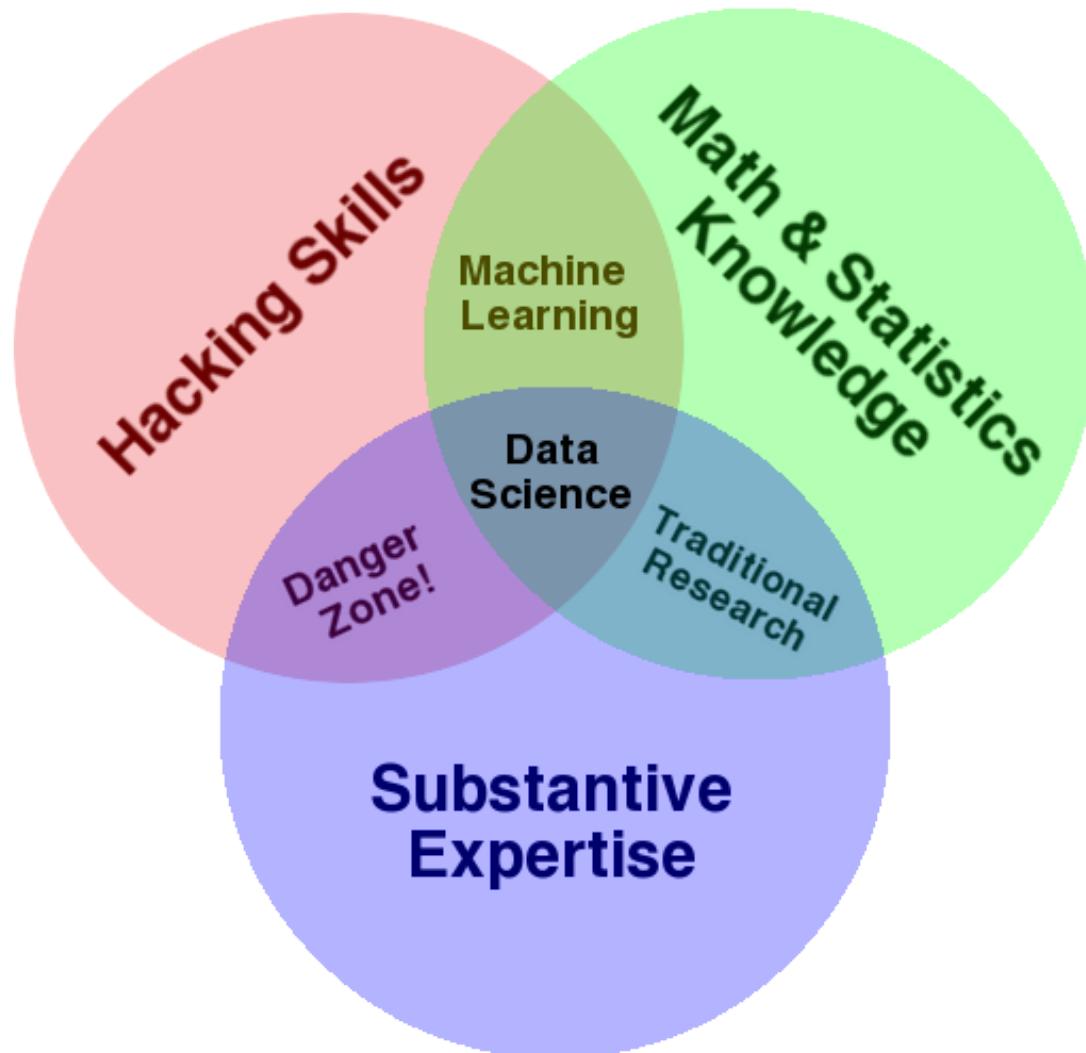


Machine Learning



<http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram>

- What is ML?
- Supervised Learning:
 - Regression
 - Classification
- Unsupervised Learning:
 - Clustering
 - Anomaly Detection
- Data Science Pipelines

What is Machine Learning?

Short Answer: The offspring of Statistics and Computer Science

Better Answer: A set of models which aim to learn something about a data set to apply that knowledge to new data

- Using labels from *training* data to **classify** new objects (e.g. images, digits, webpages)
- Learning the relationship between explanatory *features* and response variable to **predict** for new data (e.g., stock market)
- Discovering natural **clustering** structure in data
- Detecting **low-dimensional structure** in high-dimensional data
- Finding **outliers** in large data sets

Machine Learning with scikit-learn



*Make sure you have
version ≥ 0.13
installed*

conda update scikit-learn
pip install scikit-learn

<http://scikit-learn.org/stable/>

also: <http://mipy.sourceforge.net/>, <http://orange.biolab.si/>

Supervised Learning

Use *training* set of (\mathbf{x}, y) pairs to learn to predict y for new \mathbf{x}

Regression - predicting **continuous outcome** (y) variable
from a vector of input features (\mathbf{x})

- Linear Regression: `linear_model.LinearRegression`
- Lasso & Ridge Reg.: `linear_model.Lasso` / `linear_model.Ridge`
- Gaussian Process Regression: `gaussian_process.GaussianProcess`
- Nearest Neighbor Regression: `neighbors.KNeighborsRegressor`
- Support Vector Regression: `svm.SVR`
- Regression Trees: `tree.DecisionTreeRegressor`
... and more!

[About](#) [Citation Policy](#) [Donate a Data Set](#) [Contact](#)

Google™ Custom Search

Search

Machine Learning Repository

[Center for Machine Learning and Intelligent Systems](#)[View ALL Data Sets](#)

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 253 data sets as a service to the machine learning community. You may [view all data sets](#) through our searchable interface. Our [old web site](#) is still available, for those who prefer the old format. For a general overview of the Repository, please visit our [About page](#). For information about citing data sets in publications, please read our [citation policy](#). If you wish to donate a data set, please consult our [donation policy](#). For any other questions, feel free to [contact the Repository librarians](#). We have also set up a [mirror site](#) for the Repository.

Supported By:



In Collaboration With:



Latest News:

- 2013-04-04: Welcome to the new Repository admins Kevin Bache and Moshe Lichman!
2010-03-01: [Note](#) from donor regarding Netflix data
2009-10-16: Two new data sets have been added.
2009-09-14: Several data sets have been added.

Newest Data Sets:

2013-09-06:



- Reuters RCV1
RCV2
[Multilingual](#),
[Multiview Text](#)
[Categorization](#)
[Test collection](#)

Reuters
RCV1/RCV2
[Multilingual](#),
[Multiview Text](#)

2013-09-04:



Most Popular Data Sets (hits since 2007):

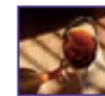
473145:

[Iris](#)

329520:

[Adult](#)

285868:

[Wine](#)

Error Estimation & Model Selection

Q: How will our model perform on future data?

So far, we've split the data, using one set to **train** the model and the other to **test** its performance

scikit-learn can do cross-validation for us!

```
from sklearn import cross_validation
```

Better procedure:
Cross-Validation

K-fold CV - Randomly split the training data into K folds. For each $k=1, \dots, K$, train model only on the data not in fold k & predict for data in fold k . Compute performance metric over CV predictions.

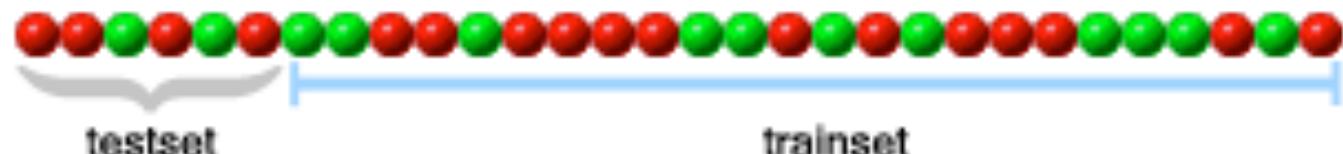
Leave-one-out (LOO) CV - K-fold CV with $K = \text{number of training points}$.

