Administrivia

- Proposal 1: no change
- Proposal 2: drop mini-project 3
  - ✓ More time for mini-project 2 and final project
  - ✓ Proposed redistribution: (45% mp + 30% fp) → (50% mp + 25% fp)
    - ✓ Final project work ~ 1 mini project x group size
  - ✓ Mini project 2 due on ~~April 07~~ April 14
  - ✓ Project proposal due on ~~April 02~~ April 07
  - ✓ Downside: no programming assignment for unsupervised learning
    - ✓ Extra credits in the next three weekly homework (10,11,12)

- Solutions to mini-project 1 posted on Moodle
  - Ask your TA if you have any questions regarding grading

Ensemble methods

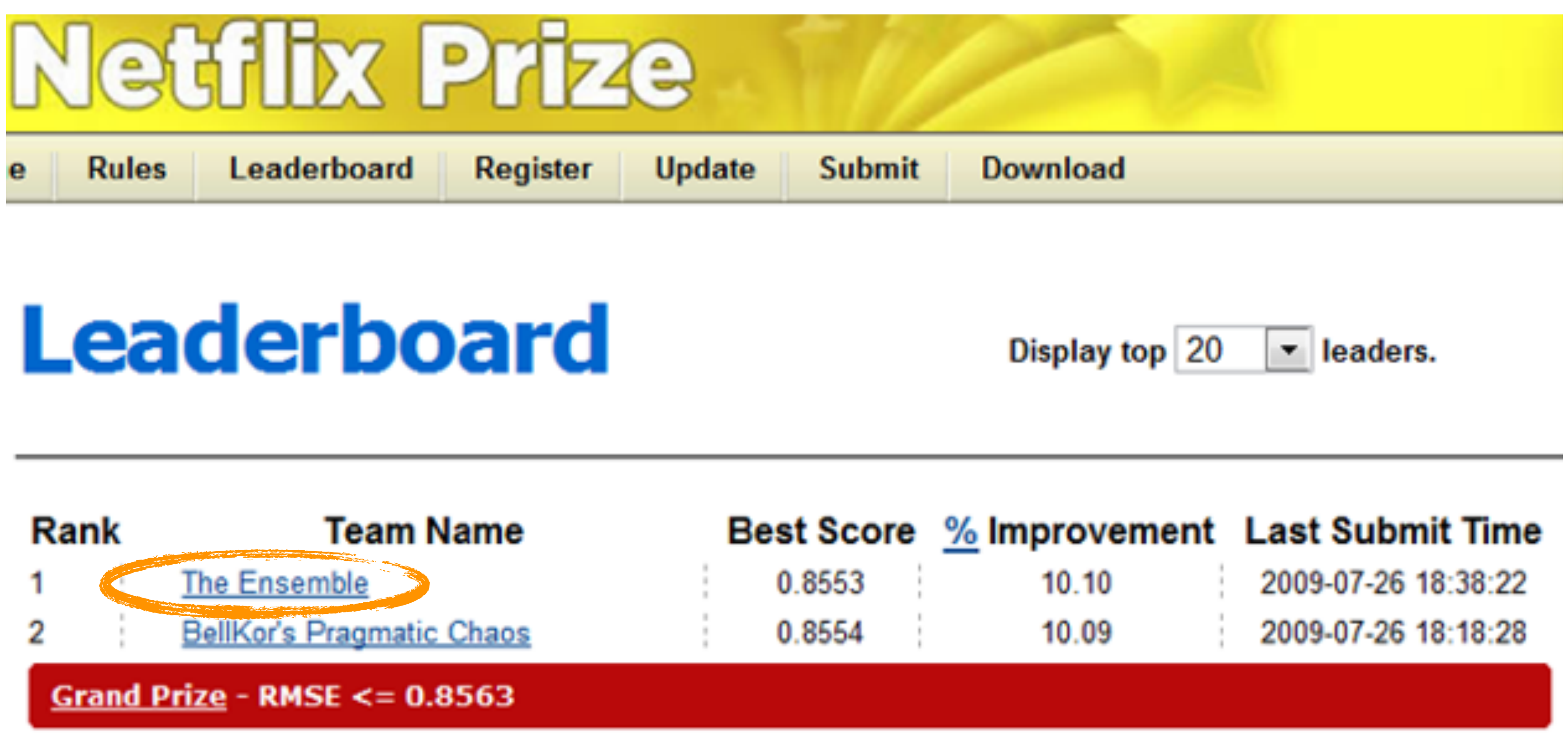
Subhransu Maji

CMPSCI 689: Machine Learning

31 March 2015

Ensembles

- Wisdom of the crowd: groups of people can often make better decisions than individuals
- Today's lecture:
  - Ways to combine base learners into ensembles
  - We might be able to use simple learning algorithms
  - Inherent parallelism in training
  - Boosting — a method that takes classifiers that are only slightly better than chance and learns an arbitrarily good classifier

