# On Identifying Important Subspaces for Approximation

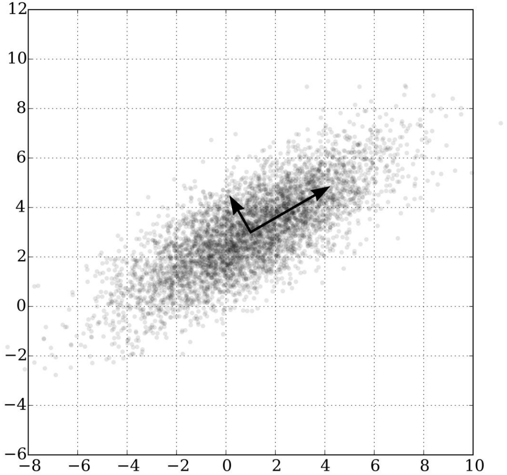
Tyler Chang, Thomas Lux

# Dimension Reduction and Feature Selection

Generally “dimension reduction” is discussed in unsupervised contexts.
(outside of approximating some function)

“Feature selection” often refers to dimension reduction *for* approximation.

**Principal Component Analysis (PCA)**




**Greedy Forward Selection**

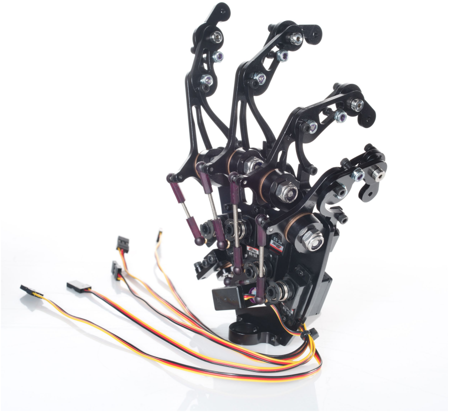
Start with no features selected

Test model using each feature individually

Add feature that maximized performance

Repeat adding one feature at a time

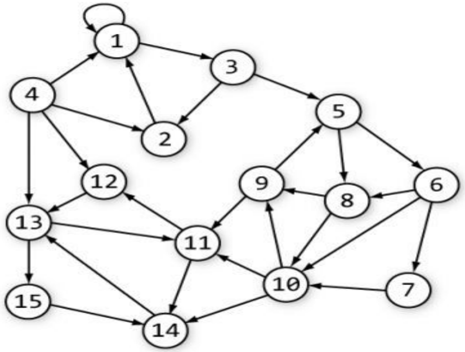
# Tough Problems in the Wild



Sophisticated control problems (e.g., robotics control)

Detecting graph/network structure

Data science predictions with high-dimensional data




# Existing Approaches to solve Tough Problems

**PCA** finds orthonormal set that maximize reconstruction variance/minimizes sum of squared residuals for input data

- PCA fails to consider effects on output

**Correlation analysis** fits a linear model and ranks dimension by linear coefficients

- Correlation analysis assumes linearity and fails to capture intervariable interactions

**LDA** finds linear separators between different classes/values

- Depends on linear separability

**Other ML techniques** such as autoencoders, regularization, and online methods

- Based on heuristics, not well understood, and convergence is not guaranteed

# Our Approach to the Tough Problems

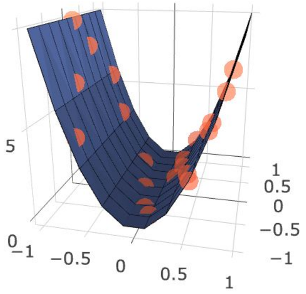
Similar to PCA, but PCA maximizes the variance of the projections onto each basis.

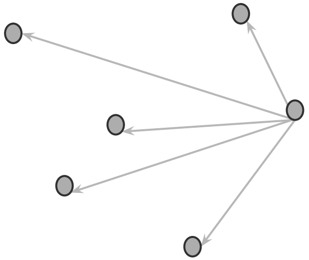
Or equivalently, PCA minimizes the sum of *squared* residuals when projecting points.

We try to maximize the 1-norm (absolute sum) of the projections, instead of variance.

This is **not** equivalent to minimizing the absolute sum of the residuals.

We incorporate function value information into the vectors, to simultaneously perform *feature selection* and *dimension reduction*.



$$Z = \left\{ \frac{(x_i - x_j)\, d(y_i, y_j)}{\|(x_i - x_j)\|_2^2} \right\}$$

# Description of PCA vs. Our Approach

PCA given $\{x_1, \dots, x_n\},\ x_i \in \mathbb{R}^d$ with mean value zero

$$\max_{\|v\|_2=1} \sum_i \frac{\langle x_i, v\rangle^2}{n-1} \qquad \text{maximizes variance}$$

$$= \min_{\|v\|_2=1} \sum_i \left(\sqrt{\langle x_i, x_i\rangle - \langle x_i, v\rangle^2}\right)^2 \qquad \text{minimizes sum of squared residuals}$$

We solve a similar problem, we want to maximize the *absolute sum* of projections for the set of points $Z$, the "between" vectors scaled by change in function value.

This will give us the "most important directions" for approximating the function.