Cross-validation generators

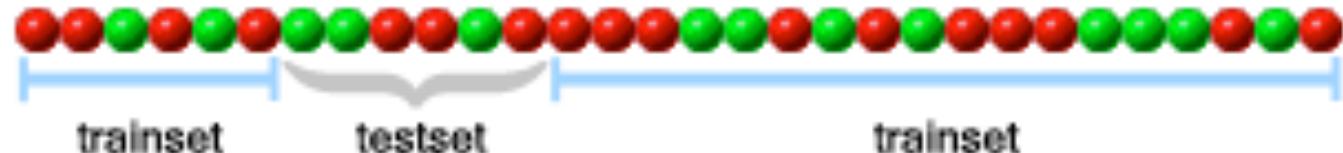
KFold (n, k)	StratifiedKFold (y, k)	LeaveOneOut (n)	LeaveOneLabelOut (labels)
Split it K folds, train on K-1 and then test on left-out	It preserves the class ratios / label distribution within each fold.	Leave one observation out	Takes a label array to group observations

ONE ITERATION OF A 5-FOLD CROSS-VALIDATION:

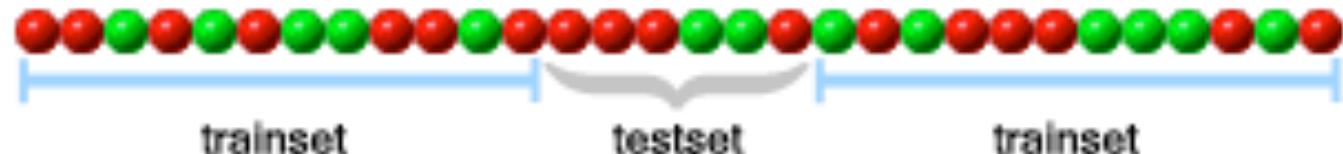
1-ST FOLD:



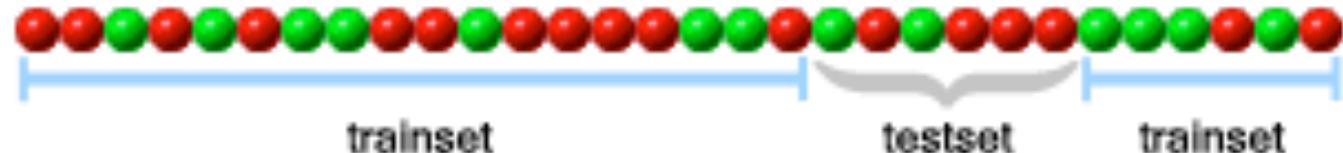
2-ND FOLD:



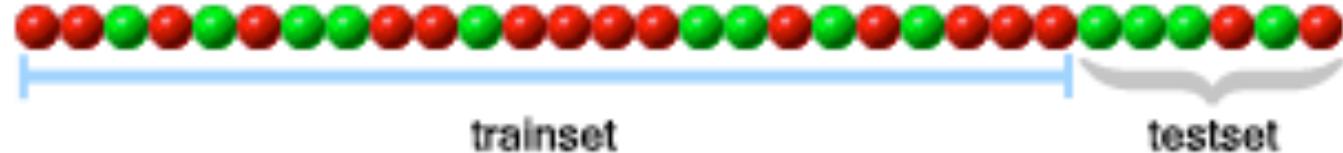
3-RD FOLD:



4-TH FOLD:



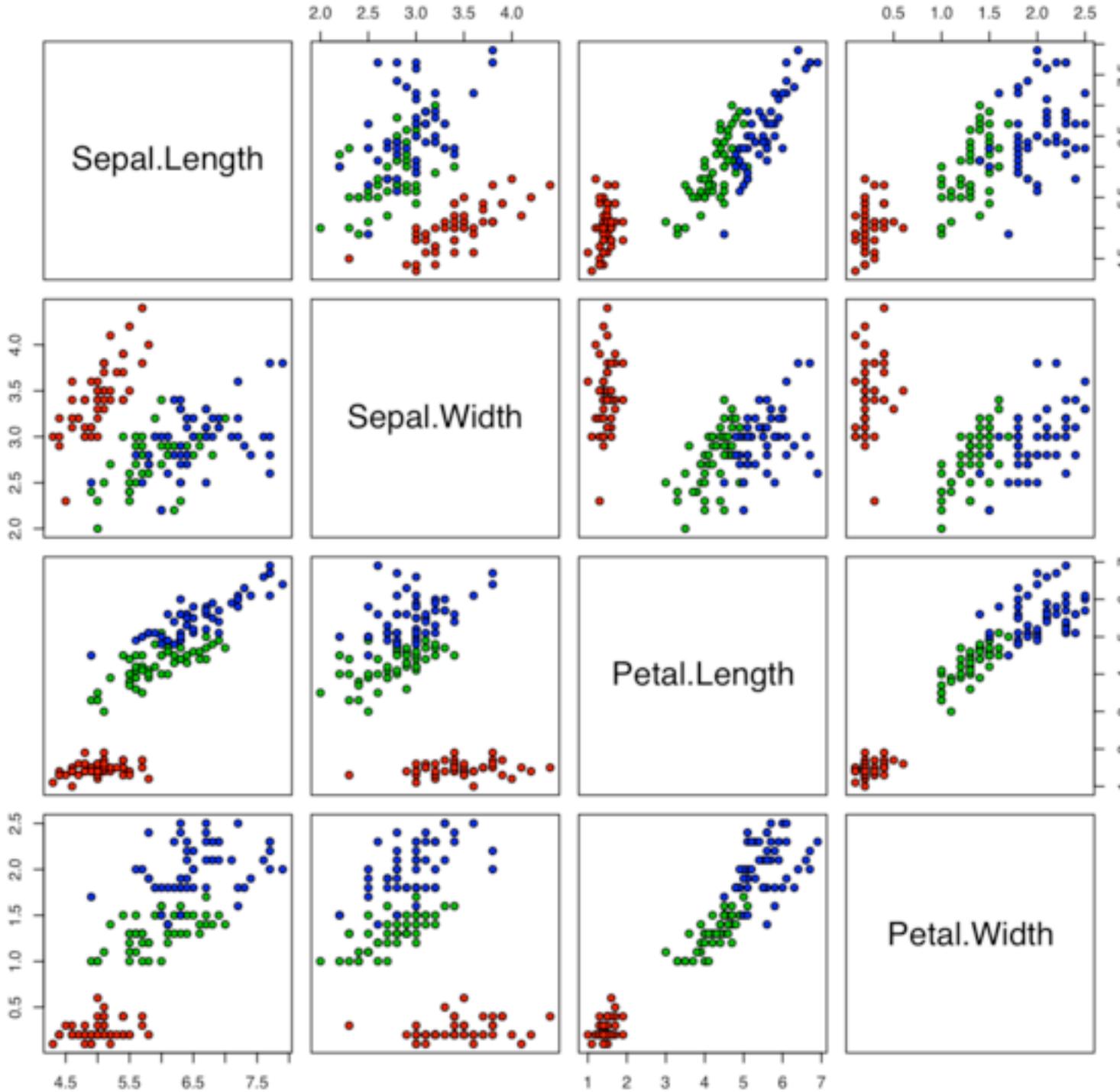
5-TH FOLD:



Supervised Learning

Classification - predicting the **discrete class** (y) of an object from a vector of input *features* (\mathbf{x})

Iris Data (red=setosa,green=versicolor,blue=virginica)



setosa



veriscolor



virginica

Supervised Learning

Classification - predicting the **discrete class** (y) of an object from a vector of input **features** (\mathbf{x})

e.g. $\mathbf{x}_i = [5.1, 3.5, 1.4, 0.2]$

[*sepal length (cm)*, *sepal width (cm)*, *petal length (cm)*, *petal width (cm)*]

