

On Identifying Important Subspaces for Approximation

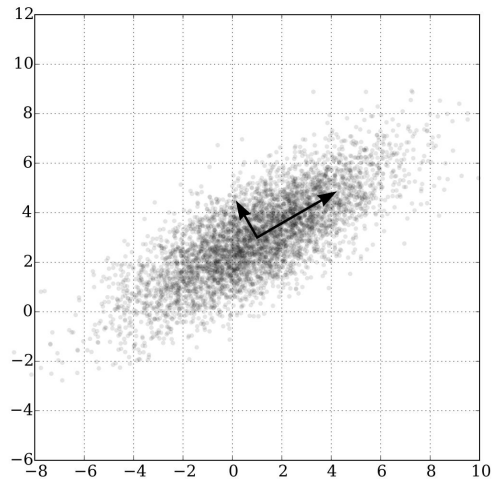
Tyler Chang, Thomas Lux

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

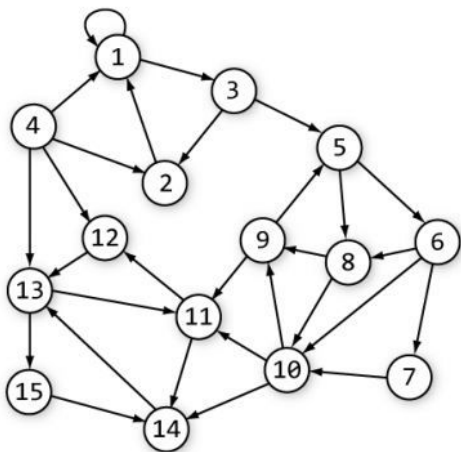
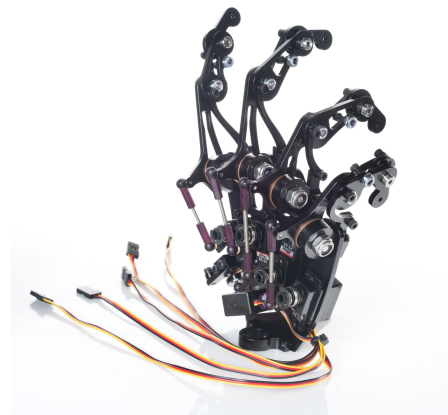
Repeat adding one feature at a time

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

Metric Component Analysis (MCA)

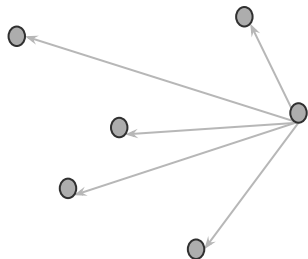
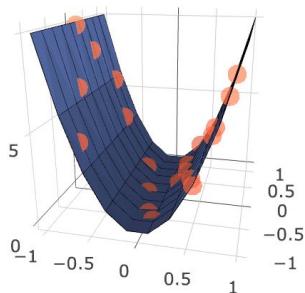
Similar to PCA, but PCA maximizes the variance of the projections onto each basis.

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

MCA maximizes the 1-norm (absolute sum) of the projections, instead of variance.

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

“Metric” refers to the incorporation of 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. MCA

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

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

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

MCA solves a similar problem, it maximizes the *absolute sum* of projections for the set of points Z , the “between” vectors scaled by change in function value.

This will give the vectors with the largest average function slope.

Description of PCA vs. MCA

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

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

MCA solves a similar problem, it maximizes the *absolute sum* of projections for the set of points Z , the “between” vectors scaled by change in function value.

This will give the vectors with the largest average function slope.

MCA is hard... we use Metric-PCA (MPCA) instead

MCA solves

$$\max_{\|v\|_2=1} \|Zv\|_1$$

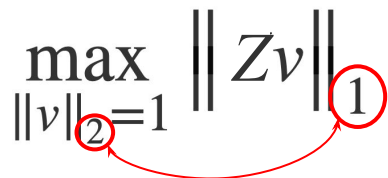
MPCA solves

$$\max_{\|v\|_2=1} \|Zv\|_2$$

Rather than finding the vector with the largest *average* function slope, we are now finding the direction with the largest *variance* in function slope.

