Instructions
- Total time: 3 hours.
- All written or printed material is allowed.
- You should answer all questions.
- Explain your steps.
- Good luck!

1. A modified version of the Perceptron algorithm is given in Figure 1.

\begin{figure}[h]
\centering
\begin{align*}
\text{INIT:} \quad & \text{training set } S = \{(x_i, y_i)\}_{i=1}^m \\
\text{INITIALIZE:} \quad & w = 0 \\
\text{LOOP:} \quad & \\
& \text{• choose example } (x_i, y_i) \text{ uniformly at random from } S \\
& \text{• let } \ell(w, x_i, y_i) = -1 - y_i w \cdot x_i \\
& \text{• if } \ell(w, x_i, y_i) > 0: \\
& \quad \text{update: } w = w + y_i x_i
\end{align*}
\caption{Large margin Perceptron}
\end{figure}

(a) Assume that \( \|x_i\| \leq R \) for all \( i \) and that there exists a vector \( u \), \( \|u\| = 1 \) such that \( y_i(u \cdot x_i) \geq \gamma \) for all \( i \). Derive an upper bound for the number of mistakes made by this new Perceptron algorithm. [15 pt]

(b) How the new bound you derived in (a) compares to the standard Perceptron bound and why? [4 pt]

(c) Write the new Perceptron algorithm in its kernel form? That is, instead of \( w \) use its implicit definition:

\[ w \cdot x = \sum_{j=1}^{i} \alpha_j x_j \cdot x \]

What are \( \alpha_j \) in this algorithm? [8 pt]

(d) What is the advantage of the negative margin Perceptron algorithm when working with kernels and huge amount of training data [7 pt]
2. LinkedIn works on a new algorithm for ranking candidates for a set of jobs $Q$. For each job $q \in Q$ there is a set of $N$ candidates. Each candidate has a set of features $\mathbf{x} \in \mathbb{R}^d$ describing his/her qualifications and a label $y \in \{-1, +1\}$ indicating if he/she is relevant for this job.

(a) Given a training set $S$ of $m$ jobs with $N$ candidates for each job,

$$S = \{(q^i, (x_1^i, \ldots, x_N^i), y^i)\}_{i=1}^m,$$

propose an algorithm for ranking the candidates. [28 pt]

(b) A new job $j_{\text{new}}$ is added to the set of available jobs $Q$. Do you need to retrain your algorithm? Explain. [5 pt]

3. Structured prediction. We would like to predict where is a triangle located in an image as in Figure 2. Each triangle is defined by a vector of three points $y = (y_1, y_2, y_3)$. The predicted triangle is denoted by $\hat{y} = (\hat{y}_1, \hat{y}_2, \hat{y}_3)$. The performance are measure by the intersection between the triangles $y$ and $\hat{y}$ divided by their union (called intersection-over-union), and denoted $\gamma(y, \hat{y})$.

(a) Propose an algorithm that predicts $\hat{y}$ and aims at maximizing $\gamma(y, \hat{y})$ given a training set of images and labeled triangles. [28 pt]

(b) Suggest a feature function for such a prediction. The feature should not be describe mathematically, just conceptually with words. [5 pt]