbf{x}_j)$;
(ii) Calculate $\beta_i = \epsilon_i/(1 - \epsilon_i)$.

The weighed error of the classifier C_2 will be: $\epsilon_2=0.07*1+0.17*1=0.24$
The term β for the probability modification will be: $\beta_2=\epsilon_2/(1-\epsilon_2)=0.24/0.76=0.32$

Pseudo-code of Adaboost

From Miroslav Kubat, An Introduction to ML



```
Input: training set, $T$, of $m$ examples; the user's choice of the induction technique

1. Let $i = 1$. For each $\mathbf{x}_j \in T$, set $p_1(\mathbf{x}_j) = 1/m$.
2. Create subset $T_i$ consisting of $m$ examples randomly selected according to the given probabil-
   ities. From $T_i$, induce $C_i$.
3. Evaluate $C_i$ on each example, $\mathbf{x}_j \in T$.
   Let $e_i(\mathbf{x}_j) = 1$ if $C_i$ misclassified $\mathbf{x}_j$ and $e_i(\mathbf{x}_j) = 0$ otherwise.

    (i)  Calculate $\epsilon_i = \sum_{j=1}^{m} p_i(\mathbf{x}_j)e_i(\mathbf{x}_j)$;
    (ii) Calculate $\beta_i = \epsilon_i/(1 - \epsilon_i)$.

4. Modify the probabilities of correctly classified examples by $p_{i+1}(\mathbf{x}_j) = p_i(\mathbf{x}_j) \cdot \beta_i$
5. Normalize the probabilities to ensure that $\sum_{j=1}^{m} p_{i+1}(\mathbf{x}_j) = 1$.
6. Unless a termination criterion has been met, set $i = i + 1$ and go to 2.
```

***Exercise 3* Boosting**

Continue the example from slides 24-25. The current probabilities of the training examples to be used by AdaBoost are given below:

$p(x_1)$	$p(x_2)$	$p(x_3)$	$p(x_4)$	$p(x_5)$	$p(x_6)$	$p(x_7)$	$p(x_8)$	$p(x_9)$	$p(x_{10})$
0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.17	0.17	0.17

The weighed error of the classifier C_2 will be: $\epsilon_2=0.07*1+0.17*1=0.24$
The term β for the probability modification will be: $\beta_2=\epsilon_2/(1-\epsilon_2)=0.24/0.76=0.32$

Modify the probabilities of correctly classified examples by $p_{i+1}(\mathbf{x}_j) = p_i(\mathbf{x}_j) \cdot \beta_i$

The new probabilities are:

$p(x_1)$	$p(x_2)$	$p(x_3)$	$p(x_4)$	$p(x_5)$	$p(x_6)$	$p(x_7)$	$p(x_8)$	$p(x_9)$	$p(x_{10})$
0.07*0.32	0.07	0.07*0.32	0.07*0.32	0.07*0.32	0.07*0.32	0.07*0.32	0.17*0.32	0.17	0.17*.32

<u>Exercise 3</u> Boosting

Continue the example from slides 24-25. The current probabilities of the training examples to be used by AdaBoost are given below:

$p(x_1)$	$p(x_2)$	$p(x_3)$	$p(x_4)$	$p(x_5)$	$p(x_6)$	$p(x_7)$	$p(x_8)$	$p(x_9)$	$p(x_{10})$
0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.17	0.17	0.17

The weighed error of the classifier C_2 will be: ϵ_2=0.07*1+0.17*1=0.24

The term β for the probability modification will be: $\beta_2=\epsilon_2/(1-\epsilon_2)$=0.24/0.76=0.32

$\boxed{\text{Modify the probabilities of correctly classified examples by } p_{i+1}(\mathbf{x}_j) = p_i(\mathbf{x}_j) \cdot \beta_i}$

The new probabilities are:

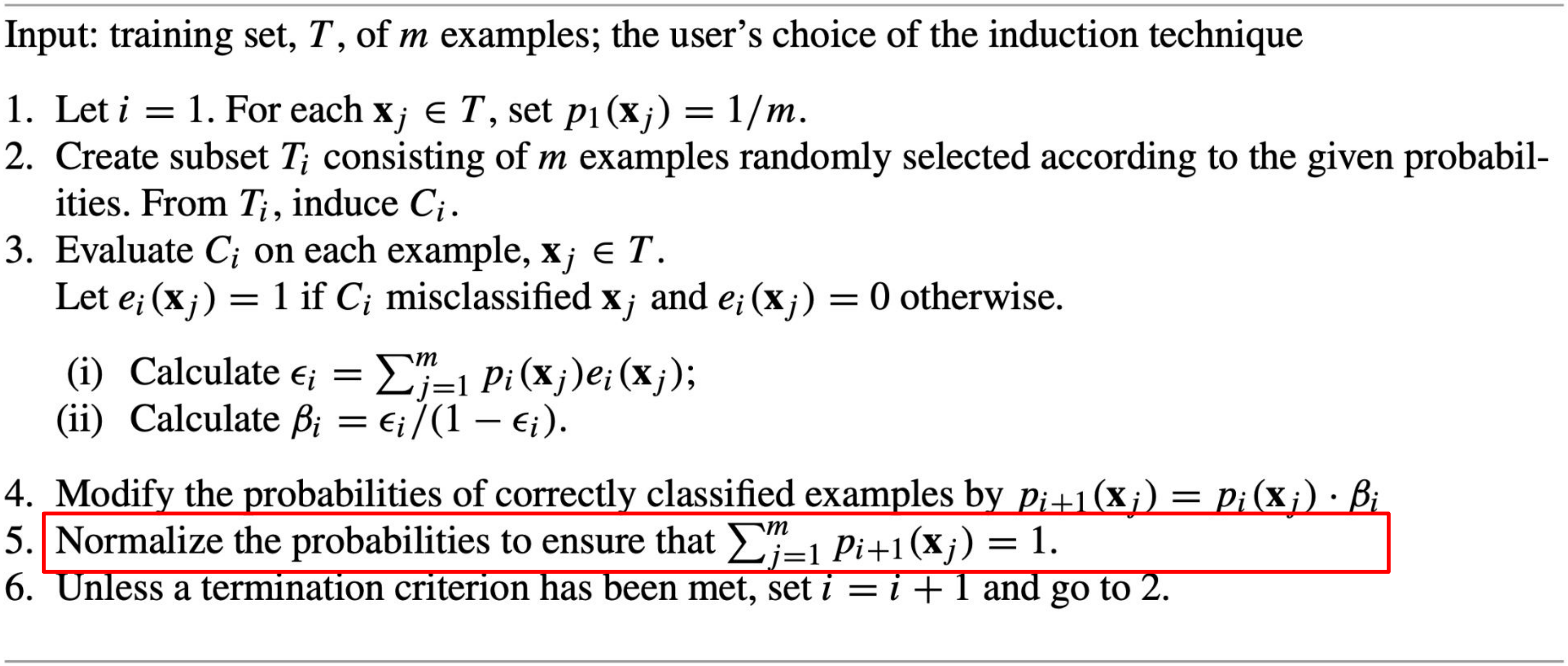
$p(x_1)$	$p(x_2)$	$p(x_3)$	$p(x_4)$	$p(x_5)$	$p(x_6)$	$p(x_7)$	$p(x_8)$	$p(x_9)$	$p(x_{10})$
0.07*0.32	0.07	0.07*0.32	0.07*0.32	0.07*0.32	0.07*0.32	0.07*0.32	0.17*0.32	0.17	0.17*.32

$\downarrow$

$p(x_1)$	$p(x_2)$	$p(x_3)$	$p(x_4)$	$p(x_5)$	$p(x_6)$	$p(x_7)$	$p(x_8)$	$p(x_9)$	$p(x_{10})$
0.022	0.07	0.022	0.022	0.022	0.022	0.022	0.054	0.17	0.054

Pseudo-code of Adaboost

From Miroslav Kubat, An Introduction to ML



```
Input: training set, $T$, of $m$ examples; the user's choice of the induction technique

1. Let $i = 1$. For each $\mathbf{x}_j \in T$, set $p_1(\mathbf{x}_j) = 1/m$.
2. Create subset $T_i$ consisting of $m$ examples randomly selected according to the given probabil-
   ities. From $T_i$, induce $C_i$.
3. Evaluate $C_i$ on each example, $\mathbf{x}_j \in T$.
   Let $e_i(\mathbf{x}_j) = 1$ if $C_i$ misclassified $\mathbf{x}_j$ and $e_i(\mathbf{x}_j) = 0$ otherwise.

    (i)  Calculate $\epsilon_i = \sum_{j=1}^{m} p_i(\mathbf{x}_j)e_i(\mathbf{x}_j)$;
   (ii)  Calculate $\beta_i = \epsilon_i/(1 - \epsilon_i)$.

4. Modify the probabilities of correctly classified examples by $p_{i+1}(\mathbf{x}_j) = p_i(\mathbf{x}_j) \cdot \beta_i$
5. Normalize the probabilities to ensure that $\sum_{j=1}^{m} p_{i+1}(\mathbf{x}_j) = 1$.
6. Unless a termination criterion has been met, set $i = i + 1$ and go to 2.
```