MCA is hard... we use Metric-PCA (MPCA) instead

MCA solves

$$\max_{\|v\|_2=1} \|Zv\|_1$$


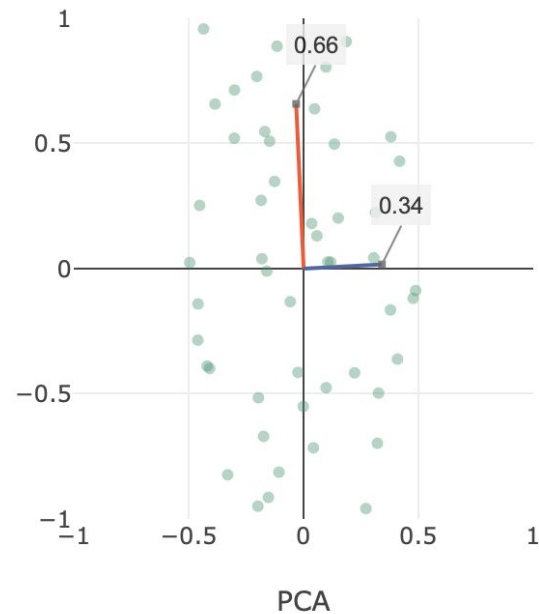
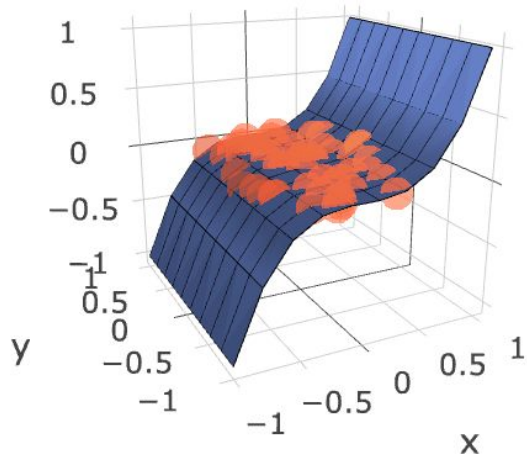
MPCA solves

$$\max_{\|v\|_2=1} \|Zv\|_2$$

This is an open problem.

Rather than finding the vector with the largest *average* function slope, we are now finding the direction with the largest *variance* in function slope.

An Example: PCA vs. MPCA



Algorithms for Evaluation

KNN – k Nearest Neighbor using 2-norm as distance metric.

Results collected so far are for *1 nearest neighbor* and *10 nearest neighbors*.

Shepard – Inverse distance weighting, predict with convex weighted sum.

Results demonstrated generally poor performance, dropped from evaluation.

Delaunay – Simplicial mesh (triangulation) for interpolation.

Data collection in progress.

LSHEP – Linear (modified) Shepard. Local linear fit using inverse distance weighting.

Data collection in progress.