e.g. $y_i = \text{"Iris-Setosa"}$

For Iris: Number of (\mathbf{x}, y) “instances” = 150
Number of classes = 3

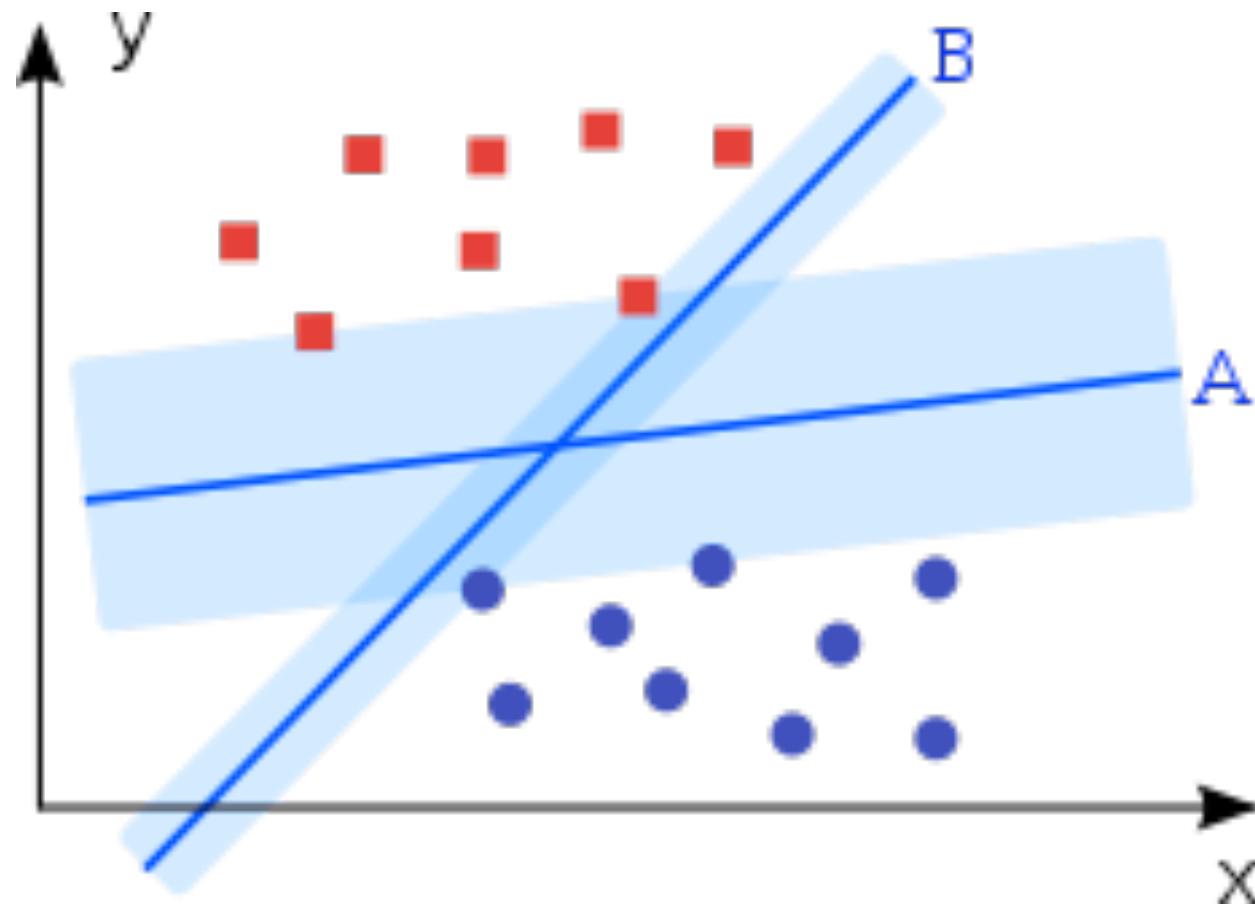
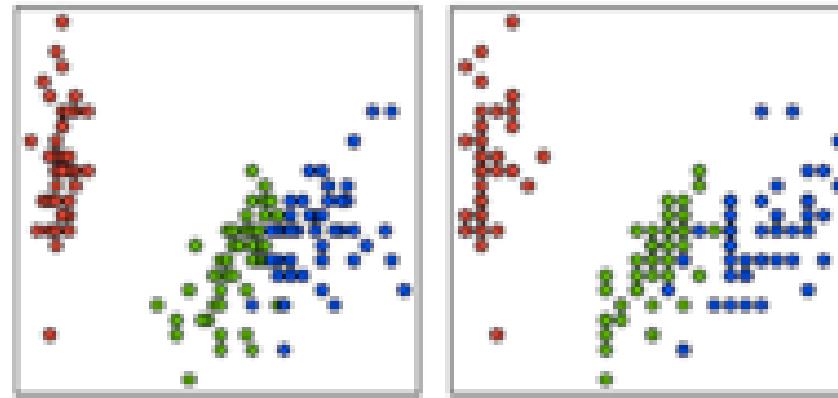
Notebook...

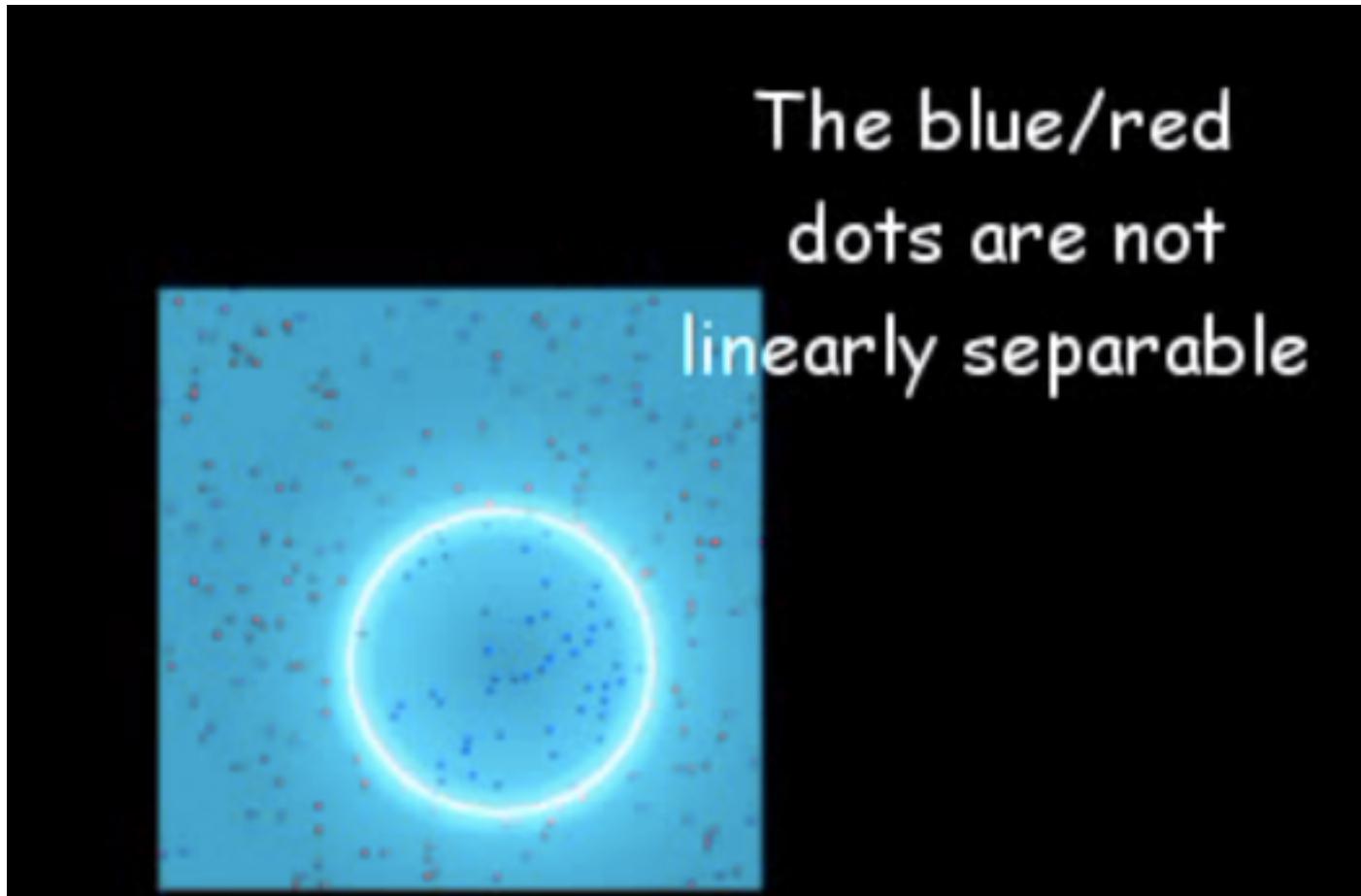
Supervised Learning, cont.