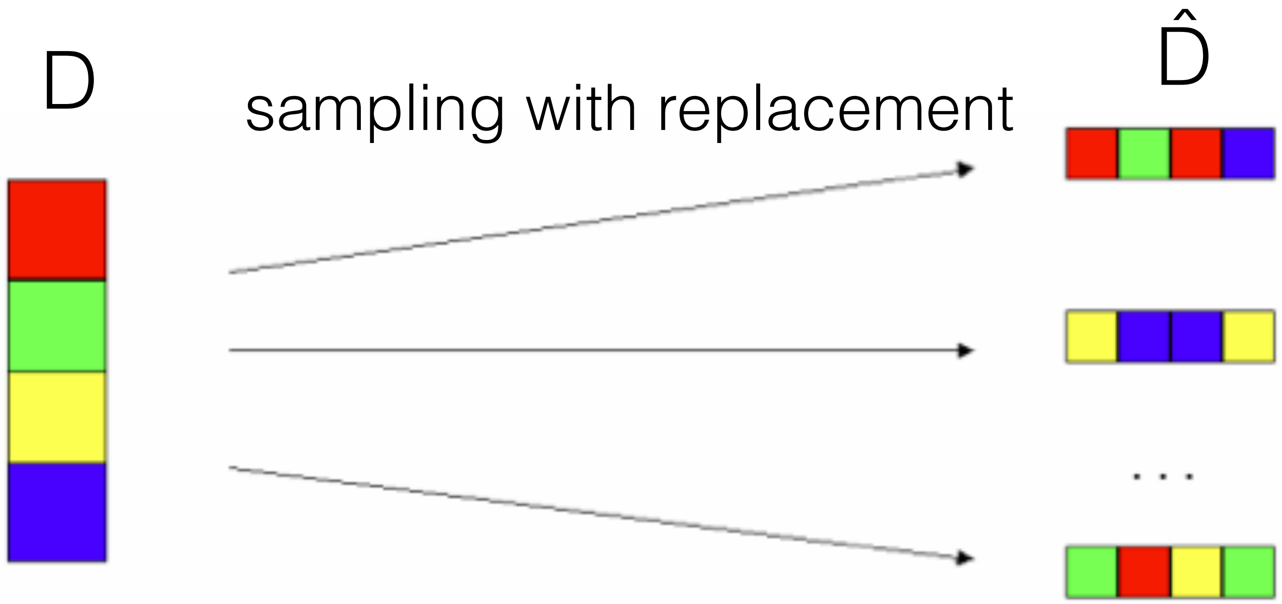


Voting multiple classifiers

- Most of the learning algorithms we saw so far are deterministic
  - If you train a decision tree multiple times on the same dataset, you will get the same tree
- Two ways of getting multiple classifiers:
  - Change the learning algorithm
    - Given a dataset (say, for classification)
    - Train several classifiers: decision tree, kNN, logistic regression, multiple neural networks with different architectures, etc
    - Call these classifiers $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})$
    - Take majority of predictions $\hat{y} = \text{majority}(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x}))$
    - For regression use mean or median of the predictions
    - For ranking and collective classification use some form of averaging
  - Change the dataset
    - How do we get multiple datasets?

Bagging

- Option: split the data into K pieces and train a classifier on each
  - A drawback is that each classifier is likely to perform poorly
- Bootstrap resampling is a better alternative
  - Given a dataset D sampled i.i.d from a unknown distribution $\mathcal{D}$, and we get a new dataset $\hat{D}$ by random sampling with replacement from D, then $\hat{D}$ is also an i.i.d sample from $\mathcal{D}$



There will be repetitions

Probability that the first point will not be selected:

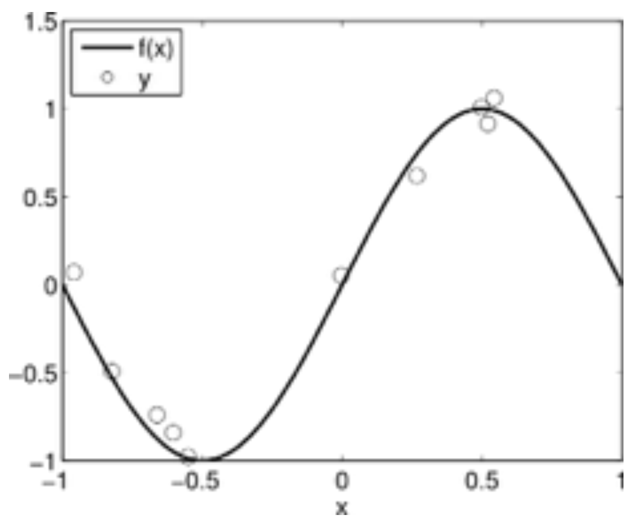
$$\left(1 - \frac{1}{N}\right)^N \longrightarrow \frac{1}{e} \sim 0.3679$$

Roughly only **63%** of the original data will be contained in any bootstrap

- Bootstrap aggregation (bagging) of classifiers [Breiman 94]
  - Obtain datasets $D_1, D_2, \ldots, D_N$ using bootstrap resampling from D
  - Train classifiers on each dataset and average their predictions

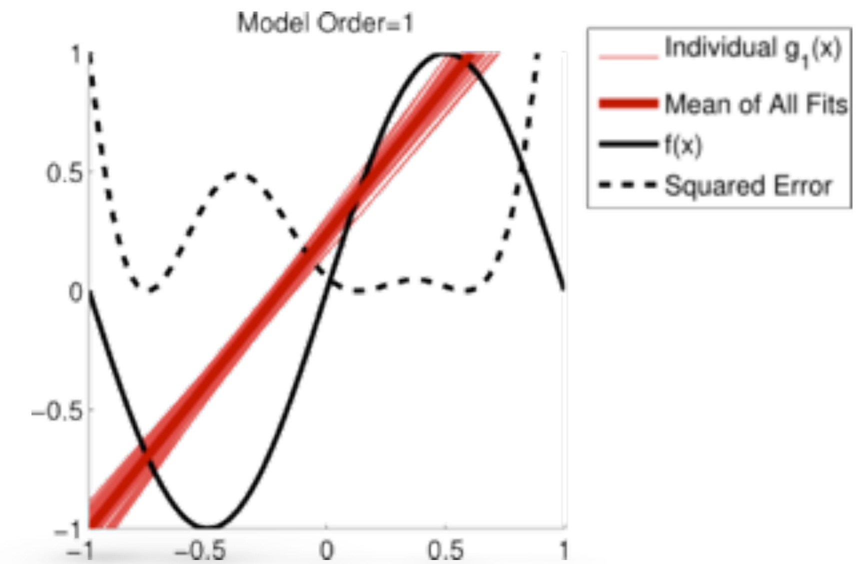
Why does averaging work?

