Applying Machine Learning to Image Data

Detroit Data Science

Jonathon Smereka

4/20/2017

Detroit Data Science Goals

- Overall:
  - Discussion about data science topics
  - Hands on problem solving (not just ‘book problems’)
  - Intro to ML/AI/data science approaches
  - Intro to some of the intuition of how/when to apply/build these approaches
- Tonight:
  - See what works and what doesn’t
  - Introduce a problem and discuss a potential solution
  - Provide code, data, and a baseline result to make it easy for you to work on this outside of here

Applying Machine Learning to Image Data

What does that mean?

Applying Machine Learning to Image Data

What does that mean?

“Machine learning is the science of getting computers to act without being explicitly programmed.” – Andrew Ng

Applying Machine Learning to Image Data

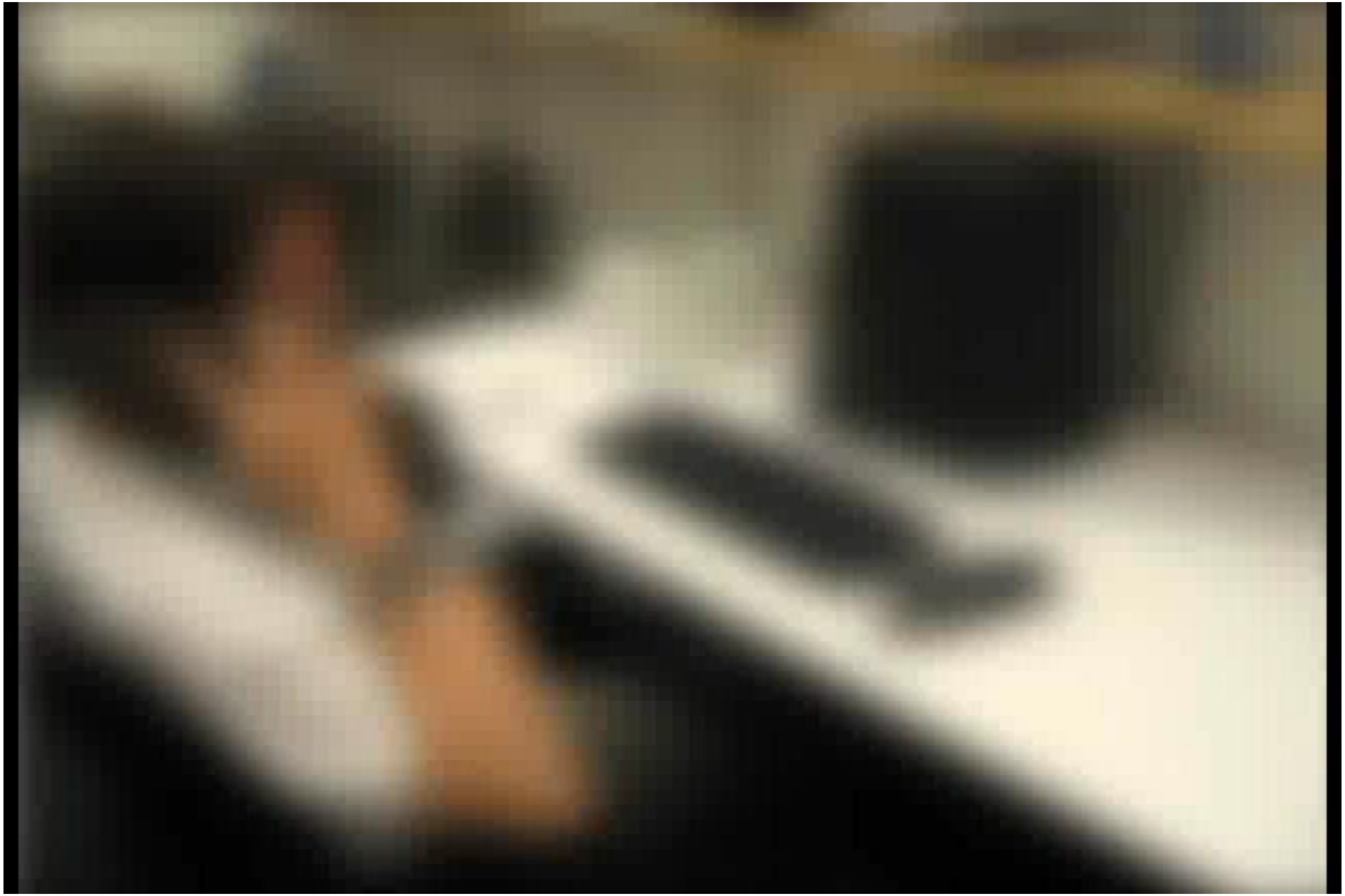
What do I want when I search for images of a “Bumblebee”?

Why is Image Data so Difficult?

Why is Image Data so Difficult?

I'm going to show you an example of how even an advanced learning mechanism can be fooled – and by advanced learning mechanism I mean you, I'll give you 3 seconds to get a look at the image

What is happening in this image?



Slide from “Recognizing and Learning Object Categories”, ICCV 2009, Li Fei-Fei, Rob Fergus, Antonio Torralba

Remember it?