Classification - predicting the **discrete class (y)** of an object from a vector of input **features (x)**

- **Logistic Regression:** `linear_model.LogisticRegression`
- **KNN Classification:** `neighbors.KNeighborsClassifier`
- **LDA / QDA:** `lda.LDA` / `lda.QDA`
- **Naive Bayes:** `naive_bayes.GaussianNB`
- **Support Vector Machines:** `svm.SVC`
- **Classification Trees:** `tree.DecisionTreeClassifier`
- **Random Forest:** `ensemble.RandomForestClassifier`
- **Multi-class & multi-label Classification is supported:**
`multiclass.OneVsRest` `multiclass.OneVsOne`
- **feature_selection:** recursive, L1, and tree-based

Support Vector Machines (1992)



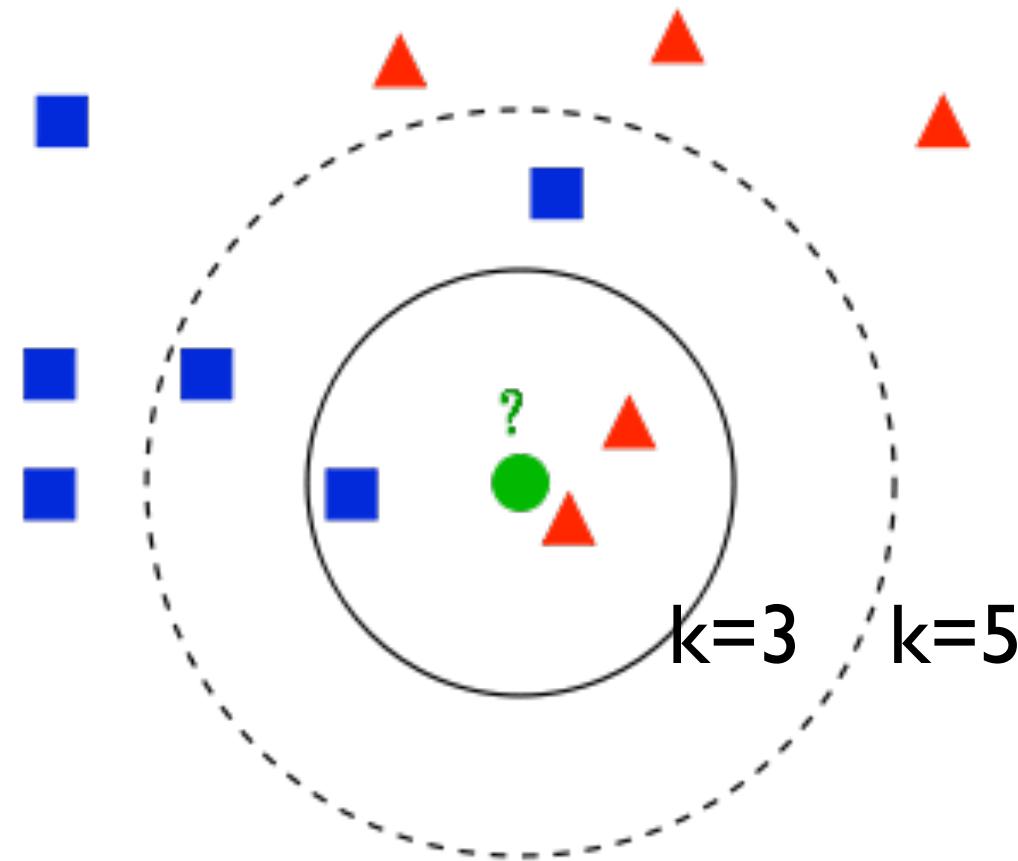


SVM with polynomial kernel visualization

<http://www.youtube.com/watch?v=3liCbRZPrZA>

Supervised Learning

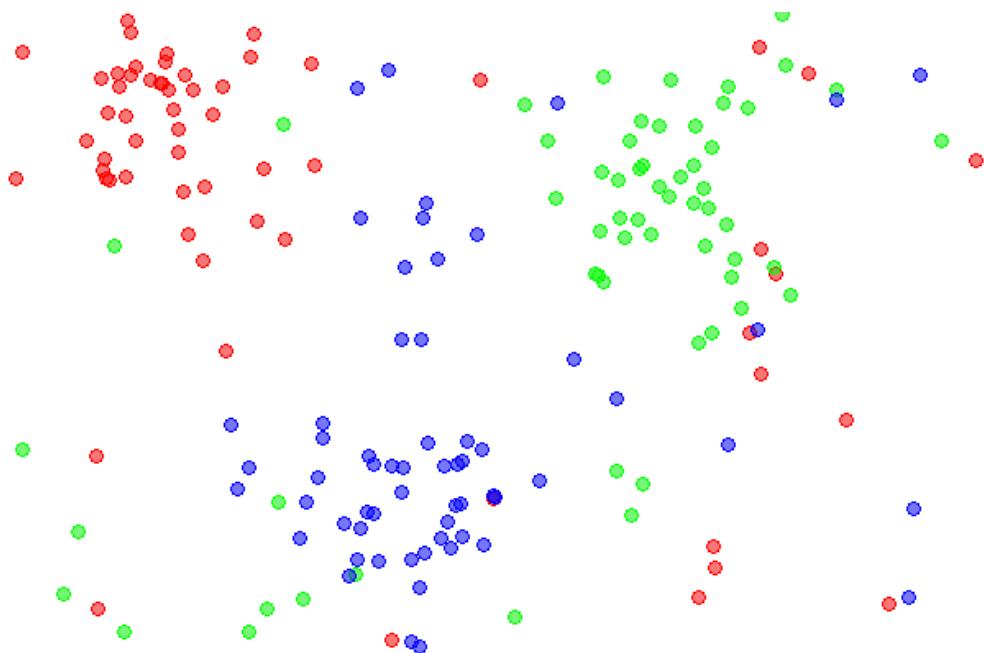
kNearestNeighbors (kNN)



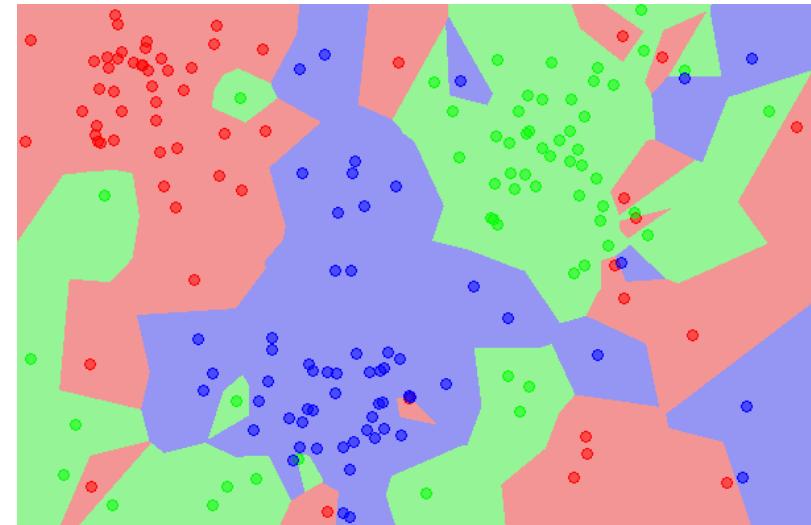
For each test point, \mathbf{x} find the k -nearest instances in the training data

Classify the point according to the majority vote of their class labels

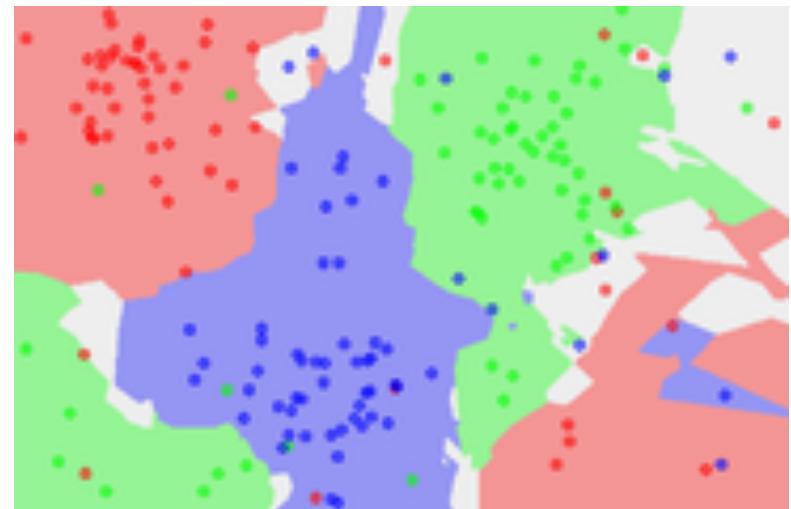
kNearestNeighbors (kNN)



