Dall-E

Ensembles and Forests

Applied Machine Learning
Derek Hoiem

Previously…

- We’ve learned how to build and apply single models
  - Nearest neighbor
  - Logistic regression
  - Linear regression
  - Trees

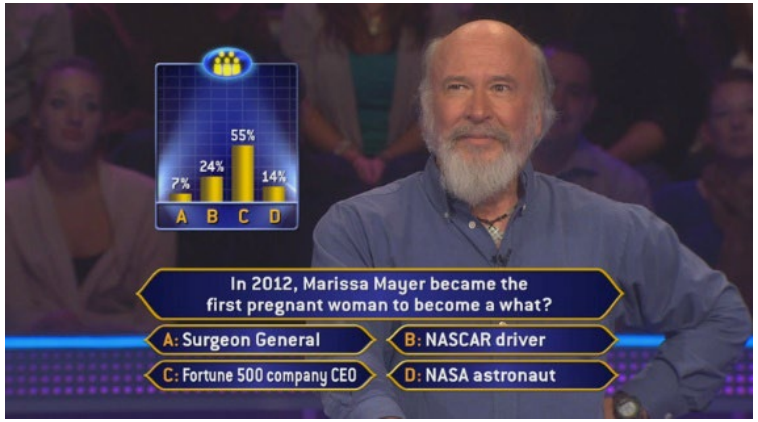
Ensemble Models

- An **ensemble** averages or sums predictions from multiple models

- Remember "Who Wants to be a Millionaire"?
  - "Poll the audience" vs "Call a friend"

- Averaging multiple "weak" predictions is often more accurate than any single predictor
  - e.g. audience success rate is 92% vs 66% for the friend

- Models can be constructed independently by sampling, or by incrementally training model to fix previous model's mistakes
  - Averaging independent predictions reduces variance
  - Incrementally fixing mistakes reduces bias



Bias-Variance Trade-off

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x}) - y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2}$$

Variance: due to limited data
Different training samples will give different models that vary in predictions for the same test sample

“**Noise**”: irreducible error due to data/problem

Bias: expected error when optimal model is learned from infinite data

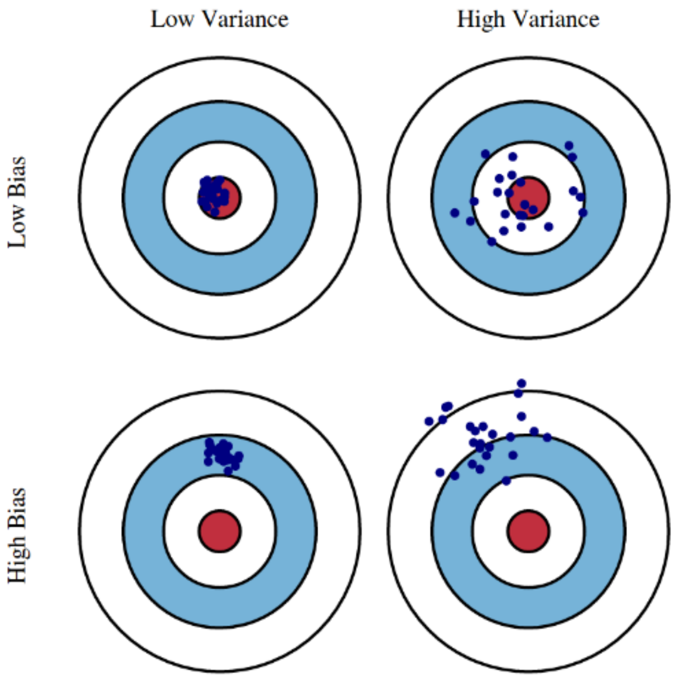
Above is for regression.
But same “expected error = variance + noise + bias^2“ holds for classification error and logistic regression.