Clearly it’s another day of work

Slide from “Recognizing and Learning Object Categories”, ICCV 2009, Li Fei-Fei, Rob Fergus, Antonio Torralba

Difficulties of using Image Data

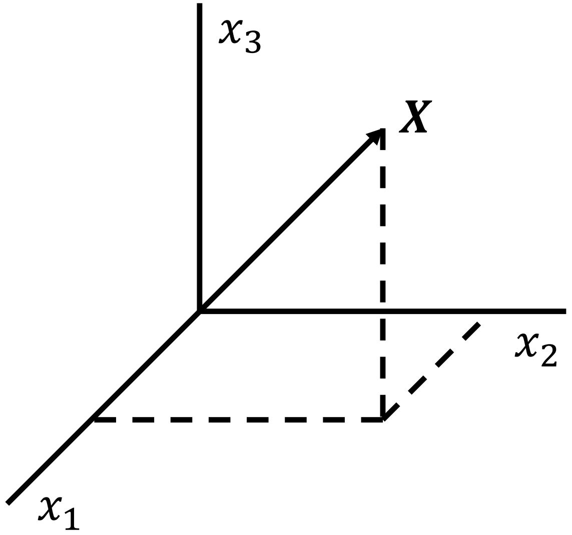
- Curse of dimensionality
- Feature engineering
- Overfitting/underfitting

Curse of Dimensionality

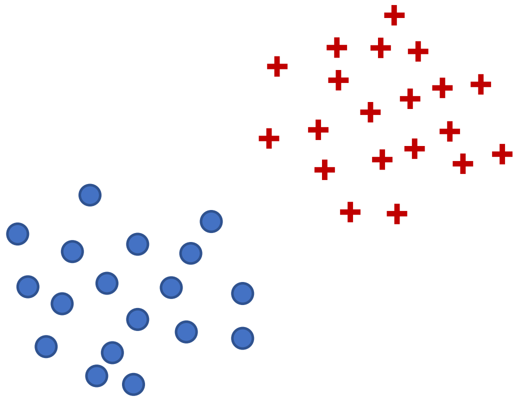
How to find structure in data embedded in a highly dimensional space

$$\boldsymbol{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

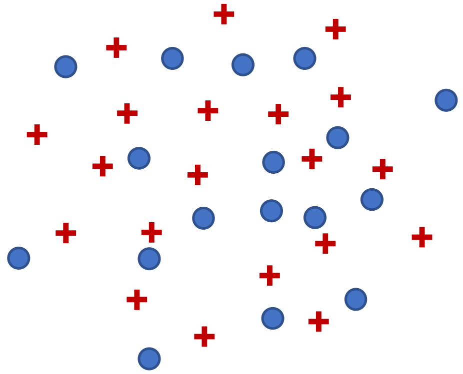
Feature Vector



Feature Space



“Good” Features



“Bad” Features

As the number of the features or dimensions grows, the amount of data we need to generalize accurately grows **exponentially**

Curse of Dimensionality

- Introduced by R. Bellman in 1957, the term refers to the problem of an exponential rise of volume of a hypersphere associated with adding extra dimensions to Euclidean space

- Not tied directly to the data dimensionality alone, rather is a joint problem involving the high number of dimensions and the inability of the processing algorithm to scale accordingly

- Every time you add another feature to increase the dimension of your input space, you're going to need exponentially more data in order to generalize the classifier more accurately

Dimensionality Reduction

- Goal:
  - Reduce the number of variables under consideration
  - Provide a compact low-dimensional encoding of a given high-dimensional data set
- Possible Solutions:
  - Remove select inputs (features) to form a smaller subset for classification
  - Combine inputs (features) to produce a new, smaller set for training
- Need:
  - Criterion for ranking good features
  - Search procedure for finding good features
- Generalization:
  - Can the process be task/problem dependent?
  - Does the result needs to be independent of the learning task?

Task-Dependent Feature Selection

- Goal: select a subset of features which provides the most output prediction capabilities (considering the input $\boldsymbol{X}$ and output $\boldsymbol{Y}$)
- Usually involves a greedy search
- Examples:
  - Data measures: Mutual information, Likelihood (via Naïve Bayes)
  - Output scores: Area under the curve (AUC), Fisher Score
  - Other classifiers: Linear Discriminant Analysis (LDA), Decision Tree, Neural Network

Task Independent Features

- Goal: replace the high-dimensional input with a small set of new features (considering only the input $\boldsymbol{X}$)
- Different from feature subset selection
- Examples:
  - Subspace approaches: Principal Component Analysis (PCA), Canonical Correlation Analysis (CCA), Singular Value Decomposition (SVD)
  - Clustering approaches: K-Means, Mixture Models, Spectral Clustering, Outlier detection

Dimensionality Reduction Issues

- Prediction accuracy vs interpretability

- Good fit vs over-fit or under-fit

- While adding structure can improve the feature discrimination ability, it may also increase the number of parameters to learn (producing a variance of estimates)