Dataset



$k=1$

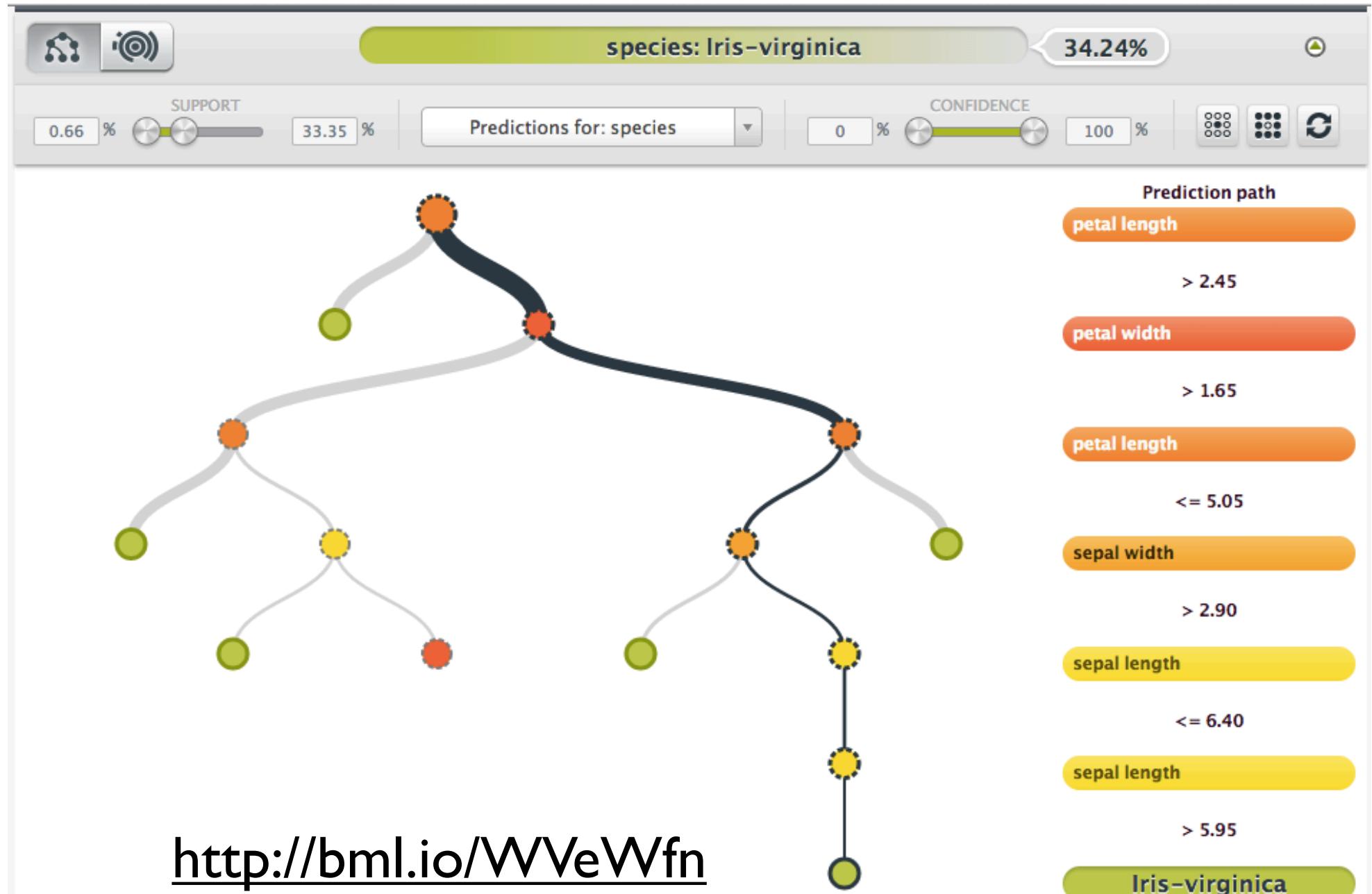


`neighbors.KNeighborsClassifier`

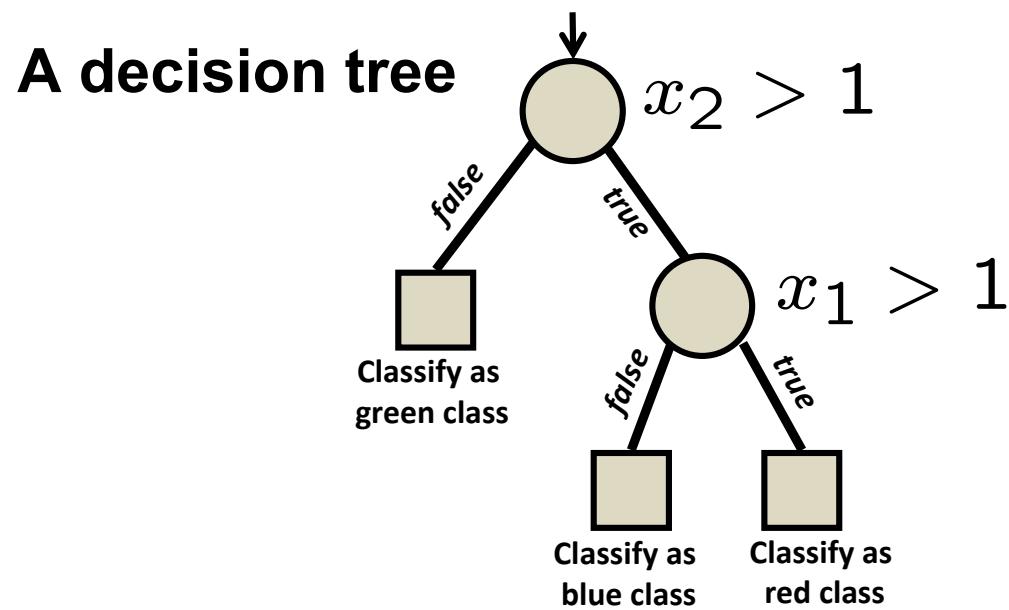
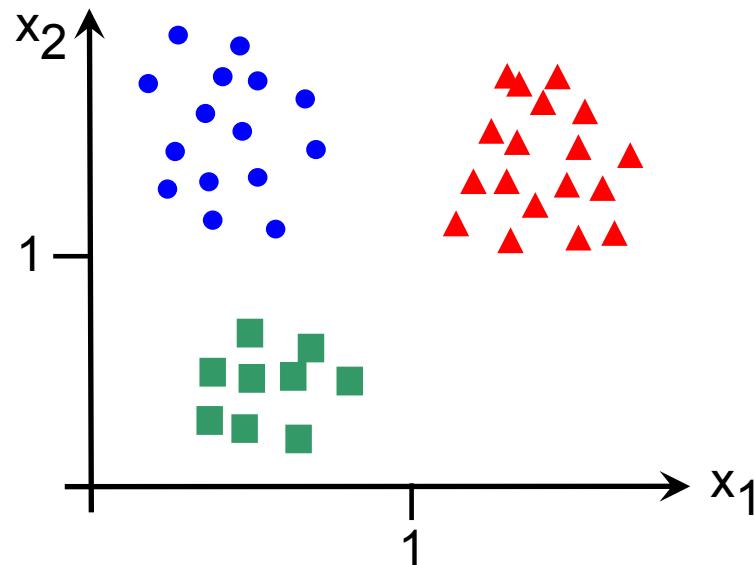
$k=5$

