Exercises
Advanced Machine Learning
Fall 2019

Machine Learning Laboratory
Dept. of Computer Science, ETH Zürich
Prof. Joachim M. Buhmann
Web https://ml2.inf.ethz.ch/courses/aml/

Teaching assistant: **Clara Meister**
meistecl@inf.ethz.ch

Series 6. November 27, 2019 (Structural SVMs and Ensemble Methods)

Problem 1 (Structural SVMs):

In this exercise we derive the SSVM objective discussed in class from a probabilistic perspective. We use the same notations, i.e. $\{(\mathbf{y}_i, z_i)\}_{i=1}^n \subset \mathcal{Y} \times \mathbb{K}$ is our data set with *structured output labels*, $\Psi(z, \mathbf{y})$ is a *joint feature map*, and $\Delta(z_i, z_j)$ is the loss function.

1. We start from the following loss function for the set of parameters $\mathbf{w}$

$$\mathcal{R}(\mathbf{w}) = -\log p(\mathbf{w}) + \sum_{i=1}^{n} \log \Big[\sum_{z} \mathcal{L}(z, z_i) p(z \mid \mathbf{y}_i, \mathbf{w}) \Big],$$

   where $\mathcal{L}(\cdot, \cdot)$ is a label-specific loss function and $p(z \mid \mathbf{y}, \mathbf{w})$ is a *yet-to-be-specified* label-conditional. Why would $\mathcal{R}(\mathbf{w})$ be a reasonable objective to drive the learning of the parameters $\mathbf{w}$?

   Hint: What is $\mathcal{R}(\mathbf{w})$ when $\mathcal{L}(z, z_i) = 1 - [\![z = z_i]\!]$ (the 0-1 loss)?[1] Where have we seen it before? What does it correspond to for an arbitrary loss function?

2. Now assume that our probabilistic model is given by the following label-conditionals and parameters prior:

$$p(z \mid \mathbf{y}, \mathbf{w}) = \frac{\exp(\mathbf{w}^T \Psi(z, \mathbf{y}))}{Z(\mathbf{y}, \mathbf{w})}$$
$$p(\mathbf{w}) = \frac{\exp(-E(\mathbf{w}))}{Z}, \tag{1}$$

   where $Z(\mathbf{y}, \mathbf{w}) = \sum_z \exp(\mathbf{w}^T \Psi(z, \mathbf{y}))$ and $Z = \sum_{\mathbf{w}} \exp(-E(\mathbf{w}))$ are the so-called *partition functions* that normalize the distributions, respectively. Also, let $\Delta(z, z_i) = \log \mathcal{L}(z, z_i)$. Show that the objective becomes

$$\mathcal{R}(\mathbf{w}) = E(\mathbf{w}) + \sum_{i=1}^{n} \Big[-\log Z(\mathbf{y}_i, \mathbf{w}) + \log \sum_{z} \exp\big(\Delta(z, z_i) + \mathbf{w}^T \Psi(z, \mathbf{y}_i)\big) \Big].$$

   Is $\mathcal{R}(\mathbf{w})$ a convex function?

3. Show the following upper bound on the objective function:

$$\mathcal{R}(\mathbf{w}) \leq E(\mathbf{w}) + \sum_{i=1}^{n} \Big[\max_{z} \{\Delta(z, z_i) + \mathbf{w}^T \Psi(z, \mathbf{y}_i)\} - \mathbf{w}^T \Psi(\mathbf{y}_i, z_i) \Big] + \text{const.} =: \mathcal{UB}(\mathbf{w}).$$

   Is $\mathcal{UB}(\mathbf{w})$ a convex function?

   Hint: Use the following *"squeezing"* inequalities for the *log-sum-exp* function:

$$\max_{z \in \mathbb{K}} f(z) \leq \log \sum_{z \in \mathbb{K}} \exp\big(f(z)\big) \leq \max_{z \in \mathbb{K}} f(z) + \log |\mathbb{K}|, \quad \forall f : \mathbb{K} \to \mathbb{R}.$$

   Can you also prove them?

[1] We use the Iverson bracket $[\![S]\!]$ that evaluates to 1 if S is true and 0 otherwise.