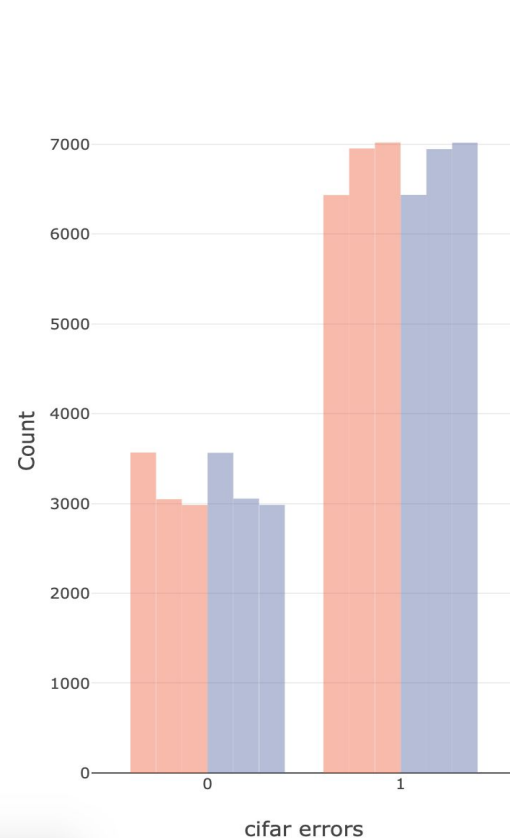
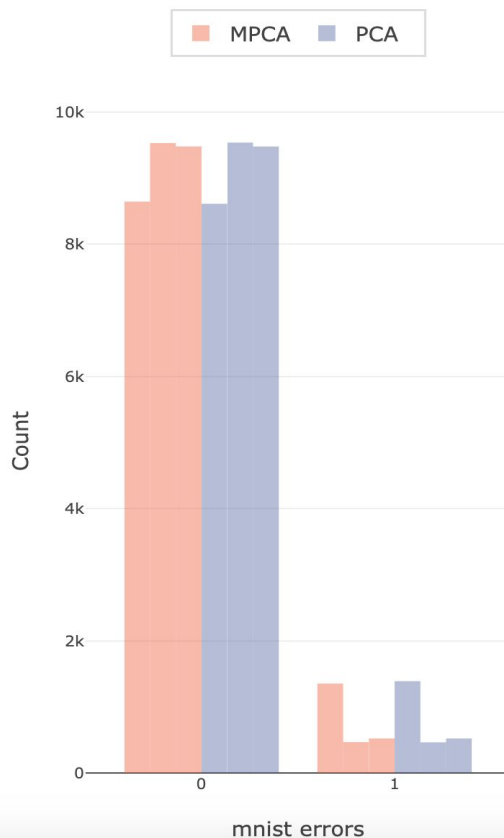
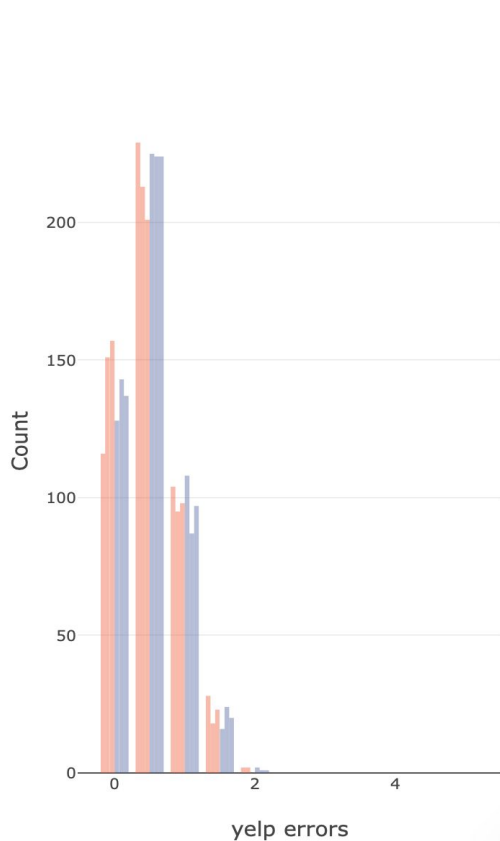
Data for Results

Yelp – 479 restaurant ratings, mostly categorical descriptors, 63 features.
Regression problem, star ratings on 0-5 scale with .5 intervals.

MNIST – 10,000 black and white images each with shape (28 x 28) having 784 channels.
Classification problem, 10 unique classes (digits).

CIFAR10 – 10,000 color images each with shape [32 x 32 x 3] having 3072 channels.
Classification problem, 10 unique classes (common objects).