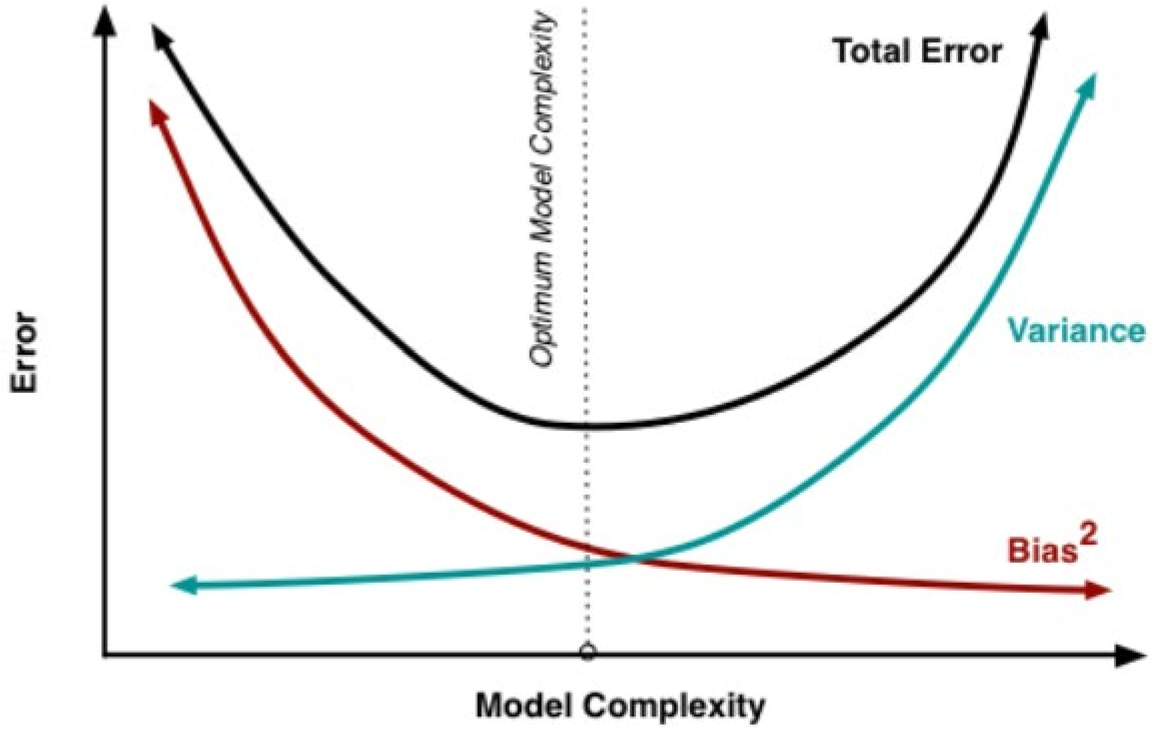
See this for derivation

Fig Sources

Bias-Variance Trade-off

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x}) - y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2}$$





See this for derivation

Fig Sources

Let’s see how ensembles battle bias and variance

- Bootstrapping
- Bagging
- Boosting (Schapire 1989)
- Adaboost (Schapire 1995)

Bootstrap Estimation

- Repeatedly draw *n* samples from *D*
- For each set of samples, estimate a statistic
- The bootstrap estimate is the mean of the individual estimates
- Used to estimate a statistic (parameter) and its variance

Bagging - Aggregate Bootstrapping

- For i = 1 .. M
  - Draw $n^* < n$ samples from D with replacement
  - Learn classifier C_i
- Final classifier is a vote of C_1 .. C_M
- Increases classifier stability / reduces variance

Random Forests

Train a collection of trees (e.g. 100 trees).

For each:

1. Randomly sample some fraction of data (e.g. 90%)
2. Randomly sample some number of features
   - For regression: suggest (# features) /3
   - For classification: suggest sqrt(# features) or log_2(# features)
3. Train a tree
4. (Optional: can get validation error on held out data)

Predict: Average the predictions of all trees

Breiman 2001 [pdf]

Adaboost Terms

- Learner = Hypothesis = Classifier
- Weak Learner: classifier that can achieve < 50% training error over any training distribution
- Strong Learner: makes prediction by combining weak learner predictions

Boosting (Schapire 1989)

- Randomly select $n_1 < n$ samples from D without replacement to obtain D_1
  - Train weak learner C_1
- Select $n_2 < n$ samples from D with half of the samples misclassified by C_1 to obtain D_2
  - Train weak learner C_2
- Select all samples from D that C_1 and C_2 disagree on
  - Train weak learner C_3
- Final strong learner is vote of weak learners

Boosting Terminology

- Learner = Hypothesis = Classifier
- *Weak Learner*: classifier that can achieve < 50% training error over any training distribution
- *Strong Learner*: makes prediction by combining weak learner predictions

Adaboost - Adaptive Boosting

- Instead of sampling, re-weight
  - Previous weak learner has only 50% accuracy over new distribution
- Learn “weak classifiers” on the re-weighted samples
- Final classification based on weighted vote of weak classifiers

https://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf (Freund Schapire ‘99)

What does it mean to “weight” your training samples?

- Some examples count more than others toward parameter estimation or learning objective
- E.g., suppose you want to estimate P(x=0 | y=0) for Naïve Bayes

Unweighted

$$\theta_{\{x = 0|y = 0\}} = \sum_{x_n, y_n \in D} \delta(x_n = 0 \text{ and } y_n = 0) / \sum_{x, y \in D} \delta(y_n = 0)$$

Weighted

$$\theta_{w,\{x = 0|y = 0\}} = \sum_{x_n, y_n \in D} w_n \delta(x_n = 0 \text{ and } y_n = 0) / \sum_{x_n, y_n \in D} w_n \delta(y_n = 0)$$

What does it mean to “weight” your training samples?

Estimate P(x=0 | y=0):

w	x	y
0.1	0	0
0.1	0	0
0.2	1	0
0.1	0	0
0.2	1	0
0.2	0	1
0.1	1	1

What does it mean to "weight" your training samples?