- Averaging reduces the variance of estimators
- Recall the bias-variance tradeoff — error = bias^2 + variance + noise

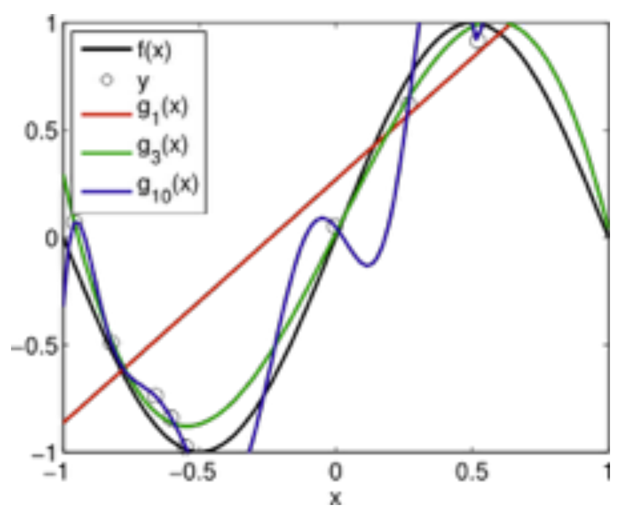


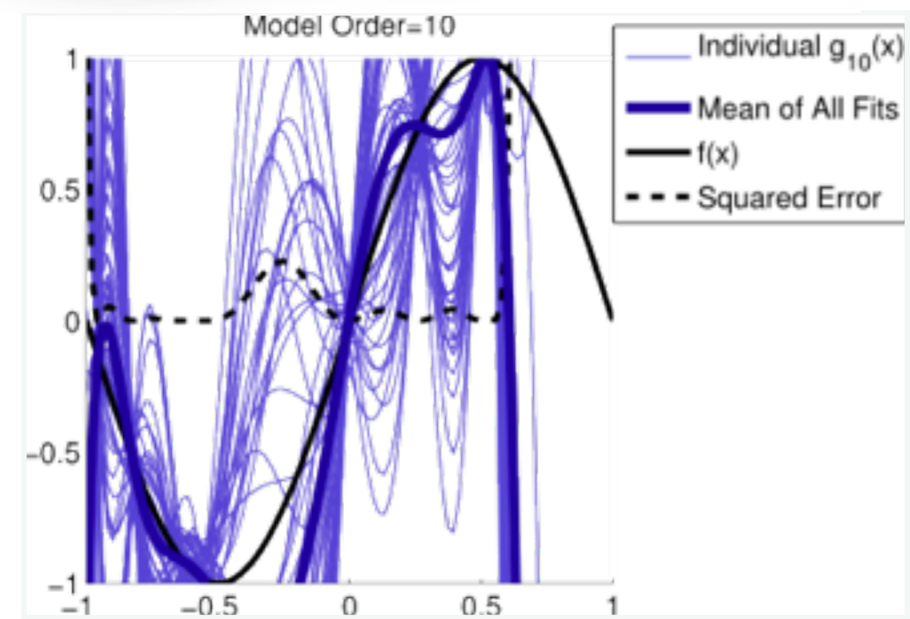
$$y = f(x) + \epsilon$$
$$f(x) = \sin(\pi x)$$
$$\epsilon = N(0, \sigma^2)$$
$$\sigma = 0.1$$

50 samples →



$$g_n(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_n x^n$$





- Averaging is a form of regularization: each model can individually overfit but the average is able to overcome the overfitting

Boosting weak learners

- Bagging reduces variance but has little impact on bias
- Boosting reduces bias — it takes a poor learning algorithm (weak learner) and turns it into a good learning algorithm (strong learner)
- We will discuss a practical learning algorithm called AdaBoost, short for adaptive boosting — one of the first practical boosting algorithm
  - Proposed by Freund & Schapire'95 — ideas originated in the theoretical machine learning community
  - It won the Gödel Prize in 2003

- Intuition behind AdaBoost: study for an exam by taking past exams
  1. Take the exam
  2. Pay less attention to questions you got right
  3. Pay more attention to questions you got wrong
  4. Study more, and go to step 1

AdaBoost algorithm

Given a weak learner $\mathcal{W}$

Algorithm 31 ADABOOST($\mathcal{W}$, $\mathcal{D}$, K)

$d^{(0)} \leftarrow \langle \frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N} \rangle$ // Initialize uniform importance to each example

for $k = 1 \ldots K$ **do**

  $f^{(k)} \leftarrow \mathcal{W}(\mathcal{D}, d^{(k-1)})$ // Train kth classifier on weighted data

  $\hat{y}_n \leftarrow f^{(k)}(x_n), \forall n$ // Make predictions on training data