Decision Trees

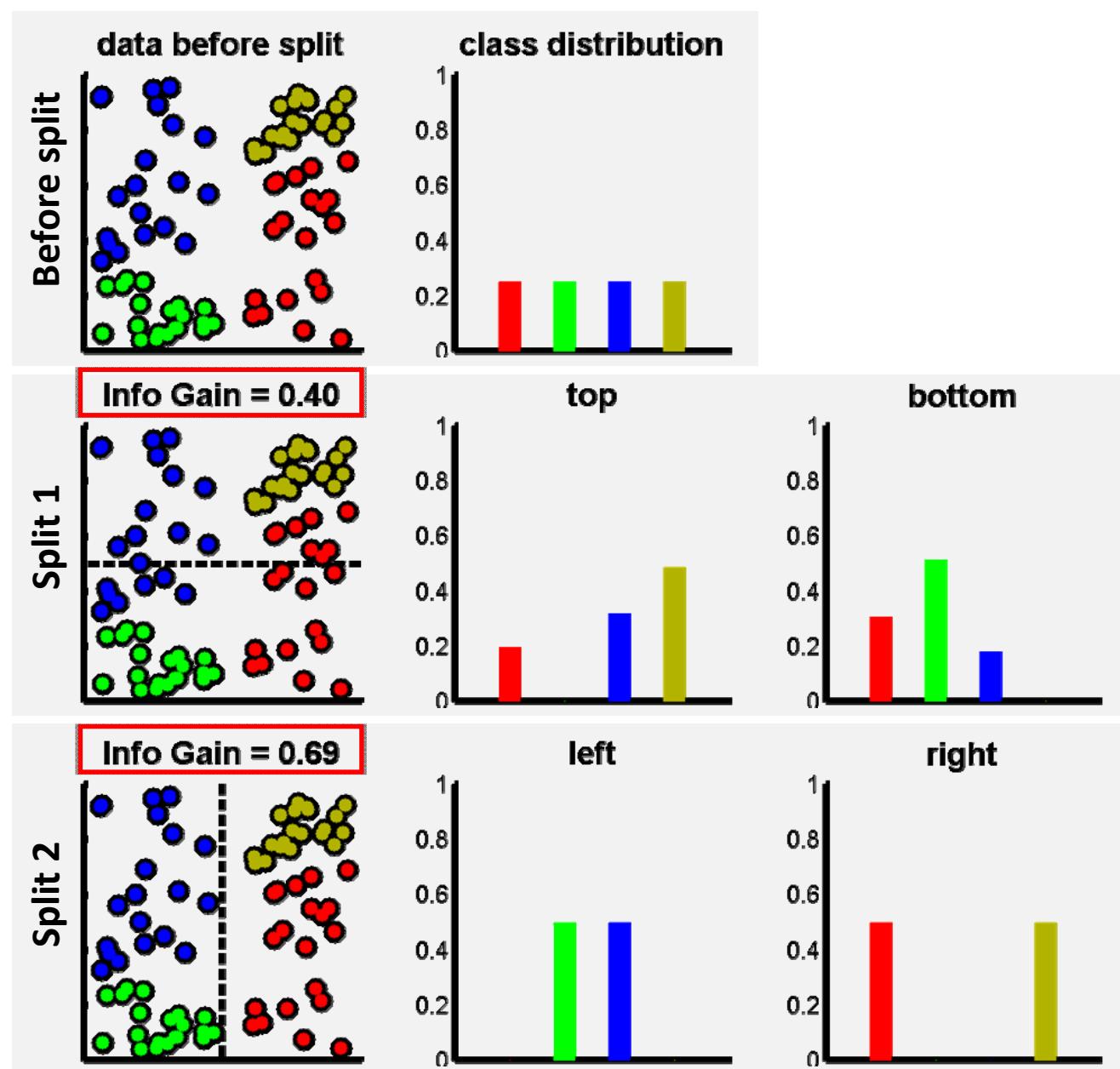
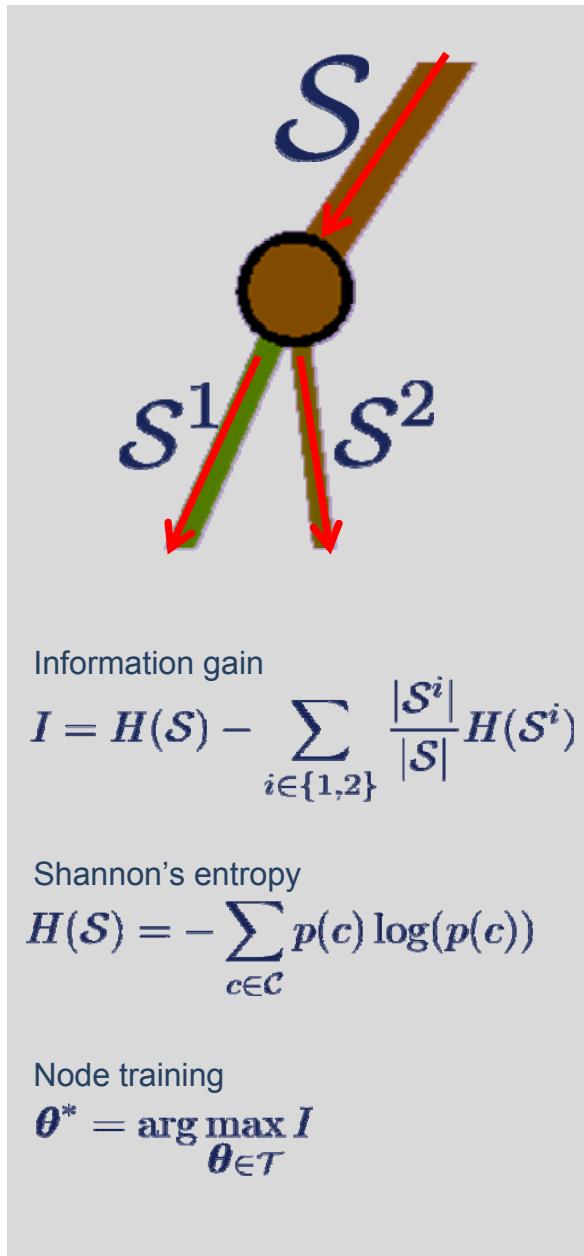
Classification and regression trees (CART)



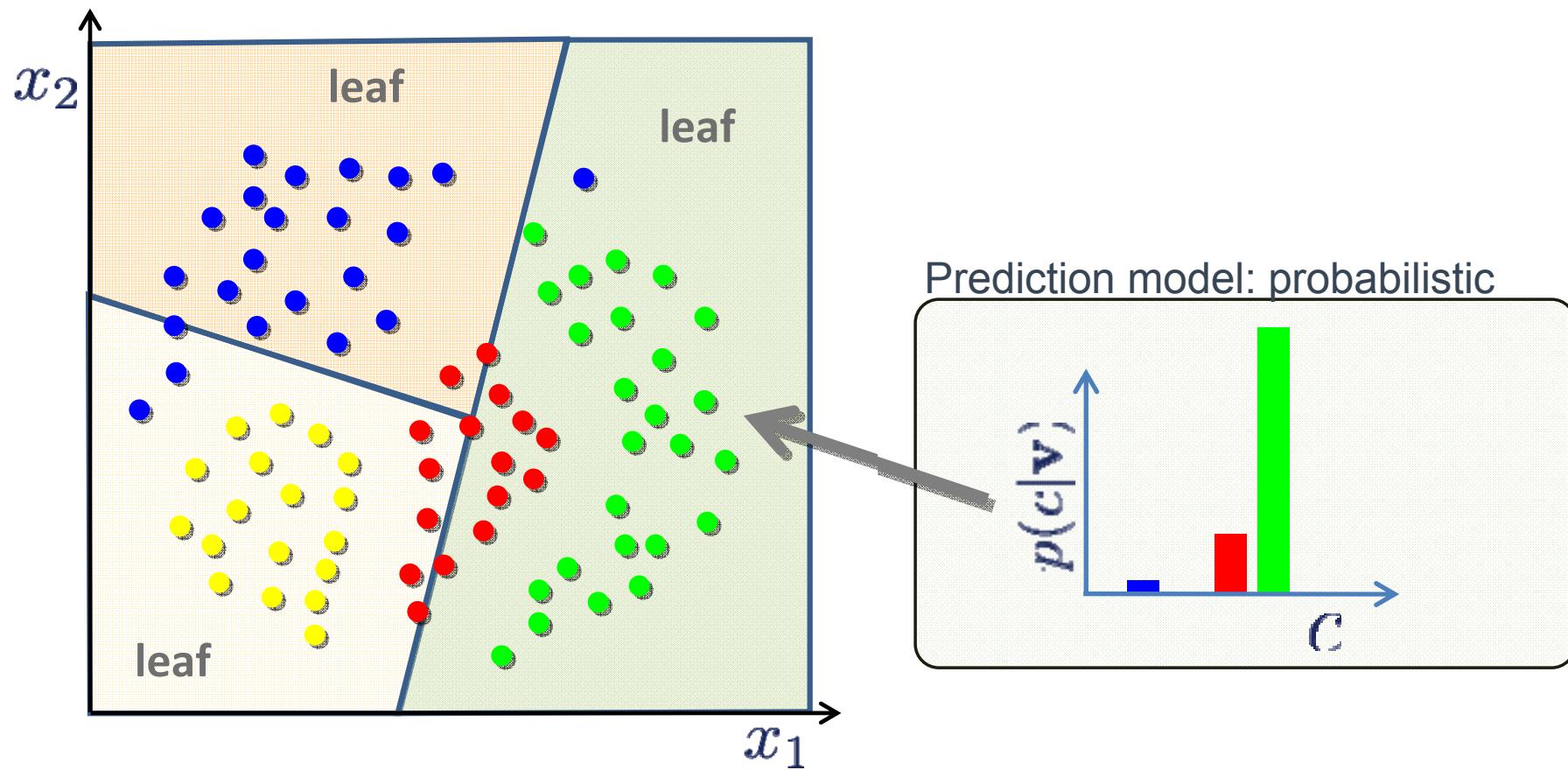
<http://bml.io/WVeWfn>



Building Trees Rigorously (Node Splitting Criteria)