Results – MPCA vs. PCA for different dimensions

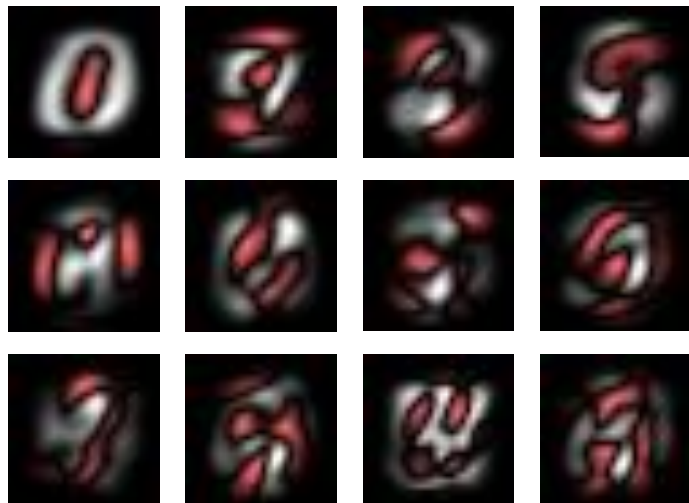


Results - Dimension Scaling

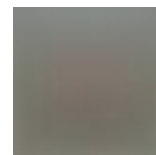
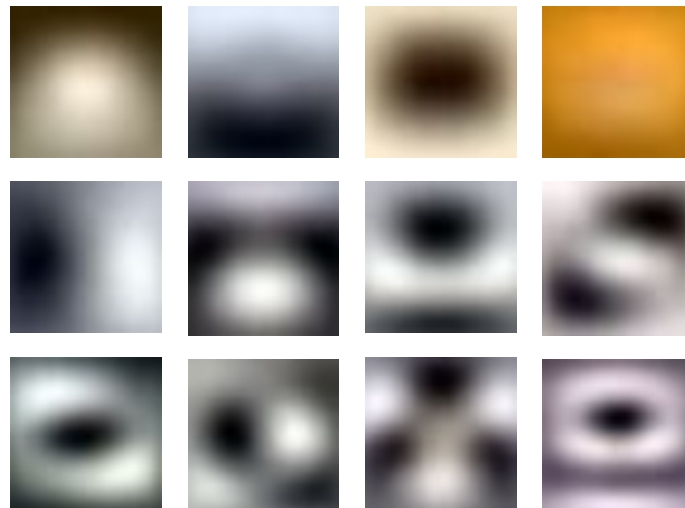
MAE (using 4-fold cross validation) for K-NN algorithms

<i>Training Data</i>	Raw data	$\frac{1}{3}$ MPCA components	1/10 MPCA components	1/100 MPCA components
<i>Yelp 64 dim</i>	0.493 (stars)	0.493 (stars)	0.496 (stars)	0.530 (stars)
<i>MNIST 784 dim</i>	5.29%	5.26%	4.63%	13.43%
<i>CIFAR10 3072 dim</i>	70.99%	70.23%	69.46%	62.46%