Estimate P(x=0 | y=0):

w	x	y
0.1	0	0
0.1	0	0
0.2	1	0
0.1	0	0
0.2	1	0
0.2	0	1
0.1	1	1

Unweighted: $P(x = 0|y = 0) = \frac{1 + 1 + 1}{1 + 1 + 1 + 1 + 1} = \frac{3}{5}$

Weighted: $P(x = 0|y = 0) = \frac{0.1 + 0.1 + 0.1}{0.1 + 0.1 + 0.2 + 0.1 + 0.2} = \frac{3}{7}$

Adaboost with Confidence Weighted Predictions (RealAB)

Real AdaBoost

1. Start with weights $w_i = 1/N$, $i = 1, 2, \dots, N$.
2. Repeat for $m = 1, 2, \dots, M$:
   (a) Fit the classifier to obtain a class probability estimate $p_m(x) = \hat{P}_w(y = 1|x) \in [0, 1]$, using weights w_i on the training data.
   (b) Set $f_m(x) \leftarrow \frac{1}{2} \log p_m(x)/(1 - p_m(x)) \in R$.
   (c) Set $w_i \leftarrow w_i \exp[-y_i f_m(x_i)]$, $i = 1, 2, \dots, N$, and renormalize so that $\sum_i w_i = 1$. $y_i \in \{-1,1\}$
3. Output the classifier $\operatorname{sign}[\sum_{m=1}^{M} f_m(x)]$.

Friedman et al. Additive Logistic Regression: A Statistical View of Boosting (2000) [pdf]

Boosted decision trees

Train

1. Initialize sample weights to uniform
2. For each tree (e.g. 10-100), based on weighted samples:
   a. Train small tree (e.g. depth = 2-4 typically)
   b. Estimate logit prediction at each leaf node
   c. Reweight samples

Predict: sum logit predictions from all trees

ML Method Comparison by Caruana (2006)

Table 3. Normalized scores of each learning algorithm by problem (averaged over eight metrics)

MODEL	CAL	COVT	ADULT	LTR.P1	LTR.P2	MEDIS	SLAC	HS	MG	CALHOUS	COD	BACT	MEAN
BST-DT	PLT	**.938**	.857	**.959**	**.976**	.700	.869	**.933**	.855	**.974**	**.915**	.878*	**.896***
RF	PLT	.876	.930	.897	.941	**.810**	.907*	.884	.883	.937	.903*	.847	.892
BAG-DT	–	.878	.944*	.883	.911	.762	.898*	.856	**.898**	.948	.856	**.926**	.887*
BST-DT	ISO	.922*	.865	.901*	.969	.692*	.878	.927	.845	.965	.912*	.861	.885*
RF	–	.876	.946*	.883	.922	.785	.912*	.871	.891*	.941	.874	.824	.884
BAG-DT	PLT	.873	.931	.877	.920	.752	.885	.863	.884	.944	.865	.912*	.882
RF	ISO	.865	.934	.851	.935	.767*	**.920**	.877	.876	.933	.897*	.821	.880
BAG-DT	ISO	.867	.933	.840	.915	.749	.897	.856	.884	.940	.859	.907*	.877
SVM	PLT	.765	.886	.936	.962	.733	.866	.913*	.816	.897	.900*	.807	.862
ANN	–	.764	.884	.913	.901	.791*	.881	.932*	.859	.923	.667	.882	.854
SVM	ISO	.758	.882	.899	.954	.693*	.878	.907	.827	.897	.900*	.778	.852
ANN	PLT	.766	.872	.898	.894	.775	.871	.929*	.846	.919	.665	.871	.846
ANN	ISO	.767	.882	.821	.891	.785*	.895	.926*	.841	.915	.672	.862	.842
BST-DT	–	.874	.842	.875	.913	.523	.807	.860	.785	.933	.835	.858	.828
KNN	PLT	.819	.785	.920	.937	.626	.777	.803	.844	.827	.774	.855	.815
KNN	–	.807	.780	.912	.936	.598	.800	.801	.853	.827	.748	.852	.810
KNN	ISO	.814	.784	.879	.935	.633	.791	.794	.832	.824	.777	.833	.809
BST-STMP	PLT	.644	**.949**	.767	.688	.723	.806	.800	.862	.923	.622	.915*	.791
SVM	–	.696	.819	.731	.860	.600	.859	.788	.776	.833	.864	.763	.781
BST-STMP	ISO	.639	.941	.700	.681	.711	.807	.793	.862	.912	.632	.902*	.780
BST-STMP	–	.605	.865	.540	.615	.624	.779	.683	.799	.817	.581	.906*	.710
DT	ISO	.671	.869	.729	.760	.424	.777	.622	.815	.832	.415	.884	.709
DT	–	.652	.872	.723	.763	.449	.769	.609	.829	.831	.389	.899*	.708
DT	PLT	.661	.863	.734	.756	.416	.779	.607	.822	.826	.407	.890*	.706
LR	–	.625	.886	.195	.448	.777*	.852	.675	.849	.838	.647	.905*	.700
LR	ISO	.616	.881	.229	.440	.763*	.834	.659	.827	.833	.636	.889*	.692
LR	PLT	.610	.870	.185	.446	.738	.835	.667	.823	.832	.633	.895	.685
NB	ISO	.574	.904	.674	.557	.709	.724	.205	.687	.758	.633	.770	.654
NB	PLT	.572	.892	.648	.561	.694	.732	.213	.690	.755	.632	.756	.650
NB	–	.552	.843	.534	.556	.011	.714	-.654	.655	.759	.636	.688	.481