Overfitting / Underfitting

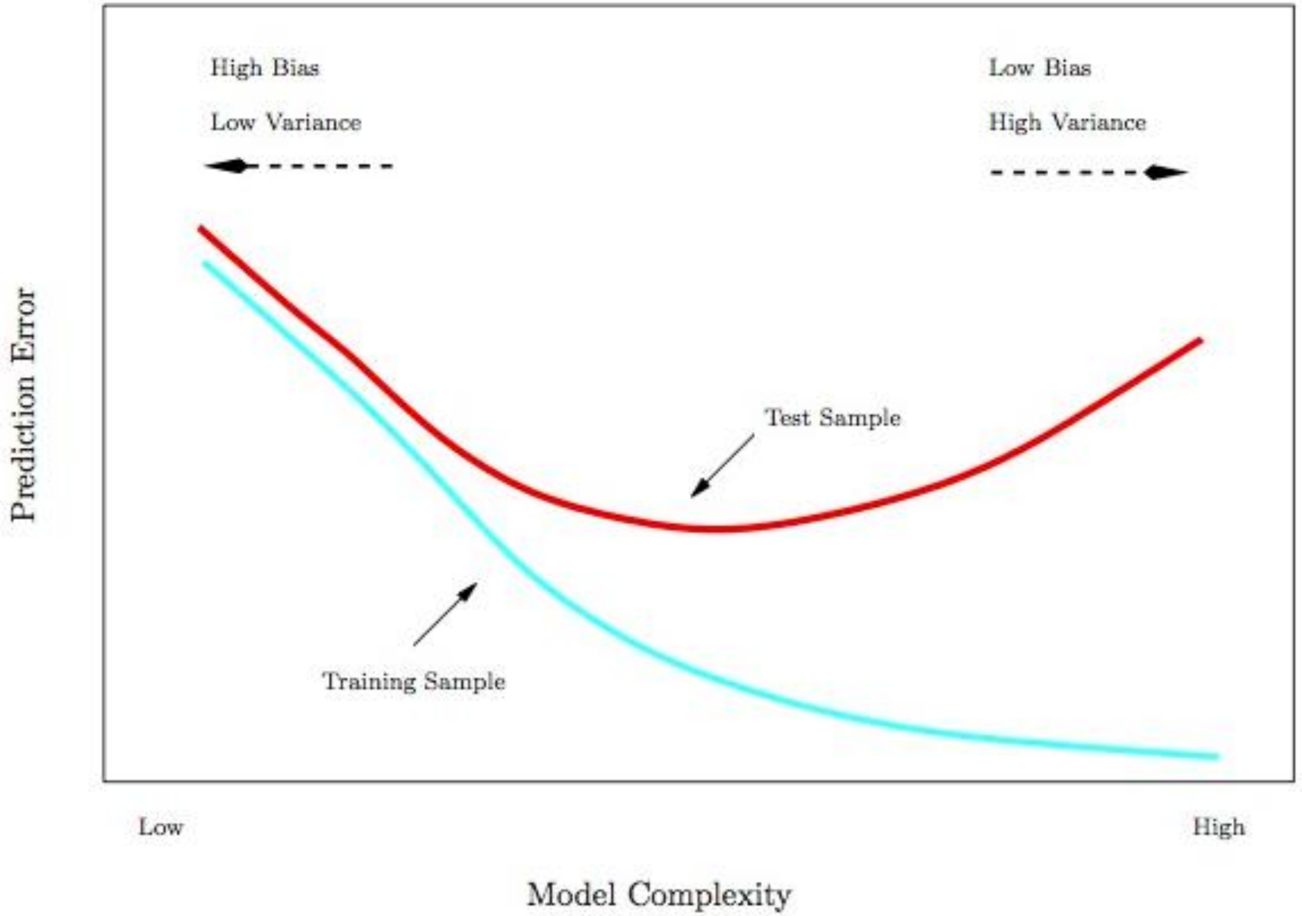


Image from Trevor Hastie and Rob Tibshirani
Stanford Statistical Learning Course

Bias = error from algorithm assumptions

Variance = error from sensitivity to small fluctuations in the training set

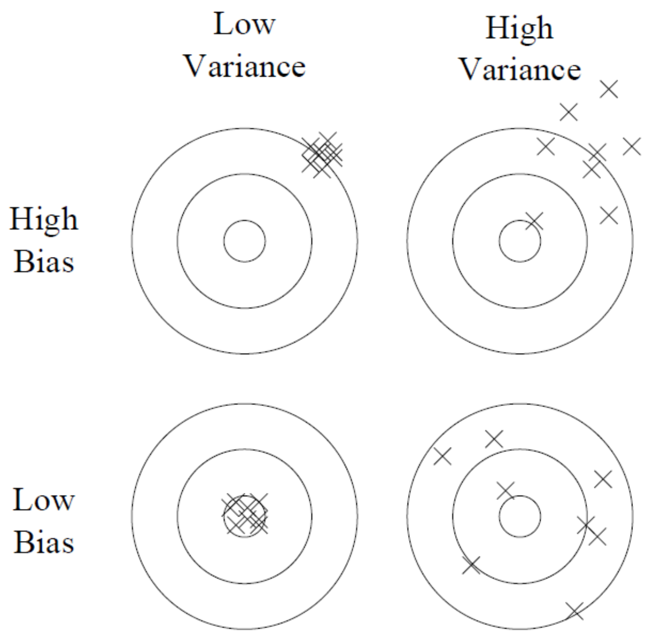
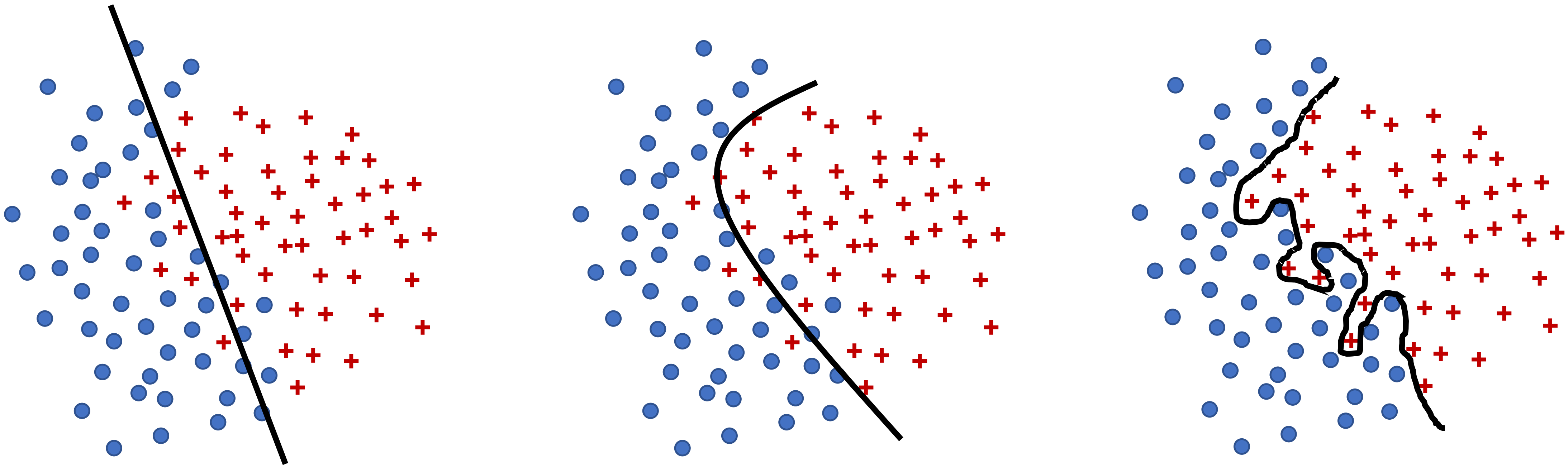