4. What should $E(\mathbf{w})$ be such that minimizing $\mathcal{UB}(\mathbf{w})$ would give rise to the same optimization problem as the SSVM primal (see last exercise sheet)?

5. Recap the previous steps and ponder on this different way of reaching the same objective for learning Structural SVMs. Can you see any problem with optimizing $\mathcal{UB}(\mathbf{w})$ instead of $\mathcal{R}(\mathbf{w})$? Any advantage?

Problem 2 (Ensemble Methods):

1. State two essential diferences between bagging and boosting.

2. How can we use the AdaBoost algorithm to detect outliers?

3. What is a potential concern with bagging when training on a data set whose labels are highly imbalanced (i.e. one class has many more training points than another)? State one measure that can be taken to alleviate this.

Problem 3 (Bagging):

A political scientist wants to understand the predictors of household income. She has collected a set of 2000 features on 5000 observations. She wants to use random forests (bagging technique) to model feature interactions but her software implementation is too slow. She comes up with the following plan: she will first run the lasso with cross-validation and extract the predictors P with nonzero coefficients in the selected model. Then she will use just the set P in the random forest.

- Is this a reasonable method? State why or why not.

Problem 4 (Boosting):

Suppose you obtain a data set $(\mathbf{x_1}, y_1) \dots (\mathbf{x_n}, y_n)$ sampled i.i.d. from some distribution $P(\mathbf{X}, Y)$. You wish to learn a mapping $f : \mathcal{X} \to \mathcal{Y}$.

Similar to the AdaBoost algorithm, gradient boosting is a methodology to greedily approximate f using gradient descent based on the additive form

$$f_M(x) = \sum_{i=1}^{M} \beta_i h_i(\mathbf{x})$$

where h_i are weak learners.

The goal is to find an approximated function $\hat{f}$ of the function f^* that minimizes the expected value of a differentiable loss function $L(y, f)$ over the joint distribution of all $(\mathbf{x}, y)$ values.

$$f^*(\mathbf{x}) = \arg\min_{f(\mathbf{x})} \mathbb{E}_{y|\mathbf{x}}[L(y, f(\mathbf{x}))]$$

At an iteration m, the greedy approach attempts to find the update $\beta_m h_m(\mathbf{x})$ that minimizes the expected loss.

$$\beta_m, h_m = \arg\min_{\beta,h} \sum_{i=1}^{n} L(y_i, f_{m-1}(\mathbf{x_i}) + \beta h(\mathbf{x_i}))$$

and then,

$$f_m(\mathbf{x}) = f_{m-1} + \beta_m h_m(\mathbf{x}).$$

The hypothesis h_m is selected to be the "closest" to the negative gradient of the loss function.

The algorithm works as follows:

1. Initialize $\hat{f}_0(\mathbf{x}) = \arg\min_h \sum_{i=1}^{n} L(y_i, h(\mathbf{x_i}))$
2. For m = 1 to M:
   (a) Compute the negative gradient
   $$-g_m(\mathbf{x_i}) = -\left[\frac{\partial L(y_i, f(\mathbf{x_i}))}{\partial f(\mathbf{x_i})}\right]_{f=\hat{f}_{m-1}(\mathbf{x_i})}, i = 1 \dots n$$
   (b) Fit a function h_m to the negative gradient by least squares
   $$h_m = \arg\min_h \sum_{i=1}^{n} (-g_m(\mathbf{x_i}) - h(\mathbf{x_i}))^2$$
   (c) Find β_m to minimize the loss
   $$\beta_m = \arg\min_\beta \sum_{i=1}^{n} L(y_i, \hat{f}_{m-1}(\mathbf{x_i}) + \beta h_m(\mathbf{x_i}))$$
   (d) Update $\hat{f}$
   $$\hat{f}_m(\mathbf{x}) = \hat{f}_{m-1} + \beta_m h_m(\mathbf{x})$$
3. Output $\hat{f}_M$ for regression or the sign of $\hat{f}_M$ for classification

1. Based on this general procedure, write the precise steps required to approximate a function $\hat{f}$ using the squared error loss $L(y, f) = (y - f)^2/2$ to perform regression.
2. Analyze what the algorithm does with this loss function and explain it in a few sentences.