BST-DT: Boosted Decision Tree
RF: Random Forest
ANN: Neural net
KNN
SVM
NB: Naïve Bayes
LR: Logistic Regression

Bold: best
*: not significantly worse than best

Calibration methods:
PLT: Platt Calibration
ISO: Isotonic Regression
- None used

[pdf]

Caruana et al. 2008: comparison on high dimensional data

Table 2. Standardized scores of each learning algorithm

DIM	761	761	780	927	1344	3448	20958	105354	195203	405333	685569	—
ACC	STURN	CALAM	DIGITS	TIS	CRYST	KDD98	R-S	CITE	DSE	SPAM	IMDB	MEAN
MEDIAN	0.6901	0.7326	0.9681	0.9135	0.8820	0.9494	0.9599	0.9984	0.9585	0.9757	0.9980	—
BSTDT	0.9962	**1.0368**	**1.0136**	0.9993	**1.0178**	0.9998	0.9904	1.0000	0.9987	0.9992	1.0000	1.0047
RF	0.9943	1.0119	1.0076	1.0025	1.0162	**1.0000**	0.9995	0.9998	1.0013	**1.0044**	1.0000	1.0034
SVM	1.0044	1.0002	1.0024	1.0060	1.0028	0.9999	**1.0156**	**1.0008**	1.0004	1.0008	**1.0003**	1.0031
BAGDT	1.0001	1.0366	0.9976	1.0017	1.0111	**1.0000**	0.9827	1.0000	0.9996	0.9959	1.0000	1.0023
ANN	0.9999	0.9914	1.0051	1.0007	0.9869	**1.0000**	1.0109	1.0001	**1.0018**	1.0029	**1.0003**	1.0000
LR	1.0012	0.9911	0.8993	**1.0108**	1.0080	0.9999	1.0141	1.0001	1.0014	1.0026	0.9999	0.9935
BSTST	1.0077	1.0363	0.9017	0.9815	0.9930	**1.0000**	0.9925	0.9999	0.9948	0.9905	0.9989	0.9906
KNN	**1.0139**	0.9998	1.0122	0.9557	0.9972	0.9999	0.9224	1.0000	0.9987	0.9698	0.9996	0.9881
PRC	0.9936	0.9879	0.9010	0.9735	0.9930	**1.0000**	1.0119	0.9999	1.0007	1.0041	1.0001	0.9878
NB	0.9695	0.9362	0.8159	0.9230	0.9724	**1.0000**	1.0005	1.0000	0.9878	0.9509	0.9976	0.9594

- Boosted Decision Trees FTW again!
- RF second again!
- But note that Adaboost underperforms in the very high dimensional datasets, where RF excels

[pdf]

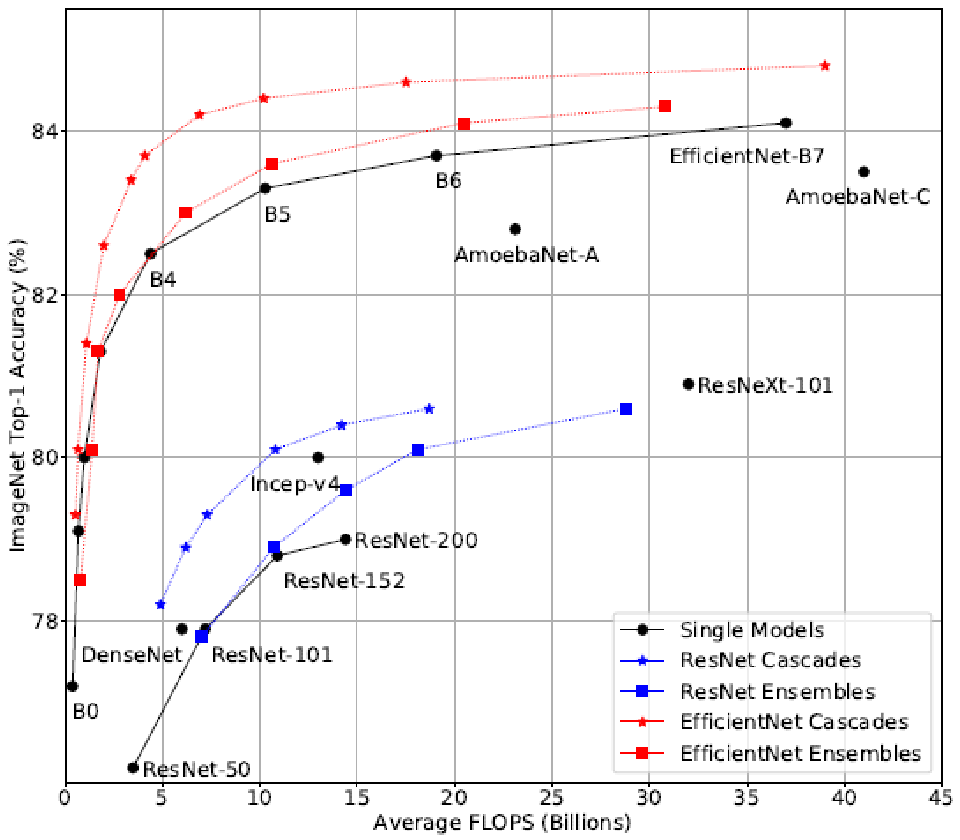
Boosted Trees and Random Forests work for different reasons