Image from Pedro Domingos, “A Few Useful Things to Know about Machine Learning”

Overfitting / Underfitting Example



Underfitting
(high bias)

the algorithm missing relevant relationships between the data

Good Fit

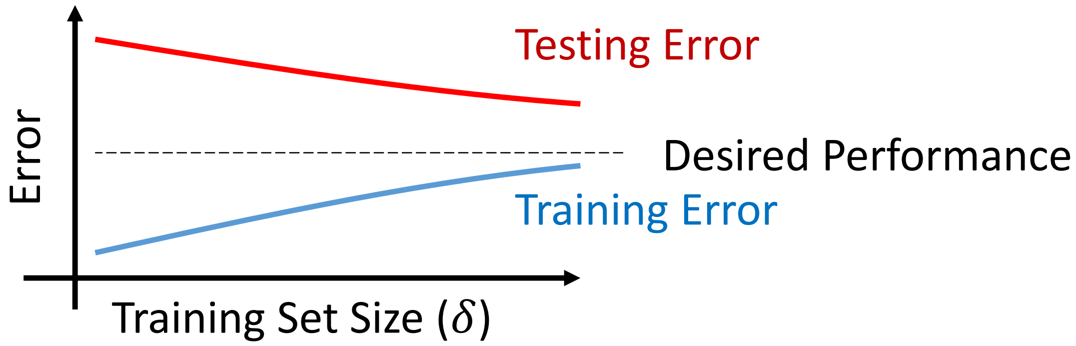
Overfitting
(high variance)

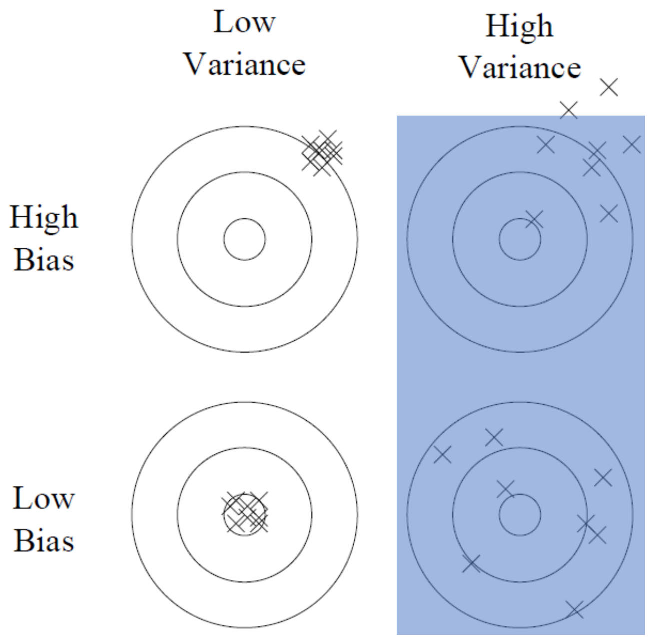
the algorithm fits too closely to the training set

Variance

- Bias = the ability of the model function to approximate the data
- Variance = the stability of the model in response to a new training example

High Variance – *associated* with overfitting



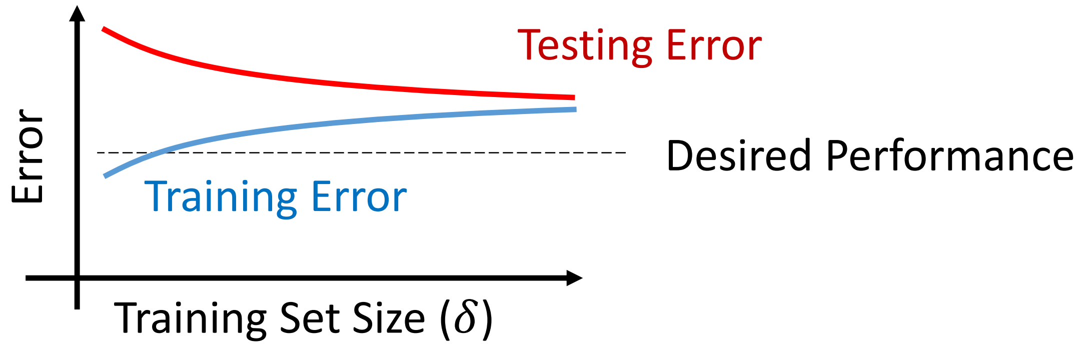


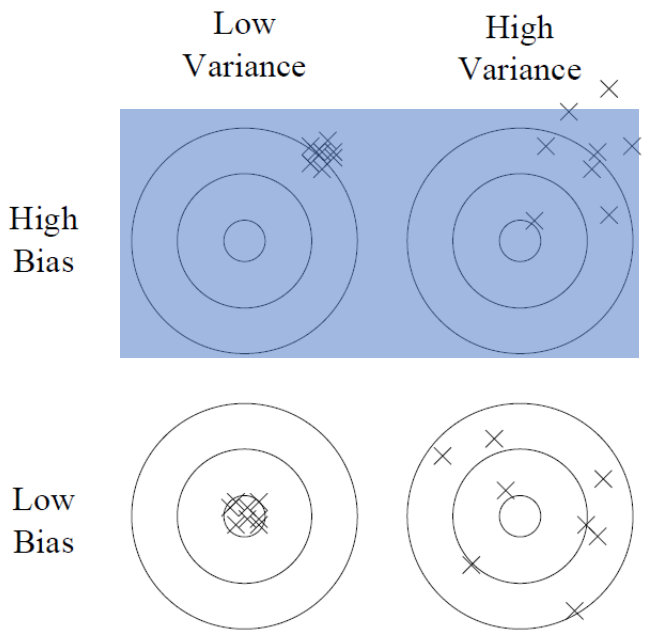
- Training error will generally increase as training data is added
- Large gap between training and testing set error
- Testing error continues to decrease with training set size

