Classification, Convexity and Neural Networks

CSC311 Summer 2025

University of Toronto

Classification Review

Classification Review

Classification is the task of predicting a discrete-valued target, given some input features. The target for a given input belongs to one of a fixed number of categories.
Types:

- Binary Classification: two categories.
- Multi-class Classification: $>$ two categories.

Multi-Class Classification

Targets:

- Targets belong to a discrete set: $\{1, 2, \ldots, K\}$.

One-hot encoding:

- One-hot vectors or encodings map target categories to unit vectors which are convenient representations.
- The k^{th} category is mapped to a vector with a 1 as the k^{th} element and 0 everywhere else.

$$(0, \ldots, 0, 1_k, 0, \ldots, 0) \in \mathbf{R}^K$$

Softmax Regression

We would like the algorithm to learn to output vectors that match the correct one-hot representation. Recall the **Softmax** function from lecture does this.

- It is the multi-class generalization of the Logistic function we used for binary prediction.
- Inputs to the the Softmax function are called **Logits**.
- The outputs can be interpreted as a probability distribution.

We'll see an example of this in the Colab.

Convexity

Here, we revisit the property of convexity and why we care about it from an optimization perspective.

Convex Sets

$$x_1, x_2 \in \mathcal{S},\ 0 \leq \lambda \leq 1 \implies \lambda x_1 + (1-\lambda)x_2 \in \mathcal{S}$$

Convex Functions

f is a convex function if for x_1, x_2 in its domain,

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$$

Why do we care about convexity?

- We saw in the Colab last week that for any convex function, local minima correspond to global minima.
- In general, gradient descent finds a critical point, which may be a local or global optima.
- However, given a convex objective, gradient descent is guaranteed to find the global optimum!

Convex Objective Functions

Common examples:

- Mean squared error
- Cross entropy loss

Metrics to Track Performance

We can use different metrics to track or evaluate the performance of our algorithm. A few examples:

- Classification accuracy: average number of data points classified correctly.
- Area under ROC curve (specific to Binary Classification)
- Confusion matrix: how frequently two classes are confused.

Recall from lecture the neuron-like processing units inspired by the human brain. Neural networks allow us to combine these to make large computations and perform optimization to learn mappings from data.
Training a neural network requires:

- Training dataset with input-target pairs.
- Objective function to measure the mismatch between targets and predicted outputs. Minimizing this function defines an optimization problem.
- Optimization solver, like gradient descent.
- Metrics to evaluate performance of the network.

Mitigating Overfitting

Large neural networks are prone to overfitting to the training dataset, or in other words, memorizing it. There are several techniques to mitigate this:

- Measure performance on validation data
- Early stopping
- L_2 regularization of the network weights