- Boosted trees
  - Use small trees (high bias, low variance) to iteratively refine the prediction
  - Combining prediction from many trees reduces bias
  - Overfitting is a danger (i.e. too many / too large trees eliminates train error but increases test error)

- Random forest
  - Use large trees (low bias, high variance)
  - Average of many tree predictions reduces variance
  - Hard to break – just train a whole bunch of trees

Other ensembles

- Can average predictions of any classifiers / regressors
  - But they should not be duplicates, so e.g. averaging multiple linear regressors trained on all features/data has no point
  - Averaging multiple deep networks (even when trained on all data) reduces error and improves confidence estimates
- Cascades: early classifiers make decisions on easy examples; later ones deal only with hard examples



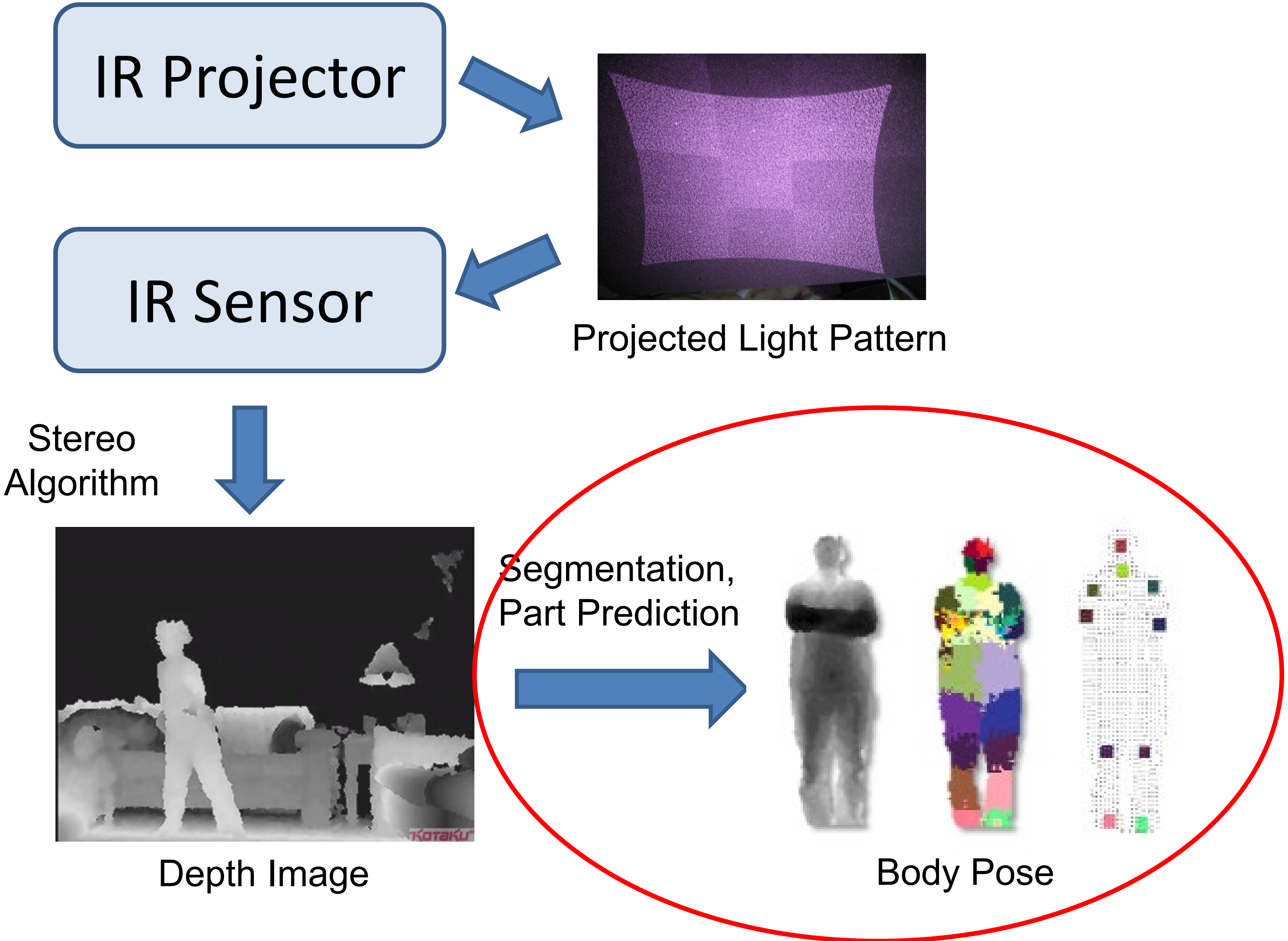
Wang et al. ICML 2022 [pdf]