  $\hat{\epsilon}^{(k)} \leftarrow \sum_n d_n^{(k-1)} [y_n \neq \hat{y}_n]$ // Compute weighted training error

  $\alpha^{(k)} \leftarrow \frac{1}{2} \log \left(\frac{1 - \hat{\epsilon}^{(k)}}{\hat{\epsilon}^{(k)}} \right)$ // Compute "adaptive" parameter

  $d_n^{(k)} \leftarrow \frac{1}{Z} d_n^{(k-1)} \exp[-\alpha^{(k)} y_n \hat{y}_n], \forall n$ // Re-weight examples and normalize

end for

return $f(\hat{x}) = \mathrm{sgn}\left[\sum_k \alpha^{(k)} f^{(k)}(\hat{x})\right]$ // Return (weighted) voted classifier

slide credit: ciml book

AdaBoost discussion

- As long as the weak learner $\mathcal{W}$ does better than chance on the weighted classification task $\alpha^{(k)} > 0$:

$$\alpha^{(k)} \leftarrow \frac{1}{2} \log \left(\frac{1 - \hat{\epsilon}^{(k)}}{\hat{\epsilon}^{(k)}} \right)$$

$$\alpha^{(k)} > 0 \text{ if } \mathcal{W} \text{ obtains error } \hat{\epsilon}^{(k)} < 0.5$$

- After each round the misclassified points are up weighted and the correctly classified points are down weighted:

$$d_n^{(k)} \leftarrow \frac{1}{Z} d_n^{(k-1)} \underline{\exp[-\alpha^{(k)} y_n \hat{y}_n]} > 1 \text{ if } y_n \neq \hat{y}_n$$

AdaBoost discussion

- Why this particular form of the weight function?

$$\alpha^{(k)} \leftarrow \frac{1}{2}\log\left(\frac{1-\hat{\epsilon}^{(k)}}{\hat{\epsilon}^{(k)}}\right) \qquad d_n^{(k)} \leftarrow \frac{1}{Z} d_n^{(k-1)} \exp[-\alpha^{(k)} y_n \hat{y}_n]$$

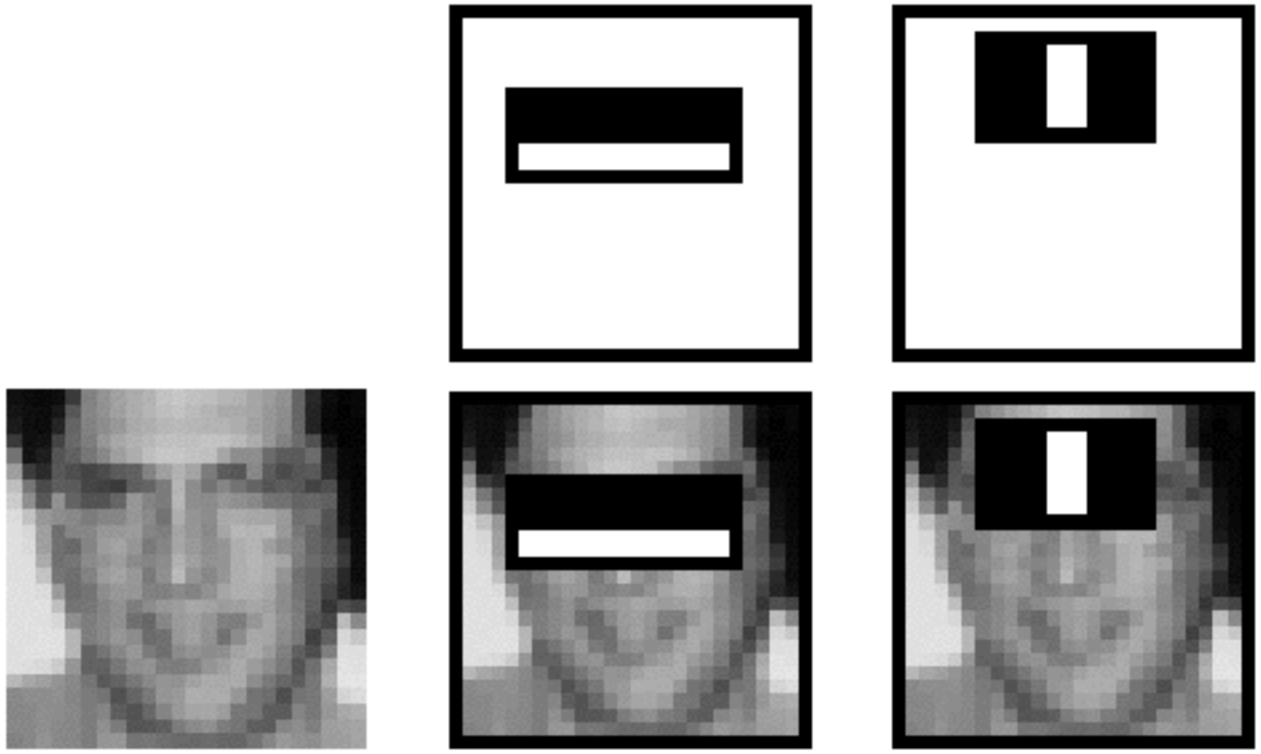
- Consider a dataset with 80 + examples and 20 - examples
  - Initially all the weights are equal
  - Weak learner returns $f^{(1)}(\mathbf{x}) = +1$ in round 1

$$\hat{\epsilon}^{(k)} = 0.2 \qquad \alpha^{(k)} = \frac{1}{2}\log 4$$

  - Positive weights after round 1: exp[-0.5 log 4] = 0.5
  - Negative weights after round 1: exp[0.5 log 4] = 2.0
  - Total weight on positives: 80x0.5 = 40
  - Total weight on negatives: 20x2.0 = 40
  - After the first round the weak learner has to do something non-trivial