$p(x_1)$	$p(x_2)$	$p(x_3)$	$p(x_4)$	$p(x_5)$	$p(x_6)$	$p(x_7)$	$p(x_8)$	$p(x_9)$	$p(x_{10})$
0.022	0.07	0.022	0.022	0.022	0.022	0.022	0.054	0.17	0.054

Normalization:
Sum of all probabilities: 0.022*6+0.054*2+0.07+0.17=0.132+0.108+0.24=0.48

Normalized probabilities:

$p(x_1)$	$p(x_2)$	$p(x_3)$	$p(x_4)$	$p(x_5)$	$p(x_6)$	$p(x_7)$	$p(x_8)$	$p(x_9)$	$p(x_{10})$
0.022/0.48	0.07/0.48	0.022/0.48	0.022/0.48	0.022/0.48	0.022/0.48	0.022/0.48	0.054/0.48	0.17/0.48	0.054/0.48

$p(x_1)$	$p(x_2)$	$p(x_3)$	$p(x_4)$	$p(x_5)$	$p(x_6)$	$p(x_7)$	$p(x_8)$	$p(x_9)$	$p(x_{10})$
0.045	0.146	0.045	0.045	0.045	0.045	0.045	0.113	0.354	0.113

We can see that the weights of the misclassified examples (x_2 and x_9) have increased while the weights of the correctly classified examples (the remaining 8 examples) have decreased. In the next iteration x_2 and x_9 will have a higher chance to be selected for the new training set, i.e. the next classifier will focus on learning to classify these more difficult examples.

Exercise 4 – Weka

1. Load the diabetes data (diabetes.arff) or another binary classification data. The two classes is the diabetes data are "tested_positive" and "tested_negative".

Choose 10-fold cross validation. Run the SVM (SMO located under "functions"). This implementation of SVM uses John Platt's sequential minimal optimization algorithm to solve the optimization problem (i.e. to train SVM classifier) and shows the computed decision boundary.

Examine the output.
a) What type of SVM is used – linear or non-linear? What type of kernel function is used? Where is the equation of the decision boundary?

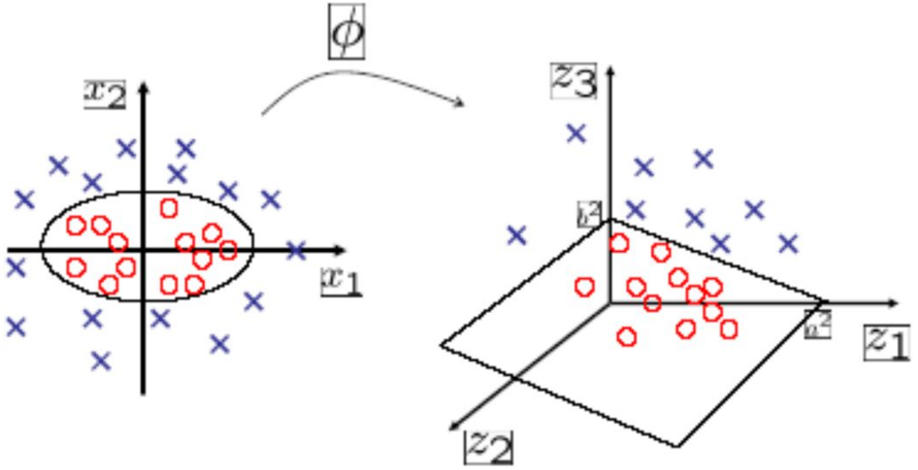
Exercise 4 – Weka

1. Load the diabetes data (diabetes.arff) or another binary classification data. The two classes is the diabetes data are “tested_positive” and “tested_negative”.