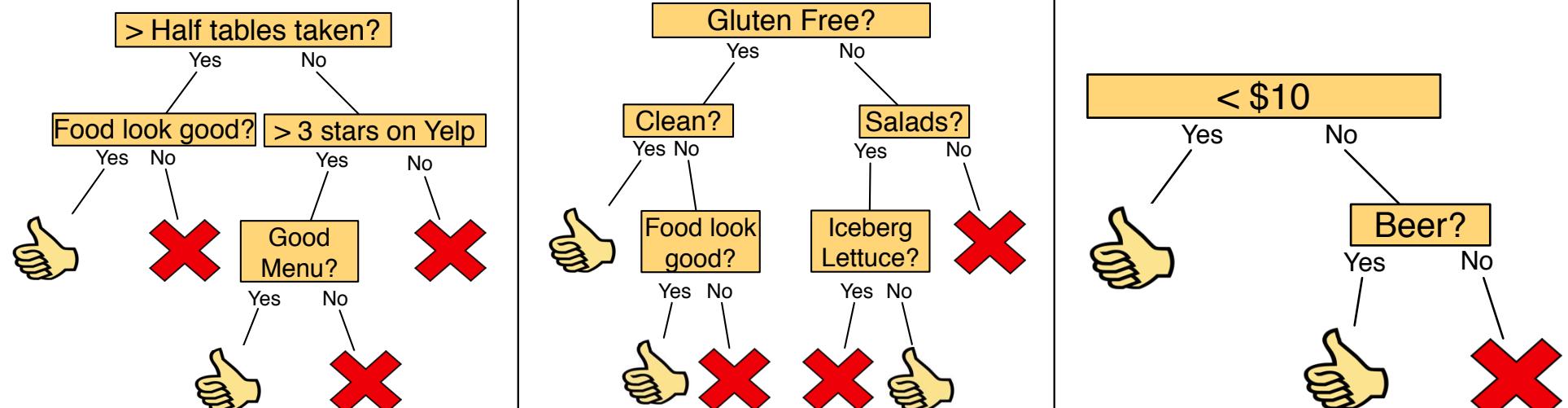
Making predictions from the Learned Tree



<http://cs.stanford.edu/people/karpathy/svmjs/demo/>

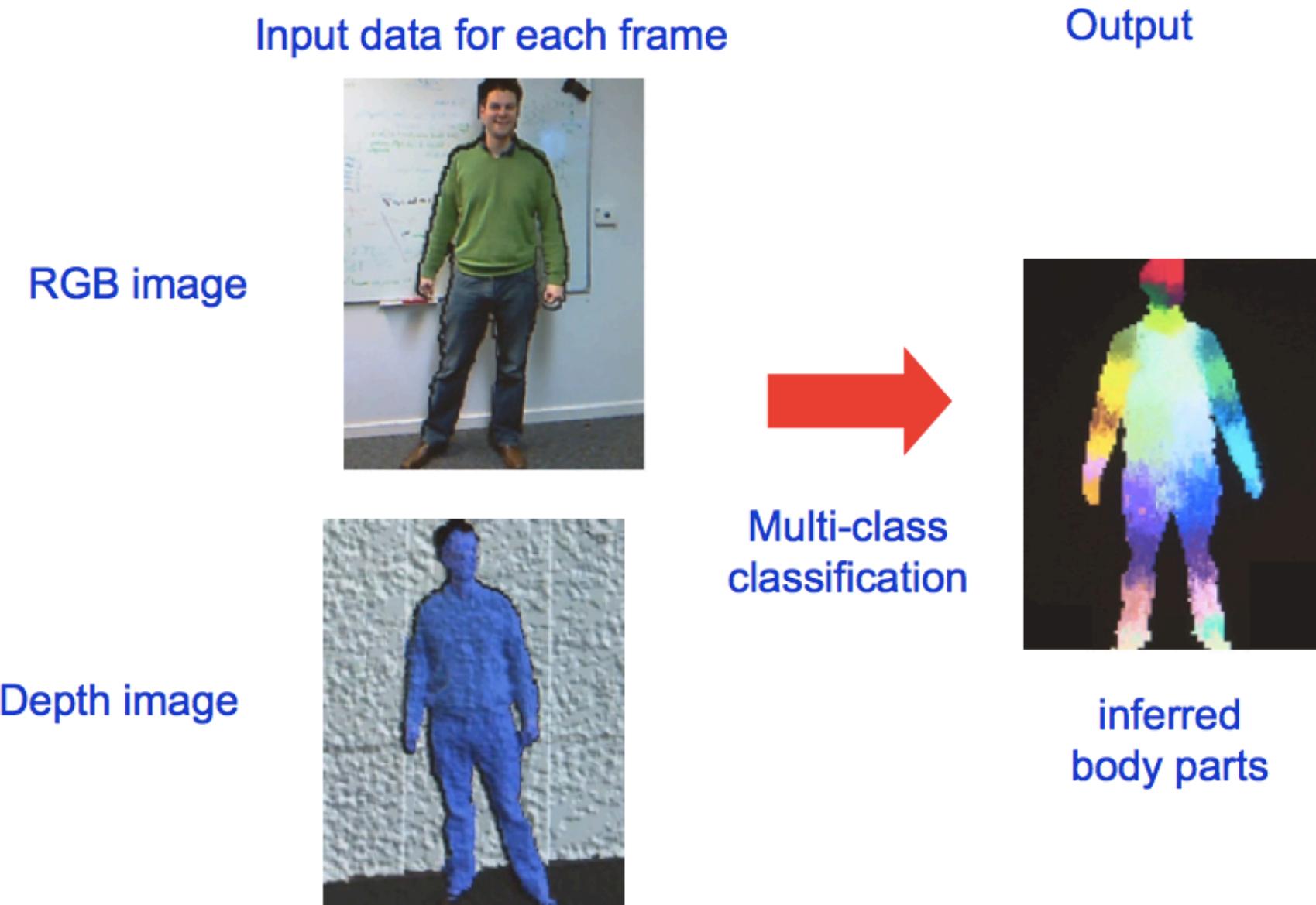
Collections of Trees (“Random Forests”)

ensembles generally increase robustness



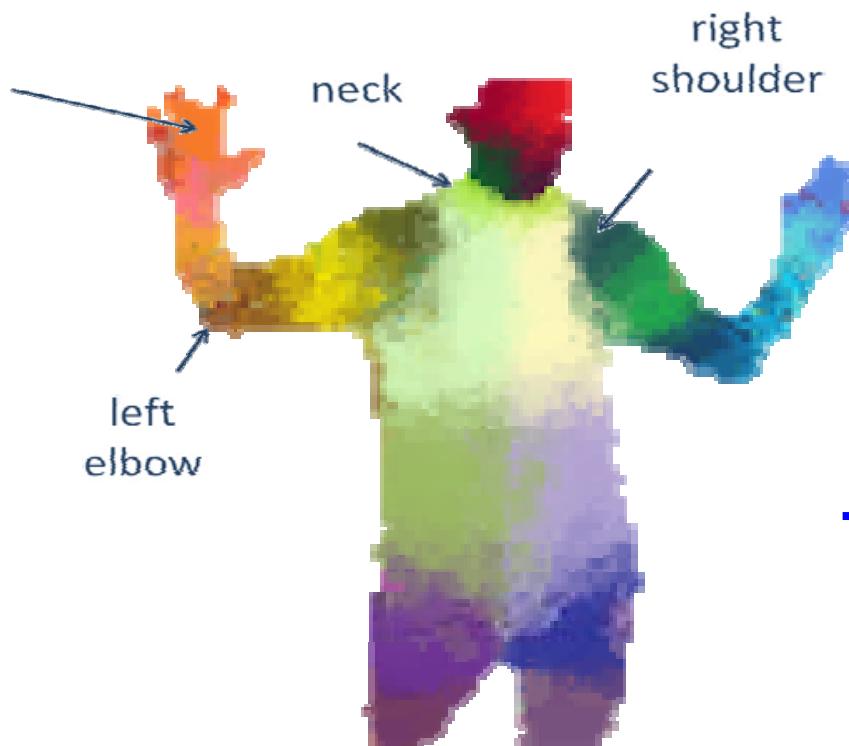
D. Eads

Kinect – random forest classifier



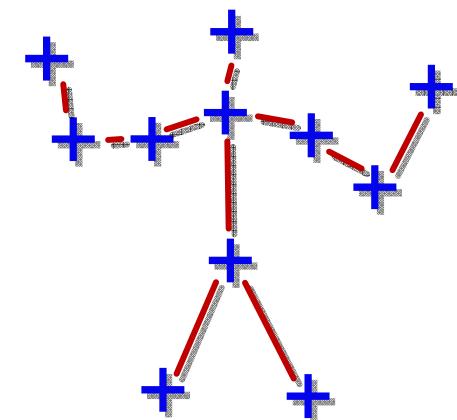
Goal: train a random forest classifier to predict body parts