AdaBoost in practice

- It is easy to design computationally efficient weak learners
- Example: decision trees of depth 1 (decision stumps)
  - Each weak learner is rather simple — can query only one feature, but by boosting we can obtain a very good classifier
- Application: Face detection [Viola & Jones, 01]
  - Weak classifier: detect light/dark rectangles in an image



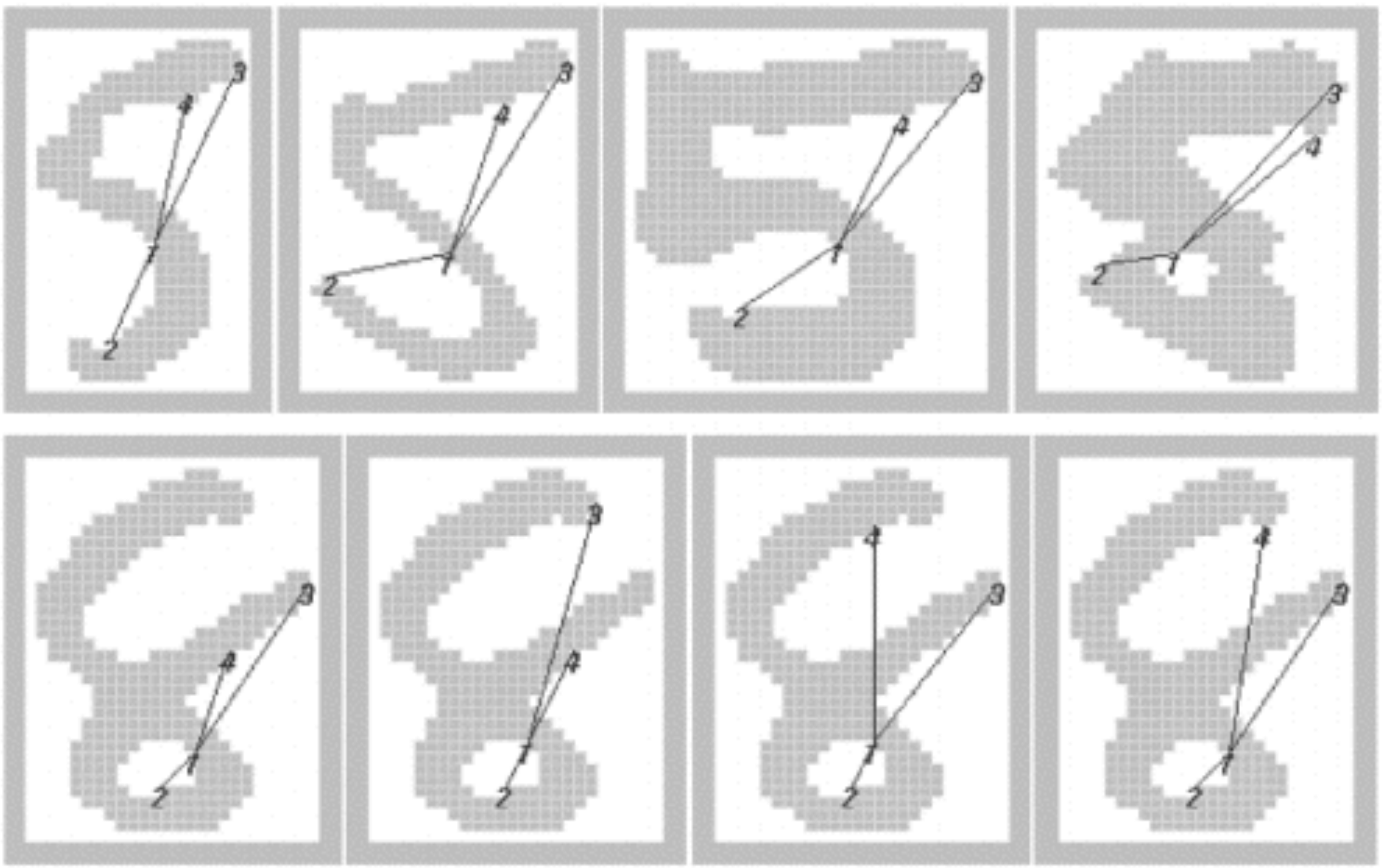
Random ensembles

- One drawback of ensemble learning is that the training time increases
  - For example when training an ensemble of decision trees the expensive step is choosing the splitting criteria
- Random forests are an efficient and surprisingly effective alternative
  - Choose trees with a fixed structure and random features
    - Instead of finding the best feature for splitting at each node, choose a random subset of size **k** and pick the best among these
    - Train decision trees of depth **d**
    - Average results from multiple randomly trained trees
  - When k=1, no training is involved — only need to record the values at the leaf nodes which is significantly faster
- Random forests tends to work better than bagging decision trees because bagging tends produce highly correlated trees — a good feature is likely to be used in all samples

Random forests in action: MNIST

- Early proponents of random forests: “Joint Induction of Shape Features and Tree Classifiers”, Amit, Geman and Wilder, PAMI 1997