Bias

- Bias = the ability of the model function to approximate the data
- Variance = the stability of the model in response to a new training example

High Bias – *associated* with underfitting

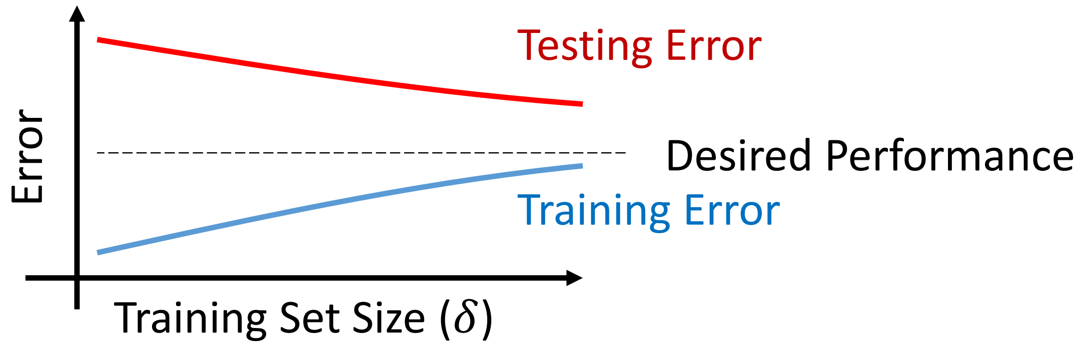




- Test error will decrease then plateau δ increases
- Small gap between training and testing set error
- High training set error

What to do about High Bias & High Variance?

High Variance – associated with overfitting



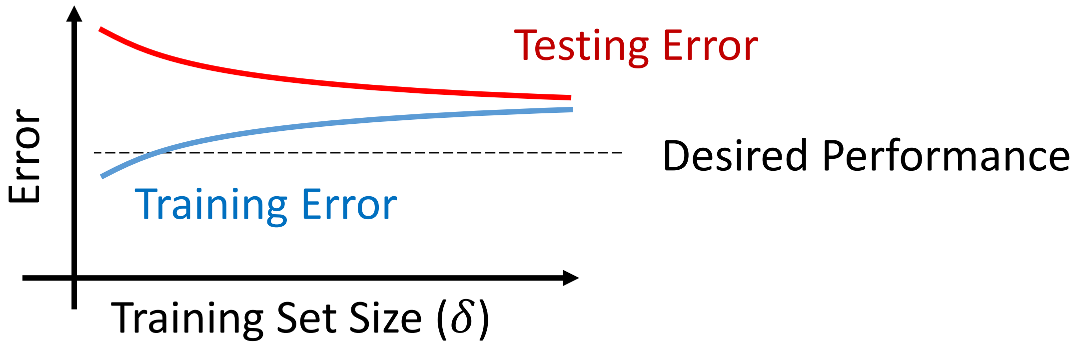
- Testing error $\downarrow$ when $\delta \uparrow$

➢ More data may help

- Training error $\uparrow$ when $\delta \uparrow$

➢ Classifier is too flexible, decreasing the number of features may help

High Bias – associated with underfitting



- More data isn't helping

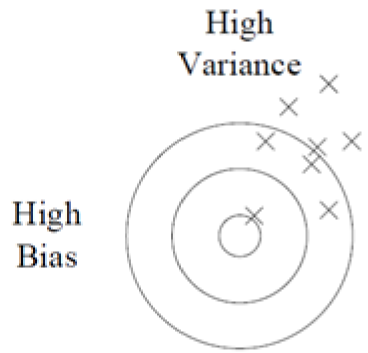
➢ Either the model or the features are failing

  ➢ Try larger or new set of features

  ➢ Change model structure

What if you have both High Variance & Bias?

- More training examples will often not help since the model won't be able to approximate the correct function to fit the data
- Don't waste time selecting features from your set
    - try new features
    - try a new model structure all together



