推荐 wikipidia 的 Simple Linear Regression，或是 pattern recognition and machine learning 的第 3 章，如果只想了解这个算法，那就有些没必要了。

先看一下 classifyInstance，主要从这里到最后求得的是什么：

```
public double classifyInstance(Instance inst) throws Exception {

    if (m attribute == null) {
        return m intercept;
    } else {
        if (inst.isMissing(m attribute.index())) {
            throw new Exception(
                    "SimpleLinearRegression: No missing values!");
        }
        return m intercept + m slope * inst.value(m attribute.index());
    }
}
```

如果 m_attribte 为 null，就返回 m_intercept，intercept 的意思是截距，也就是如果这个属性没有值，就认为是 0，如果有值，那么就是 m_intercept + m_slope * value，就是一个线性函数。m_slope 是斜率。

buildClassifier 的代码非常简单：

先拷贝一点解释（wiki）：

Suppose there are *n* data points $\{y_i, x_i\}$, where $i = 1, 2, \ldots, n$. The goal is to find the equation of the straight line. (假设有 n 个点$\{y_i, x_i\}$，其中 $i = 1, 2, \ldots, n$.目标是找到一个直线方程)

$$y = \alpha + \beta x,$$

which would provide a “best” fit for the data points. Here the “best” will be understood as in the [least-squares]() approach: such a line that minimizes the sum of squared residuals of the linear regression model. In other words, numbers *α* and *β* solve the following minimization problem. (它可以最好地拟合数据，这里最好可以用 least-squares 方法来理解：即一条可以最小化线性回归模型的误差平方的线。换句话说，alpha 和 beta 用来最小化下面的问题)

$$\text{Find } \min_{\alpha,\beta} Q(\alpha,\beta), \text{ where } Q(\alpha,\beta) = \sum_{i=1}^{n} \hat{\varepsilon}_i^{\,2} = \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2$$

Using simple [calculus]() it can be shown that the values of *α* and *β* that minimize the objective function *Q* are (用简单的微积分推导可以得到最小化目标函数 Q 的值可以如下表示)

$$\begin{aligned}\hat{\beta} &= \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2} = \frac{\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i / n}{\sum_{i=1}^{n}(x_i^2) - (\sum_{i=1}^{n} x_i)^2 / n} \\ &= \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - \bar{x}^2} = \frac{\mathrm{Cov}[x,y]}{\mathrm{Var}[x]} = r_{xy}\frac{s_y}{s_x}, \\ \hat{\alpha} &= \bar{y} - \hat{\beta}\,\bar{x},\end{aligned}$$

Substituting the above expressions (代回到上面的方程):

$$y = \hat{\alpha} + \hat{\beta} x,$$

用到的就这么多，属性 wiki 上写的也有，自己看。Weka 中实现的算法是在属性中找一个最好的属性，最后用这个属性得到的截距和斜率做为结果。

```
for (int i = 0; i < insts.numAttributes(); i++) {
```

```
    if (i != insts.classIndex()) {
        m_attribute = insts.attribute(i);

        // Compute slope and intercept
        double xMean = insts.meanOrMode(i);
        double sumWeightedXDiffSquared = 0;
        double sumWeightedYDiffSquared = 0;
        m_slope = 0;
        for (int j = 0; j < insts.numInstances(); j++) {
            Instance inst = insts.instance(j);
            if (!inst.isMissing(i) && !inst.classIsMissing()) {
                double xDiff = inst.value(i) - xMean;
                double yDiff = inst.classValue() - yMean;
                double weightedXDiff = inst.weight() * xDiff;
                double weightedYDiff = inst.weight() * yDiff;
                m_slope += weightedXDiff * yDiff;
                sumWeightedXDiffSquared += weightedXDiff * xDiff;
                sumWeightedYDiffSquared += weightedYDiff * yDiff;
            }
        }

        // Skip attribute if not useful
        if (sumWeightedXDiffSquared == 0) {
            continue;
        }
        double numerator = m_slope;
        m_slope /= sumWeightedXDiffSquared;
        m_intercept = yMean - m_slope * xMean;

        // Compute sum of squared errors
        double msq = sumWeightedYDiffSquared - m_slope * numerator;

        // Check whether this is the best attribute
        if (msq < minMsq) {
            minMsq = msq;
            chosen = i;
            chosenSlope = m_slope;
            chosenIntercept = m_intercept;
        }
    }
}
```

这里带来的干扰就是 weight，直接把它看成是 1 就可以了，斜率 m_slope /= sumWeightedXDiffSquared 用到的就是上面的公式 beta hat 等式后第一个式子，而截距用的公式是有上面的是完全一样的。写到这我才想起来，wiki 还有中文版：把公式贴一下：

残差平方和SSE是:

$$\mathrm{SSE} = \sum (y_i - \hat{y}_i)^2 = \mathbf{y}^{\mathbf{T}}\mathbf{y} - \hat{\boldsymbol{\beta}}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{y}.$$

这里用的是多元的符号，我不想再复制一次了，自己到 wiki 里搜索一下“线性回归”就可以了。如果 msq<minMsq 当然就是找到了更好的一个属性，记录下来。

```
// Set parameters
if (chosen == -1) {
    if (!m_suppressErrorMessage)
        System.err.println("----- no useful attribute found");
    m_attribute = null;
```

```
    m_attributeIndex = 0;
    m_slope = 0;
    m_intercept = yMean;
} else {
    m_attribute = insts.attribute(chosen);
    m_attributeIndex = chosen;
    m_slope = chosenSlope;
    m_intercept = chosenIntercept;
}
```

这里就是记录下最佳的属性，选中属性的 index，斜率和截距，注意上面的一句话，如果没有什么有用的属性就将 slope 设为 0，而 intercept 作为 yMean，就是平行于 x 轴的直线，在平行线中，当然是它的 msq 最小了。