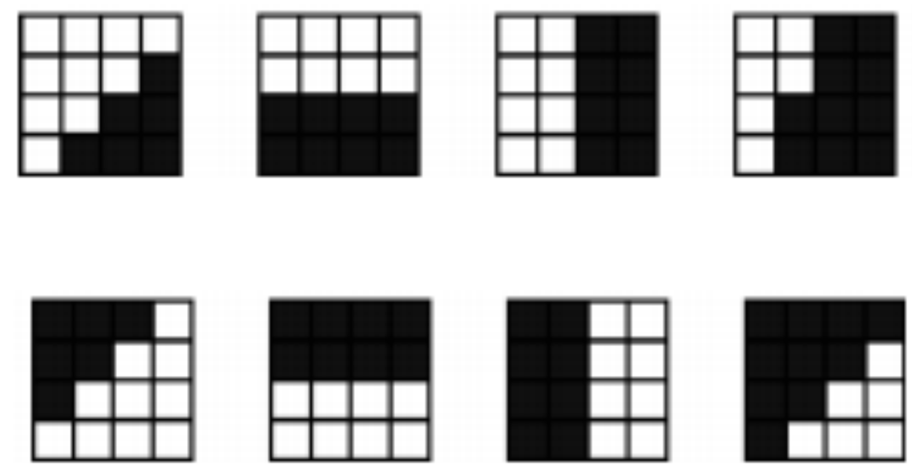
Features: arrangement of tags



Arrangements: 8 angles

tags

Common 4x4 patterns



A subset of all the 62 tags

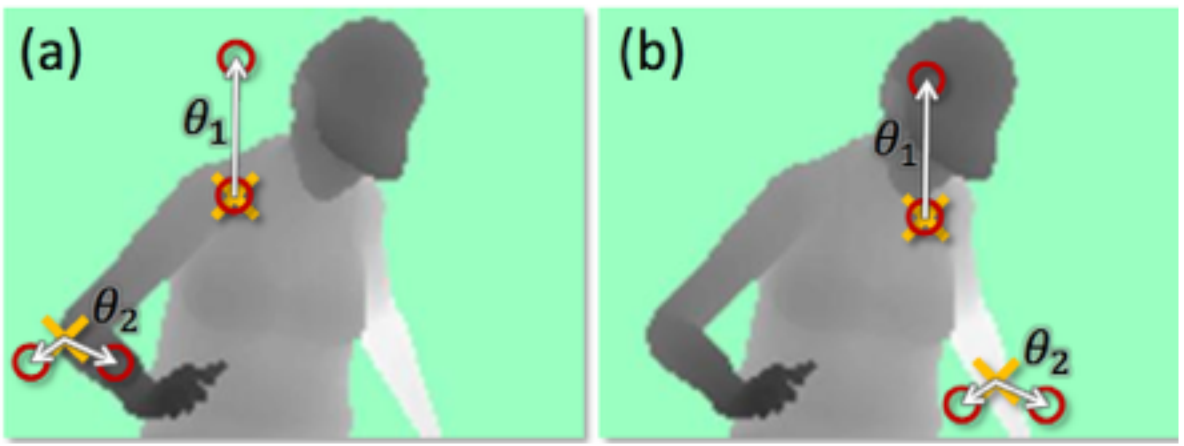
#Features: 62x62x8 = 30,752

Single tree: **7.0%** error

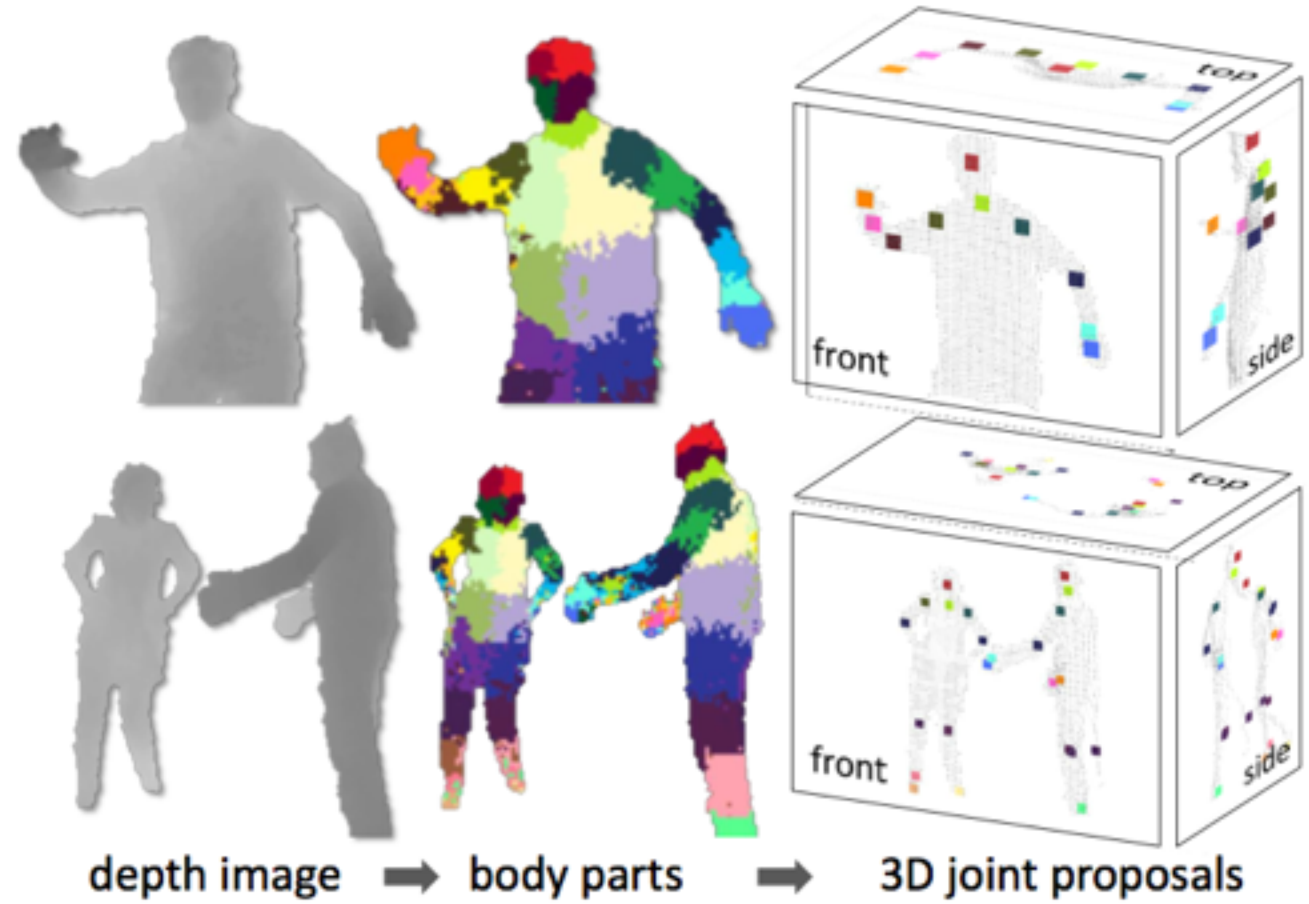
Random forest of 25 trees: **0.8%** error