Choose 10-fold cross validation. Run the SVM (SMO located under “functions”). This implementation of SVM uses John Platt's sequential minimal optimization algorithm to solve the optimization problem (i.e. to train SVM classifier) and shows the computed decision boundary.

b) Let’s try non-linear SVM. Edit the properties of SMO (by clicking on “SMO”). Keep the same type of kernel (Poly Kernel, it is a general type of kernel and we can use it to create both linear and no-linear SVMs) but change the exponent from 1 to 2 (by clicking on “Poly kernel”). Now the kernel function is nonlinear: Poly Kernel: K(x,y) = <x,y>^2.0. Run the classifier, observe the new equation of the decision boundary and the new accuracy. Compare the accuracy of the linear and non-linear SVM, which one is better? Experiment with other types of kernels (e.g. other polynomial, by increasing and exponent, or RBF).

- **Non-linearly separable data $\mathbf{x} = (x_1, x_2)$ in the original space and linearly separable in the new space**



$$\phi : (x_1, x_2) \longrightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

transformation from old to new space

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \longrightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

decision boundaries in old and new space

original space (2-dim):

(x_1, x_2)

new space (3-dim):

$(x_1^2, \sqrt{2}x_1x_2, x_2^2)$

Exercise 5 – Weka

Load the iris data (iris.arff). Choose 10-fold cross validation and run the SVM classifier.

a) Examine the output. Not 1 but 3 SVM are generated. Do you know why?

Exercise 5 – Weka

Load the iris data (iris.arff). Choose 10-fold cross validation and run the SVM classifier.

a) Examine the output. Not 1 but 3 SVM are generated. Do you know why?

Answer:
This is because the task is a multi-class problem (3-class, there are 3 types of irises). SVM handles multi-class problems by transforming them into several binary problems. In general, there are 2 methods to do this: 1-versus-rest and one-versus-one. In the first case, k classifiers are constructed, where k is the number of classes; each of them learns a hyperplane to separate 1 class versus all the remaining classes. A new example is assigned to the class for which the distance to the margin is maximal. In the second case, k(k-1)/2 classifiers (i.e. hyperplanes) are constructed, 1 class versus another class. To classify a new example, a decision function using some ad-hoc voting scheme is used.

Exercise 5 – Weka

Load the iris data (iris.arff). Choose 10-fold cross validation and run the SVM classifier.

a) Examine the output. Not 1 but 3 SVM are generated. Do you know why?

Answer:
This is because the task is a multi-class problem (3-class, there are 3 types of irises). SVM handles multi-class problems by transforming them into several binary problems. In general, there are 2 methods to do this: 1-versus-rest and one-versus-one. In the first case, k classifiers are constructed, where k is the number of classes; each of them learns a hyperplane to separate 1 class versus all the remaining classes. A new example is assigned to the class for which the distance to the margin is maximal. In the second case, k(k-1)/2 classifiers (i.e. hyperplanes) are constructed, 1 class versus another class. To classify a new example, a decision function using some ad-hoc voting scheme is used.

b) Examining again the output, which of the 2 methods does Weka use?

Exercise 5 – Weka

Load the iris data (iris.arff). Choose 10-fold cross validation and run the SVM classifier.

b) Examining again the output, which of the 2 methods does Weka use?

Answer: The second one, 1 class vs another:

```
Classifier for classes: Iris-setosa, Iris-versicolor …
Classifier for classes: Iris-setosa, Iris-versicolor …
Classifier for classes: Iris-versicolor, Iris-virginica …
```

The voting scheme to combine these 3 classifiers is not clear from the Weka’s output.

Exercise 6 – Weka

1. Load iris data (iris.arff). Choose 10-fold cross validation. Run the decision trees (J48) single classifiers.

2. Run bagging of decision trees (J48). It is under “Meta”. The default base classifier is REPTree, you need to change it to j48. Compare the performance with the single DT classifier from 1).

3. Run boosting (AdaBoost) of decision trees (J48). It is also under “Meta” and you need to change the default base classifier. Compare performance with the single classifiers from 1) and also with the bagged classifier from 2).

4. Run Random Forest of decision trees (J48). It’s under “Trees” and you need to change the default base classifier. Compare performance with the previous classifiers.

5. Use AdaBoost with OneR as a base classifier. Experiment with different number of hypotheses M. What happens with the accuracy on the training data as M increases? What about the accuracy on test data? Is this consistent with the AdaBoost theorem?

Exercise 6 – Results

Classifier	Accuracy
Single Decision Tree (J48)	96%
Bagging of Decision Trees (J48)	94.6667%
AdaBoost of Decision Trees (J48)	93.3333%
Random Forest of Random Trees (J48 not available!)	95.3333%
AdaBoost with OneR	92.6667%