Neuron

Neuron-centric Programming Model

```
public class ThreeLayerANNExample {

  /**
   * User-defined sigmoid actiavation function
   */
  public static class Sigmoid extends ActivationFunction {
    @Override
    public double apply(double input) {
      return 1.0 / (1 + Math.exp(-input));
    }

    @Override
    public double applyDerivative(double input) {
      return input * (1 - input);
    }
  }

  /**
   * User-defined cost function
   */
  public static class CrossEntropy extends CostFunction {
    @Override
    public double apply(double target, double actual) {
      double adjustedTarget = (target == 0 ? 0.000001 : target);
      adjustedTarget = (target == 1.0 ? 0.999999 : target);
      double adjustedActual = (actual == 0 ? 0.000001 : actual);
      adjustedActual = (actual == 1 ? 0.999999 : actual);
      return -adjustedTarget * Math.log(adjustedActual) - (1 - adjustedTarget)
          * Math.log(1 - adjustedActual);
    }

    @Override
    public double derivative(double target, double actual) {
      double adjustedTarget = target;
      double adjustedActual = actual;
      if (adjustedActual == 1) {
        adjustedActual = 0.999;
      } else if (actual == 0) {
        adjustedActual = 0.001;
      }
      if (adjustedTarget == 1) {
        adjustedTarget = 0.999;
      } else if (adjustedTarget == 0) {
        adjustedTarget = 0.001;
      }
      return -adjustedTarget / adjustedActual + (1 - adjustedTarget)
          / (1 - adjustedActual);
    }
  }

  public static void main(String[] args) throws Exception {
    ANNJob ann = new ANNJob();

    // set learning rate and momentum weight
    ann.setLearningRate(0.1);
    ann.setMomentumWeight(0.1);

    // initialize the topology of the model
    // set the activation function and parallel degree.
    ann.addLayer(featureDimension, Sigmoid.class, numOfTasks);
    ann.addLayer(featureDimension, Sigmoid.class, numOfTasks);
    ann.addLayer(labelDimension, Sigmoid.class, numOfTasks);

    // set the cost function to evaluate the error
    ann.setCostFunction(CrossEntropy.class);
```

```
    ann.setMaxIteration(50);
    ann.setBatchSize(500);
    ann.setInputPath(path);
    ...

  }
}
```