Random forests in action: Kinect pose

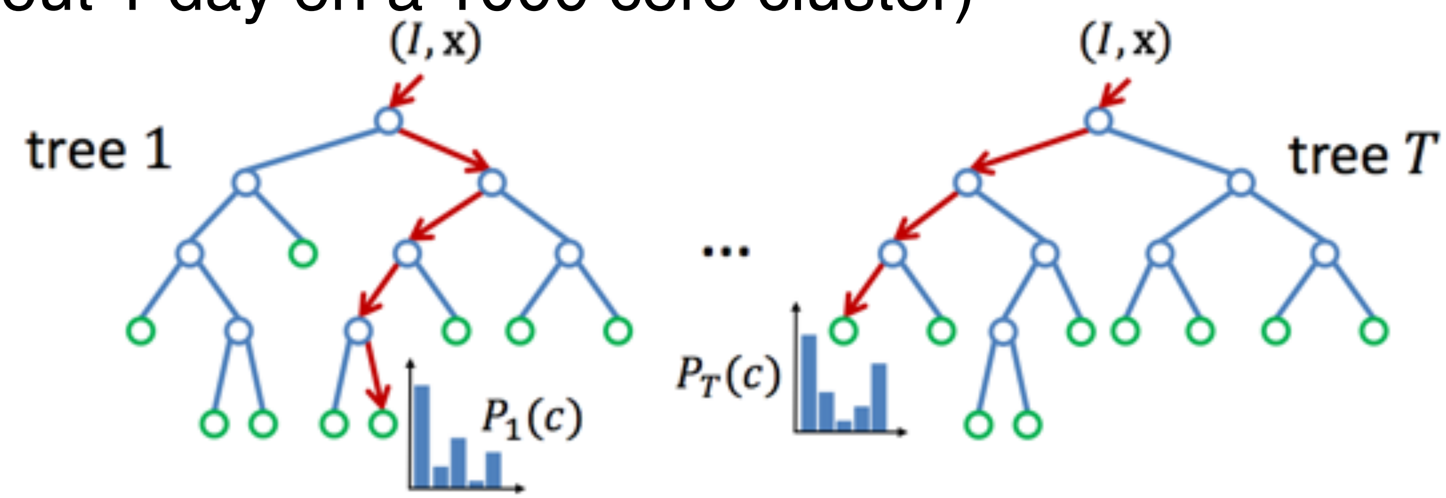
- Human pose estimation from depth in the Kinect sensor [Shotton et al. CVPR 11]