Metric Principle Components for MNIST and CIFAR



Average Images

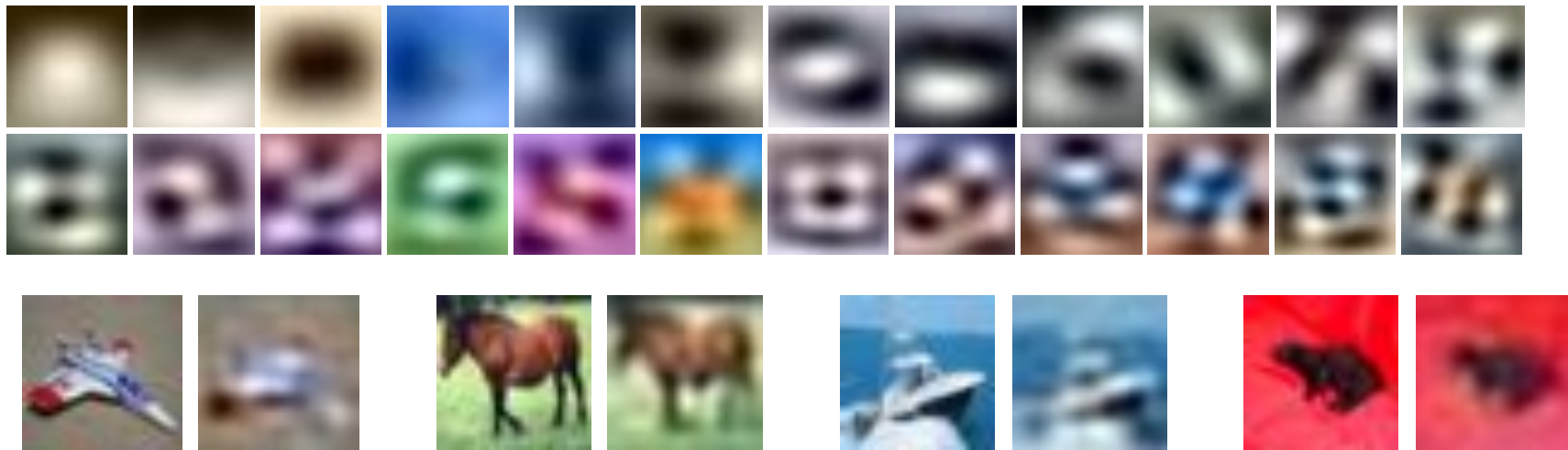




"That's all Folks!"

Early Experimental Results

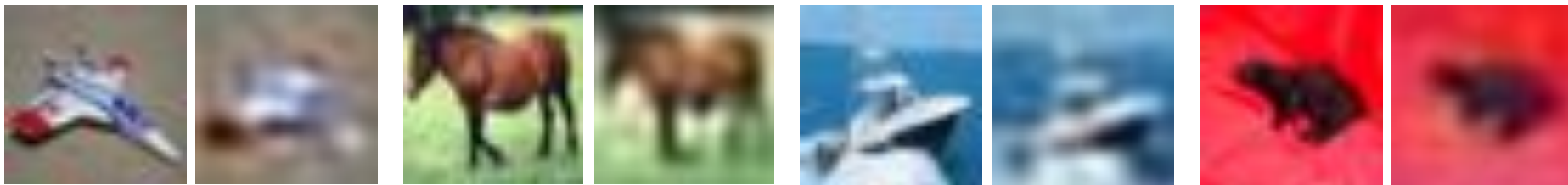
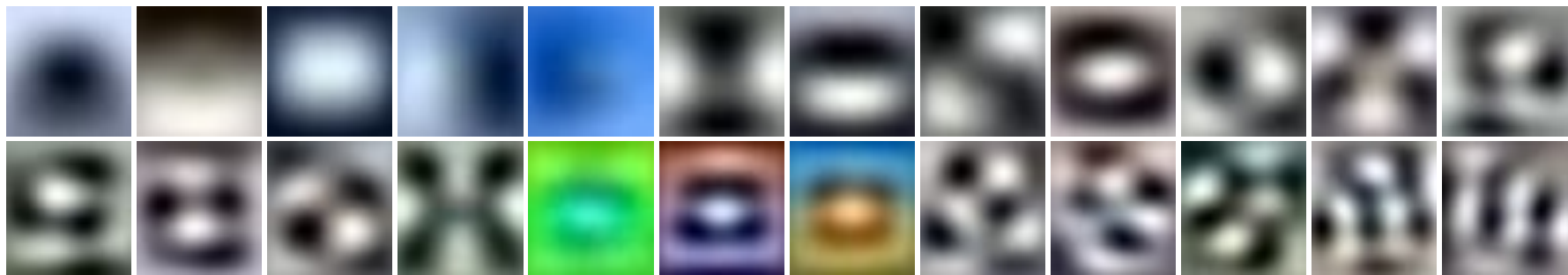
Here are 24 basis vectors generated by our method for Cifar10 data.



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).

<i>Training Data</i>	Raw 3000 Channels	100 Components	300 Components	500 Components
<i>Relative Evaluation Time</i>	1	.03	.1	.17
<i>Cifar10 Testing Accuracy</i>	23.89%	28.86%	28.09%	28.21%
<i>Best Algorithm Accuracy</i>	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.

<i>Training Data</i>	Raw 3000 Channels	100 1-Norm Components	100 PCA Components	100 LDA Components
<i>Cifar10 Testing Accuracy</i>	23.89%	28.86%	29.52%	17.48%
<i>Best Algorithm Accuracy</i>	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