Regularization

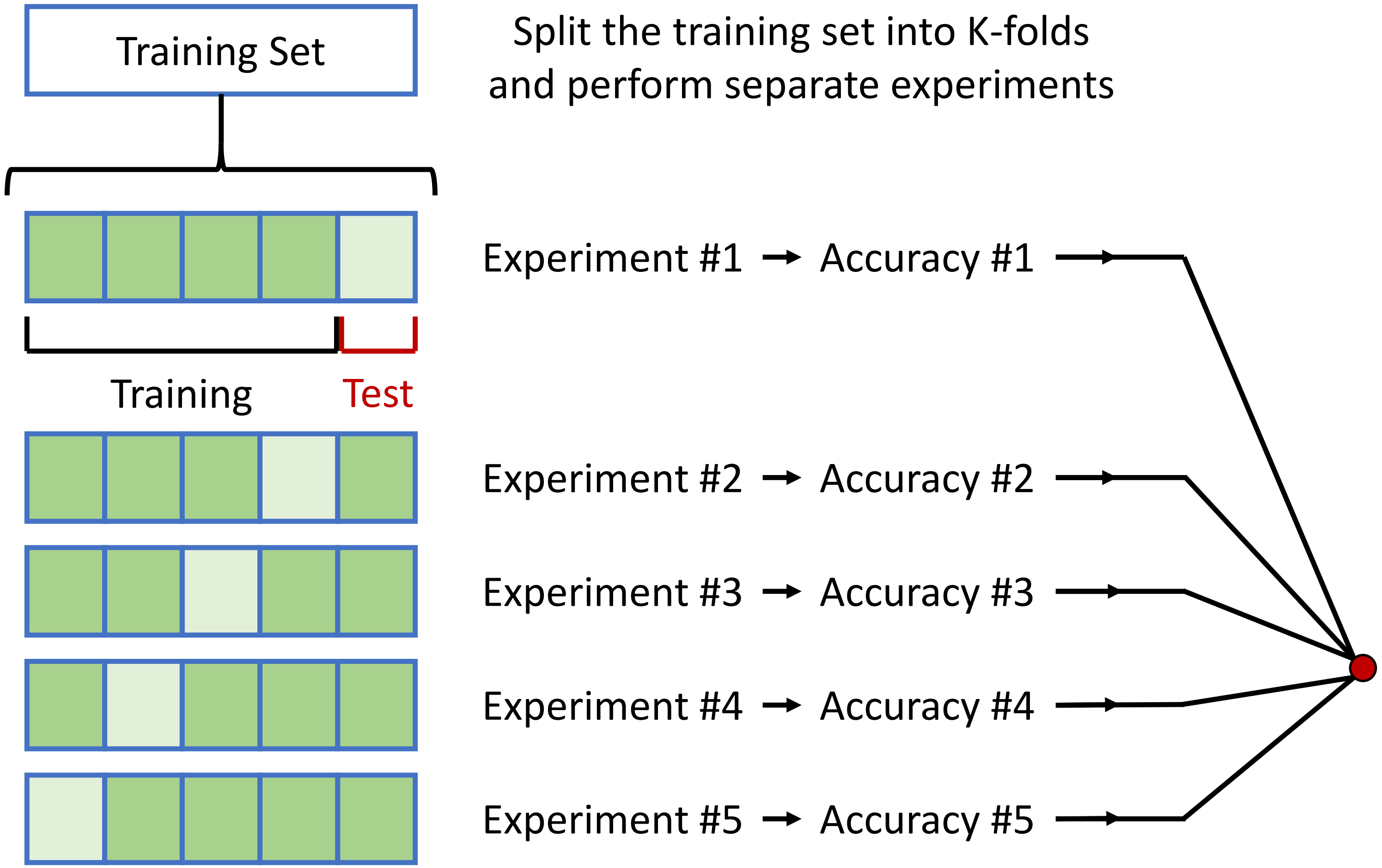
- Want to tune the model complexity to manage overfitting
- Introduce a penalty or additional information into the problem
- Examples:
  - For an error or loss function usually seen as L1 and L2 normalization

$$\arg\min \sum \mathcal{L}\big(\mathrm{y}, f(\mathrm{x})\big) + \lambda R(f)$$

$\ell_1 : R(f) = \|f\|_1$ = encourages sparsity
$\ell_2 : R(f) = \|f\|_2^2$ = enforces smoothness (doesn't force sparsity)

  - For Deep learning there's Dropout (randomly drop nodes along with their connections during training) and Batch Normalization (normalize each training min-batch)

Cross-validation



Process for optimal hyperparameter selection (e.g., λ), repeated for each parameter combination

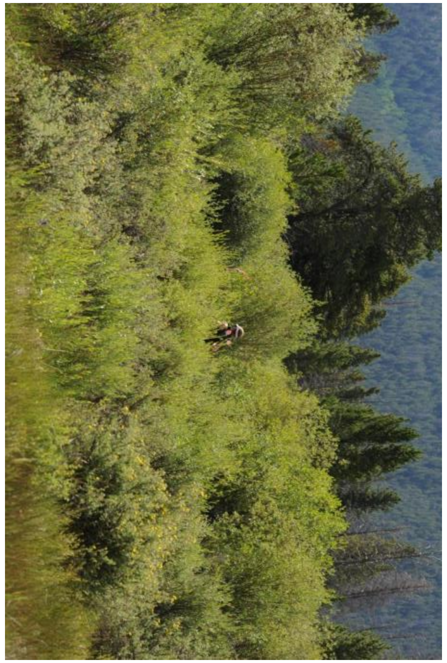
Total accuracy is the average across the folds

Use the set of parameters which define the model with the highest accuracy

Tonight's Task: Image Orientation Detection

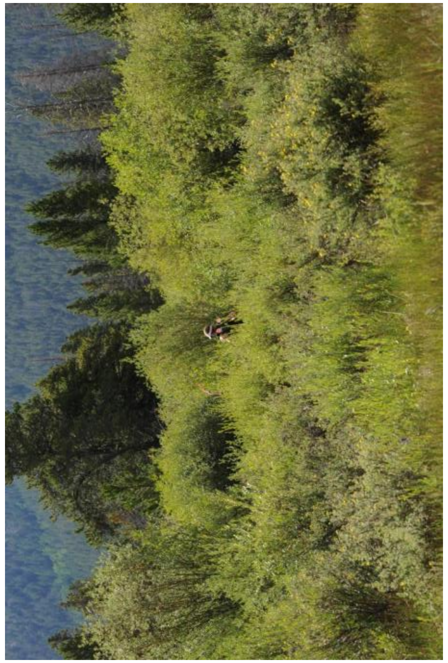
0°

90°



180°

270°

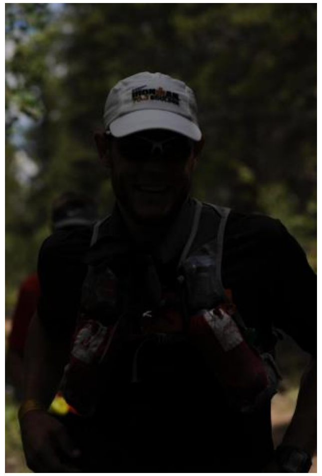


Tonight’s Task: Image Orientation Detection

- 4 possible classes: $0^{\circ}, 90^{\circ}, 180^{\circ}, 270^{\circ}$
- Image features will be provided for you
  - Histogram of Oriented Gradients
  - Spatial color moments (3 mean and 3 variance values of L, U, and V)
  - Normalized spatial color moments
  - Principal Component Analysis
  - Linear Discriminant Analysis
- // TODO: Choose the combination of features and parameter values for training an SVM for classification

Training Data: 2149 images of people on bikes

Testing Data: 626 images of people running



Tonight’s Task: What’s included?