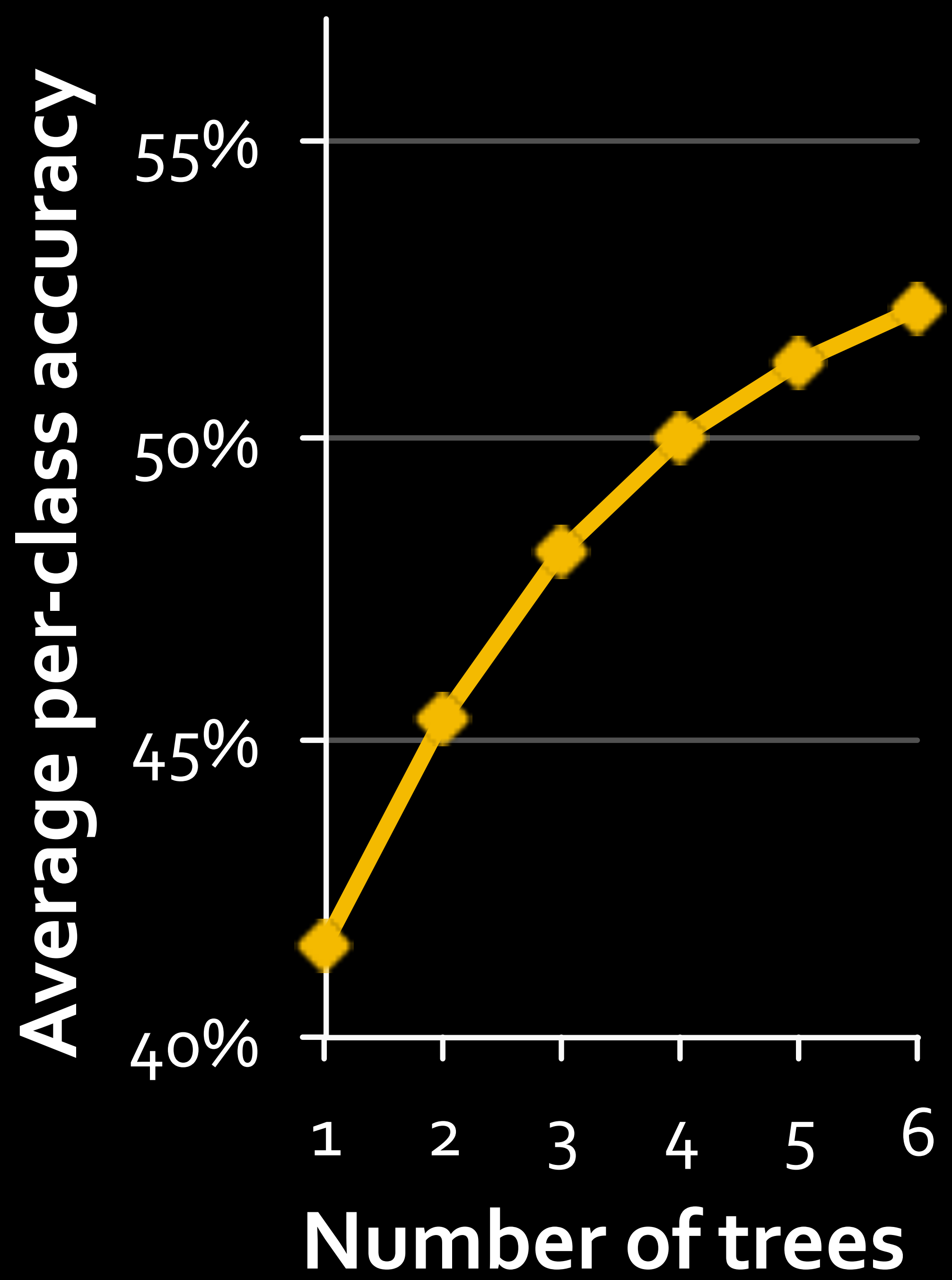
$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$



Training: 3 trees, 20 deep, 300k training images per tree, 2000 training example pixels per image, 2000 candidate features θ, and 50 candidate thresholds τ per feature (Takes about 1 day on a 1000 core cluster)



Number of trees



ground truth

inferred body parts (most likely)

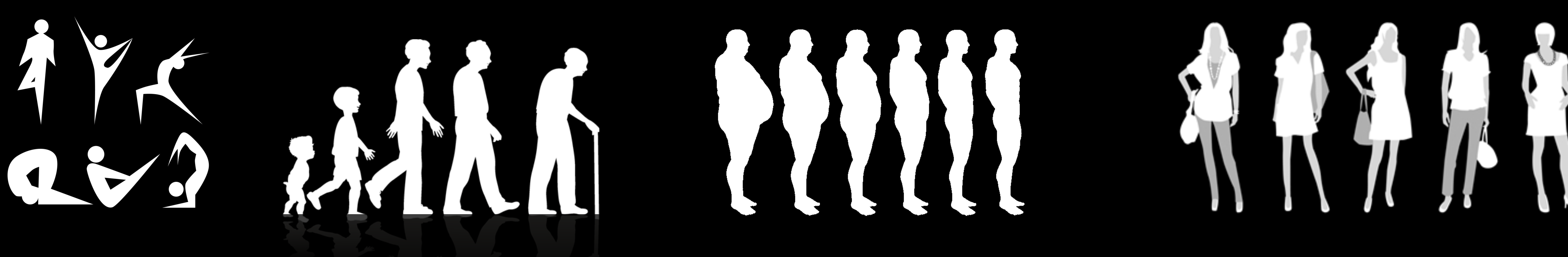
Synthetic training data

Record mocap
500k frames
distilled to 100k poses

Retarget to several models

Render (depth, body parts) pairs

Train invariance to:



Summary

- Ensembles improve prediction by reducing the variance
- Two ways of creating ensembles
  - Vary the learning algorithm
    - Training algorithms: decision trees, kNN, perceptron
    - Hyperparameters: number of layers in a neural network
    - Randomness in training: initialization, random subset of features
  - Vary the training data
    - Bagging: average predictions of classifiers trained on bootstrapped samples of the original training data
- Boosting combines weak learners to make a strong learner
  - Reduces bias of the weak learners
- Ensembles of randomly trained decision trees are efficient and effective for many problems

Slides credit

- Some of the slides are based on CIML book by Hal Daume III
- Bias-variance figures — https://theclevermachine.wordpress.com/tag/estimator-variance/
- Figures for random forest classifier on MNIST dataset — Amit, Geman and Wilder, PAMI 1997 — http://www.cs.berkeley.edu/~malik/cs294/amitgemanwilder97.pdf
- Figures for Kinect pose — “Real-Time Human Pose Recognition in Parts from Single Depth Images”, J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, R. Moore, A. Kipman, A. Blake, CVPR 2011