Good Features are Hard to Come By

2. The number of features is often too large for most feature selection algorithms to work well.
3. Principal Component Analysis (PCA) is often used to reduce the number of features before applying feature selection algorithms — but this risks throwing away crucial information.

An Algorithm to Discover a Small Set of Good Features Automatically

**Input:** lots of samples in $\mathbb{R}^D$, a small set of features $f^d$ with unacceptable high error rate.

**Goal:** new features that improve error rate.

**Repeat**
- Project samples in $\mathbb{R}^D$ down to $\mathbb{R}^d$ by $f^d$, $D \gg d$.
- Train and test a Nearest Neighbor classifier in $\mathbb{R}^d$.
- Find clusters of errors.

**Repeat**
- Select a tight cluster containing both types of errors.
- Project the cluster back to the null space $N(f^d)$.
- Find a separating hyperplane in the null space: confirm the new feature performs well.

**Until** the feature lowers the error rate sufficiently.
- Add the feature to the feature set, and set $d = d + 1$.

**Until** the error rate is satisfactory to the user.

Experiments

**Error rate as a function of number of features:**
- **4 manual:** 4 manually chosen features + 10 discovered features
- **28 manual:** 28 manually chosen features
- **PCA:** 14 features chosen by PCA
- **1PCA:** 1 feature chosen by PCA + 13 discovered features

**Training set:** 117.0M machine print samples
- 8.9M handwritten samples

**Test set:**
- 89.8M machine print samples
- 4.3M handwritten samples

Automatic Feature Discovery is Computationally Efficient

1. It took us two years of trial-and-error to find the 28 features manually.
2. For the automatically discovered features:
   - **Extract Error:** linear in the number of features, approx. 15 CPU seconds per feature
   - **Clustering:** superlinear in the number of features, 110 CPU seconds for two features to 47 CPU minutes for ten.
   - **Calculating Null Space:** average 8 CPU seconds.
   - **Populating the Error Cluster:** sublinear in the number of features, 21 CPU seconds for two features to 2.5 CPU minutes for ten.
   - **Training:** linear in the number of features, 4 CPU minutes for each feature on each image.
   - **Classification:** superlinear in the number of features, 1.5 CPU hours for five features and less than 6.5 hours for 14

Conclusion: Our algorithm competes well with both the widely used PCA method and tedious and expensive manual search