- Python code: function_list.py and student.py
- Random assignment and rotation of training and test images with extracted features:
  - /img_idx/trainingdata.pckl and /img_idx/testingdata.pckl – image indices
  - /features/trainingdata_*_8x8.pckl – training data features
  - /features/testingdata_*_8x8.pckl – testing data features
- Conference and journal papers which concern image orientation detection: /papers/*.pdf
- An overview SVMs & the feature extraction approaches: /ppt/features.pdf

Results

All Results

Dim Reduction	# of LDA Components	HOG	SpCM	SpCM + HOG	Normalized SpCM	Normalized SpCM + HOG
LDA	2	72.52%	75.40%	88.66%	28.59%	24.12%
	3	73.48%	76.04%	91.37%	18.21%	**11.34%**
PCA+LDA	2	70.93%	77.48%	89.14%	25.24%	71.57%
	3	74.92%	76.68%	**92.97%**	19.97%	70.77%

Using 200 PCA vectors, 8x8 SpCM block configuration, image size of 200x200 pixels, 20x20 rectangular HOG cell size, 1x1 HOG cell normalization, and 4 HOG orientations per cell

Best result was Spatial Color Moments (SpCM) + HOG features with PCA and LDA dimensionality reduction techniques

SVM Parameters: ‘poly’ kernel, C = 1, degree = 3, gamma = 0.001

How necessary was PCA on the best results?

Confusion Matrices for SpCM+HOG using 3 LDA components:

No significant change in the number of correct classifications

With PCA (92.97% Accuracy)

True Label \ Predicted Label	0°	90°	180°	270°	
0°	145	6	4	4	14
90°	3	151	1	1	5
180°	4	1	144	7	12
270°	0	4	9	142	13
	7	11	14	12	

With PCA
(92.97% Accuracy)

Without PCA (91.37% Accuracy)

True Label \ Predicted Label	0°	90°	180°	270°	
0°	146	10	3	0	13
90°	8	147	1	0	9
180°	4	2	148	2	8
270°	2	2	20	131	24
	14	14	24	2	

Without PCA
(91.37% Accuracy)

Largest differences

LDA performance overall?

Dim Reduction	# of LDA Components	HOG	SpCM	SpCM + HOG	Normalized SpCM	Normalized SpCM + HOG
LDA	2	72.52%	75.40%	88.66%	28.59%	24.12%
	3	73.48%	76.04%	91.37%	18.21%	**11.34%**
PCA+LDA	2	70.93%	77.48%	89.14%	25.24%	71.57%
	3	74.92%	76.68%	**92.97%**	19.97%	70.77%

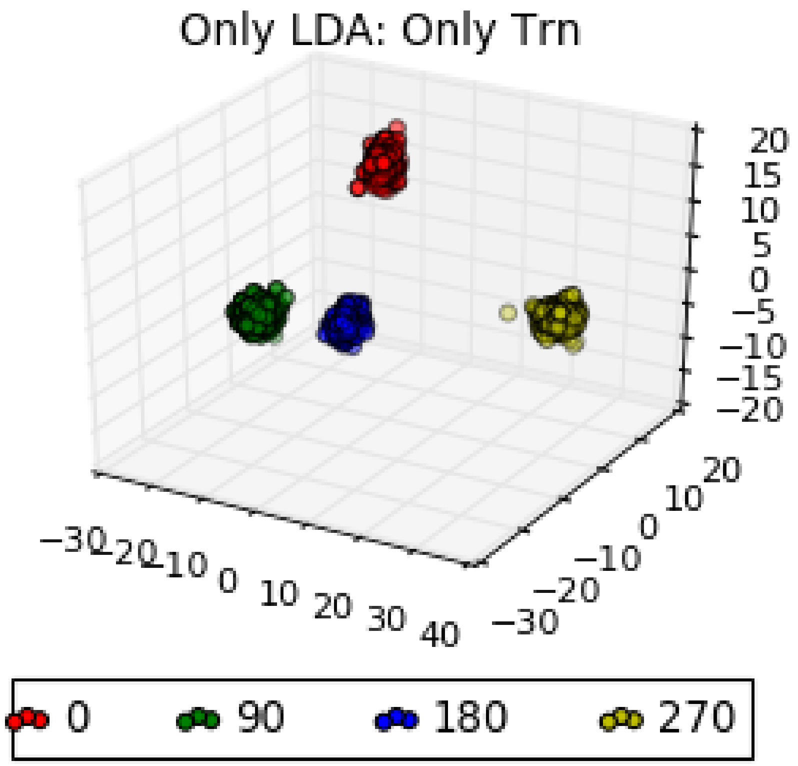
- When using Normalized SpCM features, fewer LDA components was more effective
- When not using Normalized SpCM features, more LDA components was more effective

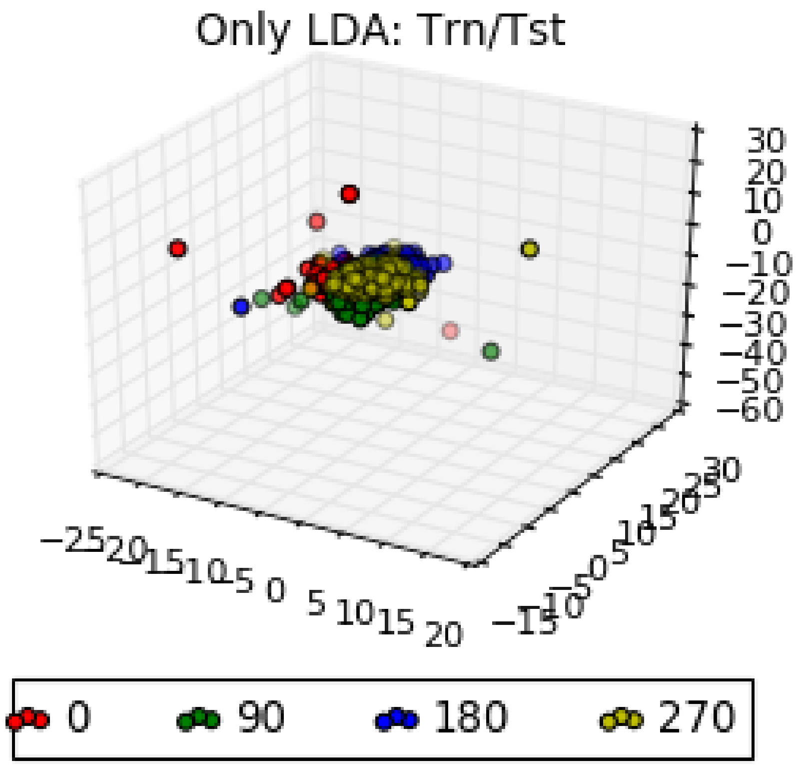
PCA performance overall?

