Machine Learning Worksheet 10

Programming Exercise: Linear Classification

In this exercise you will extend an implementation of logistic regression (optimized with gradient descent) to linear classification with the "hinge" and "soft zero-one" loss functions. You will try to understand the behavior of the different loss functions in the non- and linearly separable case, as well as analyze the impact of initialization and regularization.

To this end you are provided an IPython Notebook (homework-10.ipynb) with large parts of the necessary code. So you only have to write small parts of the code yourself. Of course, you may also use any other programming language and write your own code.

Set yourself a time limit for this exercise sheet and focus on understanding rather than writing perfect, extensive answers. Feel free to create helpful additional plots to accompany your answer.

You may either submit a "texed PDF" as always (do not forget to include your code) or submit the notebook in PDF format. To print to PDF, run:

```
ipython nbconvert homework-10.ipynb --to latex --post PDF
```

1 Hinge loss

The hinge loss is given as

$$\mathcal{L}(\mathbf{x}_i) = \max(0, 1 - y_i \tilde{z}_i),$$

where $y_i = \mathbf{x}_i^T \mathbf{w}$ is the model output and z_i the target variable ($w_0 = b$ and $x_{i,0} = 1$).

Note that in this case the computation uses class labels $\tilde{z} = 2z - 1 \in \{-1, 1\}$ instead of $z \in \{0, 1\}$.

For multiple samples $\mathbf{X}$ and respective outputs $\mathbf{y}$ and $\tilde{\mathbf{z}}$ the loss is $\mathcal{L}(\mathbf{X}) = \sum_i \mathcal{L}(\mathbf{x}_i)$.

Problem 1: Try to understand what the hinge loss does and explain it in a few of words:

Problem 2: Derive a sub-gradient of the hinge loss $\frac{\mathrm{d}\mathcal{L}(\mathbf{x}_i)}{\mathrm{d}\mathbf{w}}$. (You may assume that $w_0 = b$. Take 0 for the sub-gradient at non-differentiable points. Otherwise deriving a sub-gradient is equivalent to deriving the gradient.)

Problem 3: Implement the hinge loss.

2 Soft Zero-one loss

The soft zero-one loss is given as

$$\mathcal{L}(\mathbf{x}_i) = \left(\sigma(\beta y_i) - z_i\right)^2,$$

with $y_i = \mathbf{x}_i^T \mathbf{w}$ the model output and z_i the target variable.

Problem 4: Explain the soft zero-one loss in a few words.

Problem 5: Derive the gradient $\frac{\mathrm{d}\mathcal{L}(\mathbf{x}_i)}{\mathrm{d}\mathbf{w}}$.

Problem 6: Implement the soft zero-one loss.

3 Understanding

Problem 7: Create a linearly separable dataset and perform logistic regression

1. with very small initial weights and
2. with large initial weights.

Train multiple models for each scenario and describe what you observe: Do the models always converge to the same values? Do you understand why?

- Use NO regularization.
- Set momentum = 0.
- Keep your training set constant.

Problem 8: How does this change if you use L_2-regularization?

Problem 9: Now use the soft zero-one loss on a linearly separable dataset. Using no regularization and keeping other parameters constant, change the steepness parameter. What is the connection between steepness and the magnitude of the weight vector? What happens if the steepness is too large?

Problem 10: (optional) Look at the hinge loss. Create an overlapping data set. What happens at the end of training? (This effect will be more visible if you set the learning rate to high values.) Check to see if this also happens with the logistic loss and the soft zero-one loss (do not forget to set the steepness parameter back to sensible values).

Problem 11: Work with datasets that contain outliers. Set the parameters to values that make sense to you and compare the behavior of the three loss functions. Which one handles outliers best in your opinion?