Answer these questions

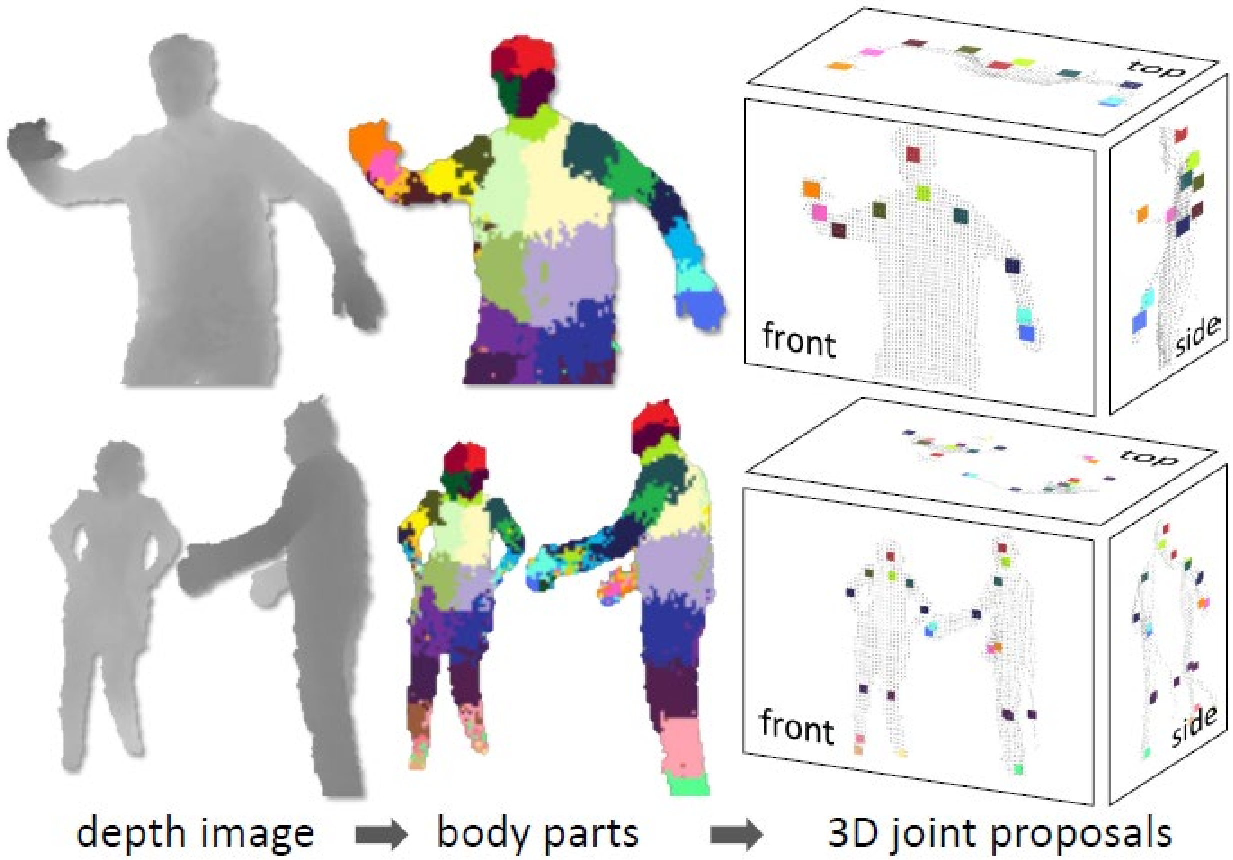
- Think about: Suppose you had an infinite sized audience to poll for a multiple choice question.
  - y={A, B, C, D}, where A is correct answer
  - A randomly sampled audience member will report an answer with probability P(y)
- What condition guarantees a correct answer?
- If your friend is a random member of the audience, what is the probability that his or her answer is correct?