Dim Reduction	# of LDA Components	HOG	SpCM	SpCM + HOG	Normalized SpCM	Normalized SpCM + HOG
LDA	2	72.52%	75.40%	88.66%	28.59%	24.12%
	3	73.48%	76.04%	91.37%	18.21%	**11.34%**
PCA+LDA	2	70.93%	77.48%	89.14%	25.24%	71.57%
	3	74.92%	76.68%	**92.97%**	19.97%	70.77%

- In general, it appears that including PCA can improve performance
- In one case the performance improvement was significant (otherwise the result was only incremental and not significant)

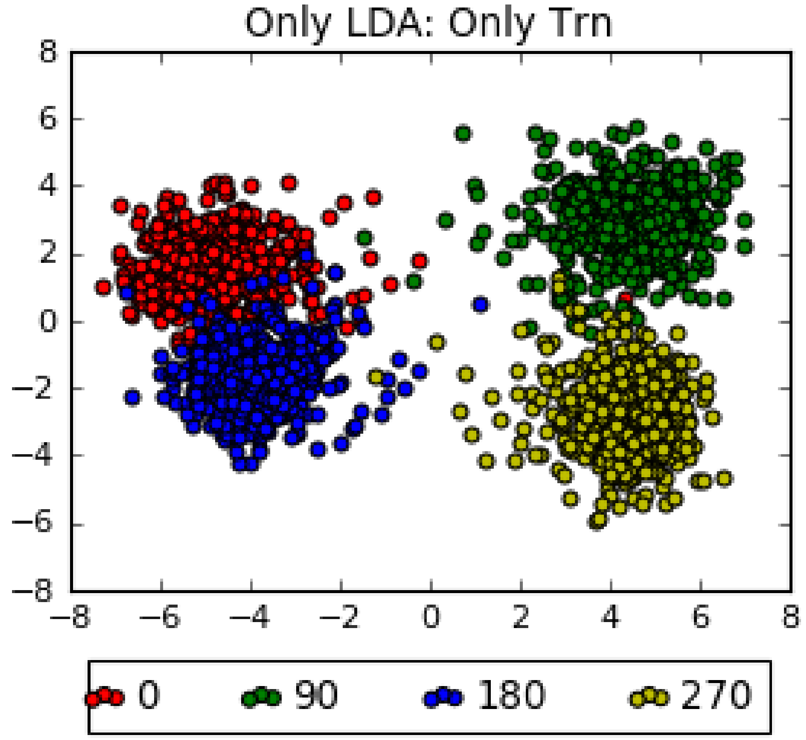
Why didn’t Normalization for SpCM work?

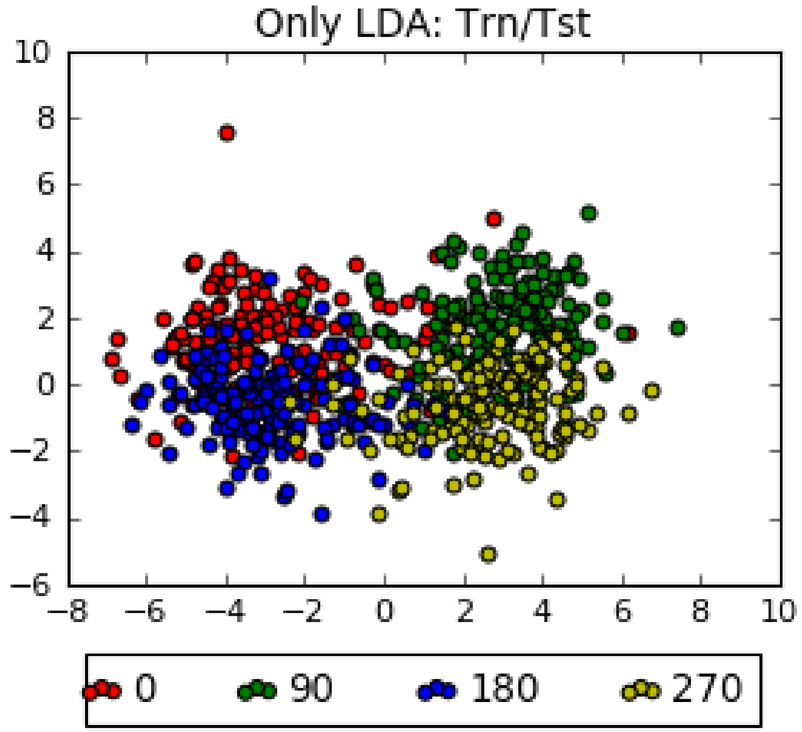




Overfitting!

The exception: PCA+LDA on Normalized SpCM+HOG





Got lucky, but the classifier is probably not very generalizable

Possible Extensions

- Different Features
- Increase the amount of training data
- Adding semantic information (context)
- More robust classifier
- Changing parameters (e.g., block size, image resolution, etc.)
- Using a different dimensionality reduction technique

Wrap up

- Features are hard
- Remember that because you have a hammer, not everything is a nail
- Things to try on your own:
  - Examine at the decision boundaries for the SVMs found during grid search
  - Train a classifier and test it on the unknown test data (found on github)
  - Download a dataset from one of the included research articles to train/test your classifier and compare results