# Description of PCA vs. Our Approach

PCA given $\{x_1, \ldots, x_n\},\ x_i \in \mathbb{R}^d$ with mean value zero

$$\boxed{\max_{\|v\|_2=1} \sum_i \langle x_i, v \rangle}$$

$$= \min_{\|v\|_2=1} \sum_i \left( \sqrt{\langle x_i, x_i \rangle - \langle x_i, v \rangle^2} \right)^2$$

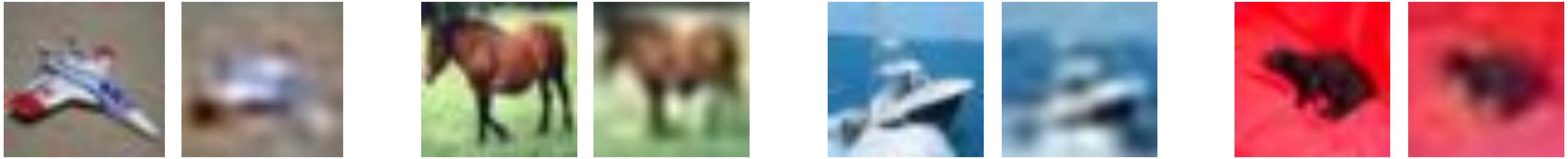
minimizes sum of squared residuals

We solve a similar problem, we want to maximize the *absolute sum* of projections for the set of points $Z$, the "between" vectors scaled by change in function value.

This will give us the "most important directions" for approximating the function.

# Early Experimental Results

Here are 24 basis vectors generated by our method for Cifar10 data.

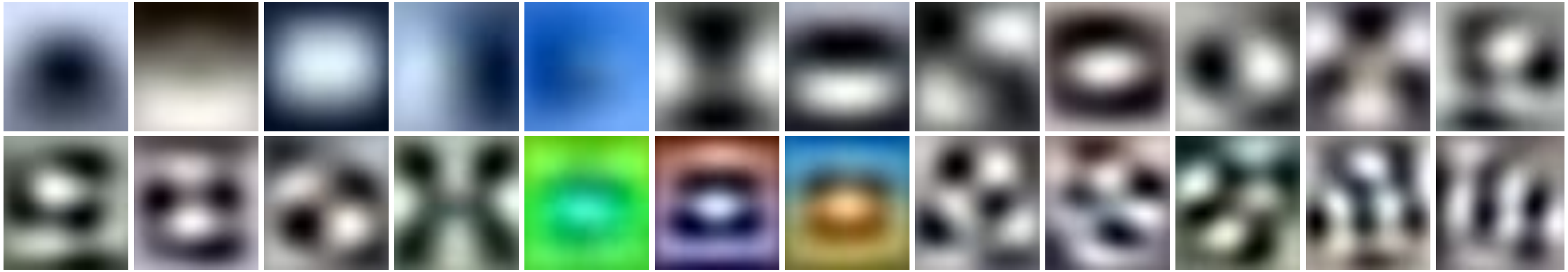


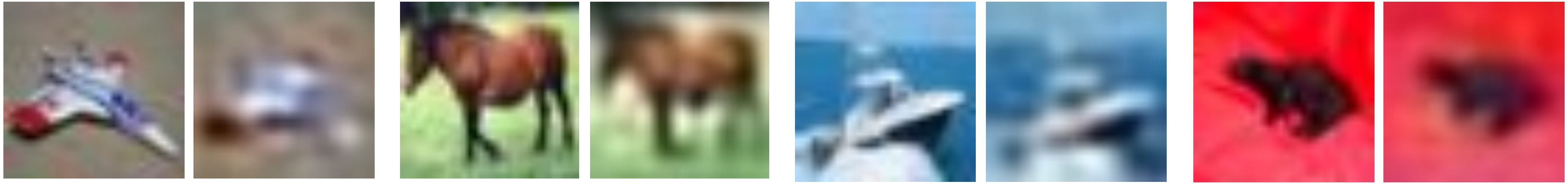
Subjectively similar to the largest coefficient Fourier decomposition components.

Likewise for early layers of a (image recognizing) convolutional neural network.

# Results with Only PCA

The first 24 components when performing PCA on the images (unmodified).





# Successes and Failures with Our Method

We used the Nearest Neighbor algorithm with the computed *important* components (20).

| *Training Data* | Raw 3000 Channels | 100 Components | 300 Components | 500 Components |
|---|---|---|---|---|
| *Relative Evaluation Time* | 1 | **.03** | .1 | .17 |
| *Cifar10 Testing Accuracy* | 23.89% | **28.86%** | 28.09% | 28.21% |
| *Best Algorithm Accuracy* | Fractional Max Pooling 96.53% | | | |

Using our transformation still does not obtain the quality of results that can be achieved by knowing the underlying data is *shift invariant* (in a special way).

# The Performance of Existing Methods

Here's the comparison with existing methods. PCA (on the raw images) is slightly better.

| *Training Data* | Raw 3000 Channels | 100 1-Norm Components | 100 PCA Components | 100 LDA Components |
|---|---|---|---|---|
| *Cifar10 Testing Accuracy* | 23.89% | 28.86% | **29.52%** | 17.48% |
| *Best Algorithm Accuracy* | Fractional Max Pooling 96.53% | | | |

We conclude that image recognition is not the best example of this method.

# The Path Forward

Attempt to predict other classes of functions

- Anomaly detection in images
- Feature weighting in other data science applications (I.e., Yelp data, other image problems, etc.)
- Graph structure problems
- Complicated control problems

Look at other interpolation/approximation techniques

- Triangulations
- Inverse distance weightings


"That's all Folks!"