- After that we'll do a detailed example with pose estimation

Example in detail: Depth from Kinect with RFs



Goal: estimate pose from depth image



Real-Time Human Pose Recognition in Parts from a Single Depth Image
Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake
CVPR 2011

Goal: estimate pose from depth image

RGB

Depth

Part Label Map

Joint Positions

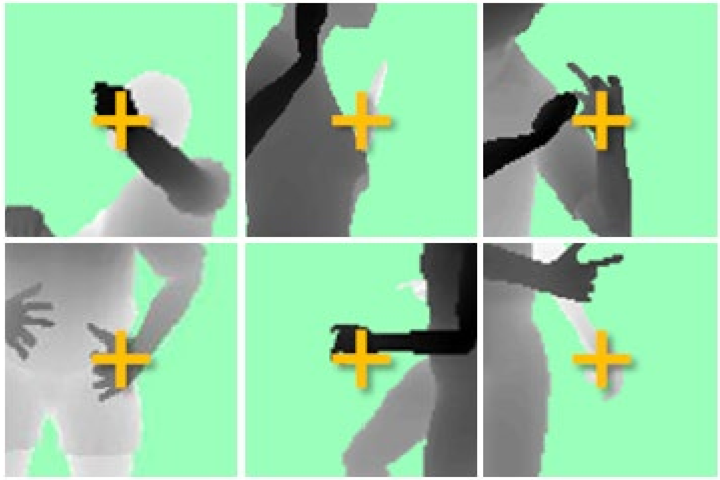
http://research.microsoft.com/apps/video/default.aspx?id=144455

Challenges

- Lots of variation in bodies, orientation, poses
- Needs to be very fast (their algorithm runs at 200 FPS on the Xbox 360 GPU)