left hand



right
shoulder

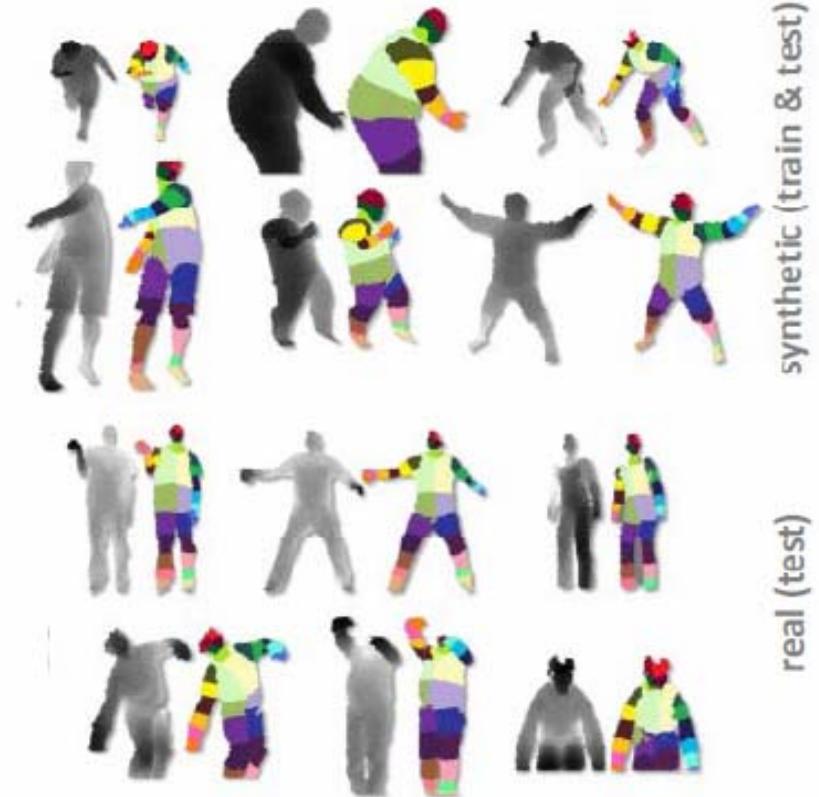
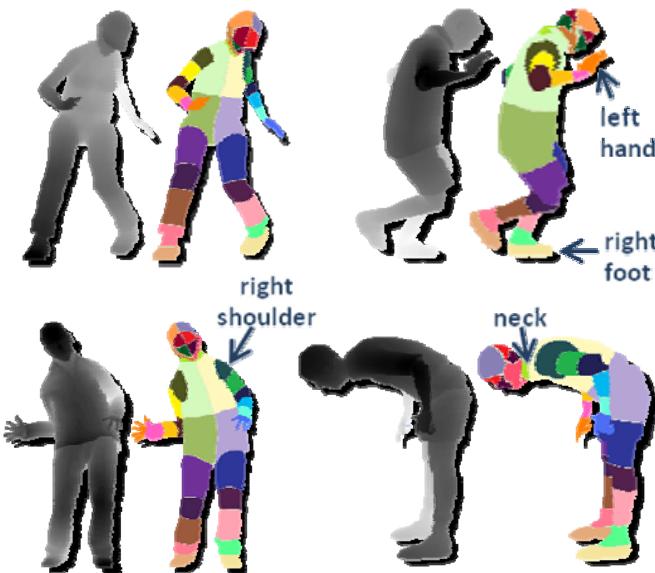
left
elbow



fit stickman model and
track skeleton

Training/test Data

From motion capture system



e.g. 1 million training examples

Example result



Input depth image (bg removed)



Inferred body parts (posterior)



body parts in 3D

Error Estimation & Model Selection

Q: How do I choose which model and parameters to use?

KNN with what # of neighbors?

SVM which what kernel & bandwidth?

RF with how many trees and which mtry?

GP with what kernel & bandwidth?

Solution: use `grid_search.GridSearchCV`

```
grid_search.GridSearchCV(estimator, param_grid, loss_func, n_jobs, cv=None)
```

Computes cv -fold cross-validated `loss_func` (or `score_func`) of estimator over a `param_grid` on `n_jobs` cores, and returns the best model!

Error Estimation & Model Selection

Q: What evaluation metrics are available?

Loss Functions

`metrics.zero_one`(y_true, y_pred)

Zero-One classification loss

`metrics.hinge_loss`(y_true, pred_decision[, ...])

Cumulated hinge loss (non-regularized).

`metrics.mean_square_error`(y_true, y_pred)

Mean square error regression loss

Or write your own!!

Score Functions

`metrics.zero_one_score`(y_true, y_pred)

Zero-One classification score

`metrics.auc`(x, y)

Compute Area Under the Curve (AUC)

`metrics.precision_score`(y_true, y_pred[, ...])

Compute the precision

`metrics.recall_score`(y_true, y_pred[, pos_label])

Compute the recall

`metrics.fbeta_score`(y_true, y_pred, beta[, ...])

Compute fbeta score

`metrics.f1_score`(y_true, y_pred[, pos_label])

Compute f1 score

Evaluation Plots

`metrics.confusion_matrix`(y_true, y_pred[, ...]) Compute confusion matrix to evaluate the accuracy of a classification

`metrics.roc_curve`(y_true, y_score) Compute Receiver operating characteristic (ROC)

`metrics.precision_recall_curve`(y_true, ...) Compute precision-recall pairs for different probability thresholds

To the Notebook!



Unsupervised Learning

Clustering: K-means, Hierarchical Clustering, Mixture Models, Affinity Propagation, Spectral Clustering and lots of evaluation metrics!

Manifold Learning: Isomap, Local Linear Embedding, Hessian Eigenmap, Local Tangent Space Alignment

Matrix Factorization: PCA, Kernel PCA, Sparse PCA, ICA, NMF, Dictionary Learning

Outlier Detection: Elliptic envelope, One-class SVM

Covariance Estimation: Sparse, Shrunk, and Robust estimators

learn is missing a few things...

- Kernel smoothing / Loess
- Multivariate Adaptive Regression Splines (MARS)
- Boosting
- Multidimensional scaling (MDS)
- Neural Networks / Self-Organizing Maps
- Kalman Filtering / Particle Filtering
- Hidden Markov Models (not maintained)
- Missing data imputation / surrogate splitting
- Bayesian model fitting / non-parametrics

Breakout #2

Classify the famous **Iris** data, first used by R.A Fisher.

To load in the data and split into training / testing sets:

```
iris = datasets.load_iris()  
Xtr = iris.data[::2,:]  
Xte = iris.data[1::2,:]  
Ytr = iris.target[::2]  
Yte = iris.target[1::2]
```

- - 1. Choose your favorite classification model.
 - 2. Find the parameters that minimize the 3-fold cross-validation misclassification rate over the training set.
 - 3. Apply this optimized model to predict the class of each object in the held-out set.
- a) What is your best 3-fold CV 0-1 score?
- b) What is your 0-1 score when applying it to testing set?

http://nbviewer.ipython.org/urls/raw.github.com/amueller/tutorial_ml_gkbionics/master/5%2520-%2520k%2520Nearest%2520Neighbors.ipynb