Pose Examples

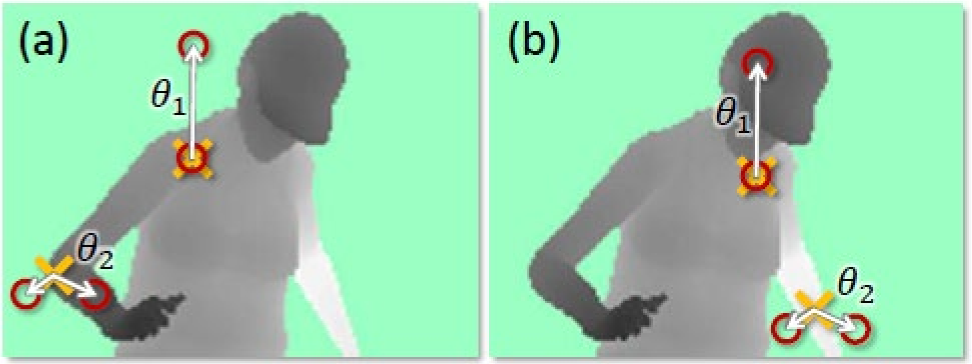
Examples of one part



Extract body pixels by thresholding depth

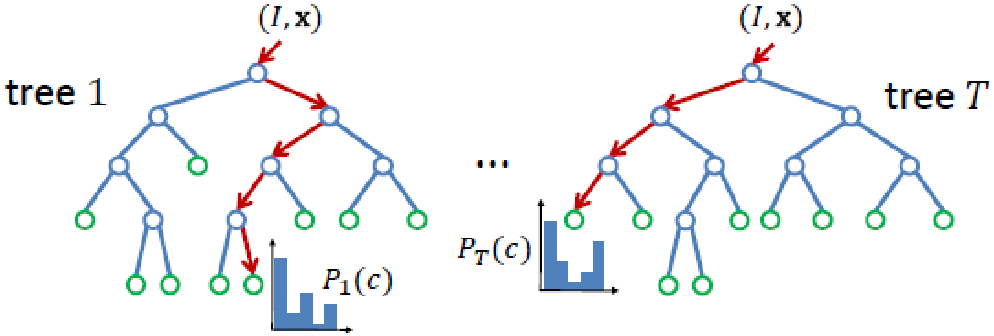
Basic learning approach

- Very simple features



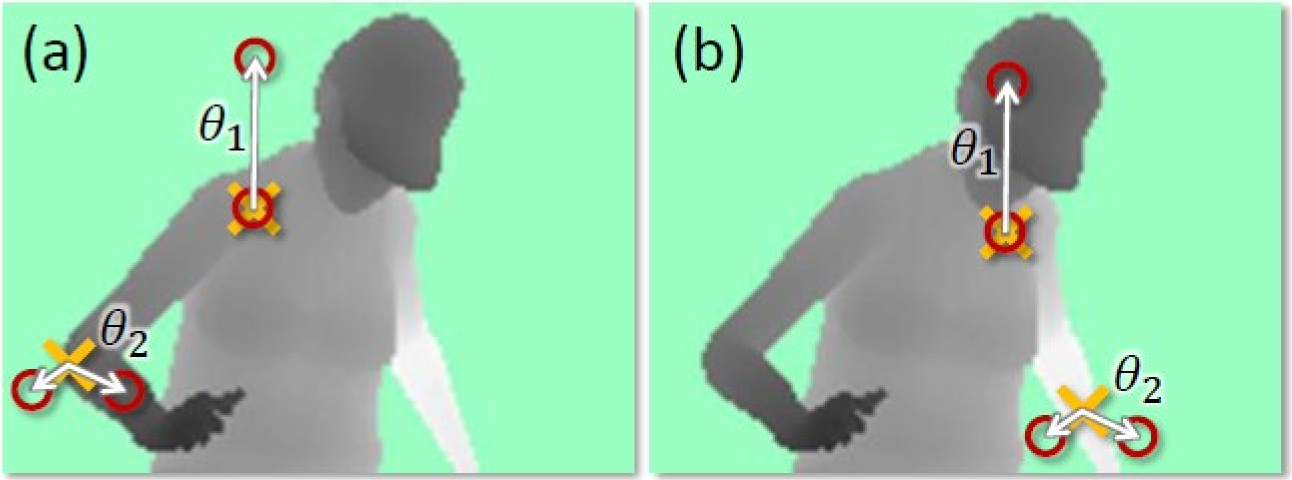
- Lots of data

- Flexible classifier



Features

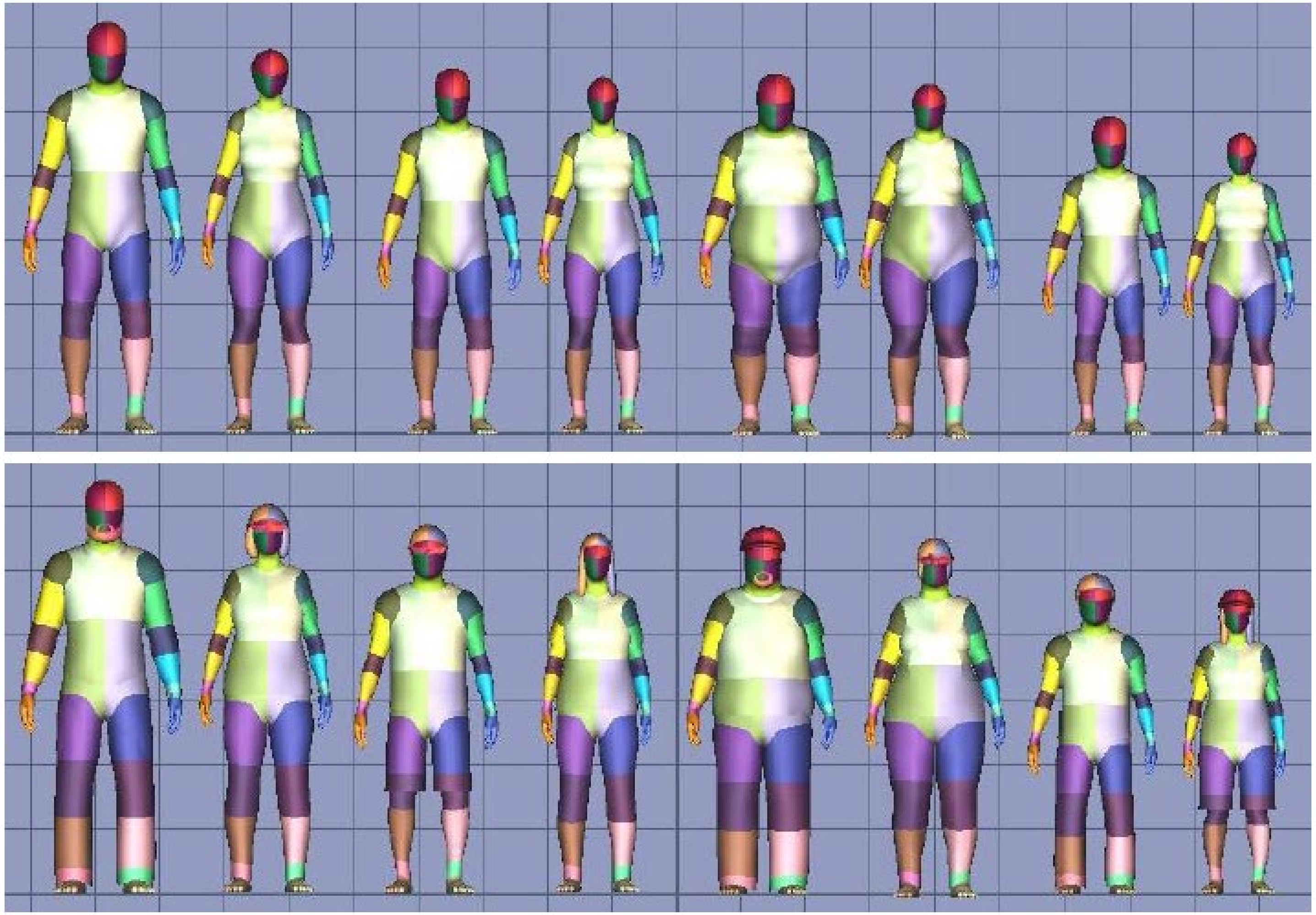
- Difference of depth at two offsets
  - Offset is scaled by depth at center



Get lots of training data

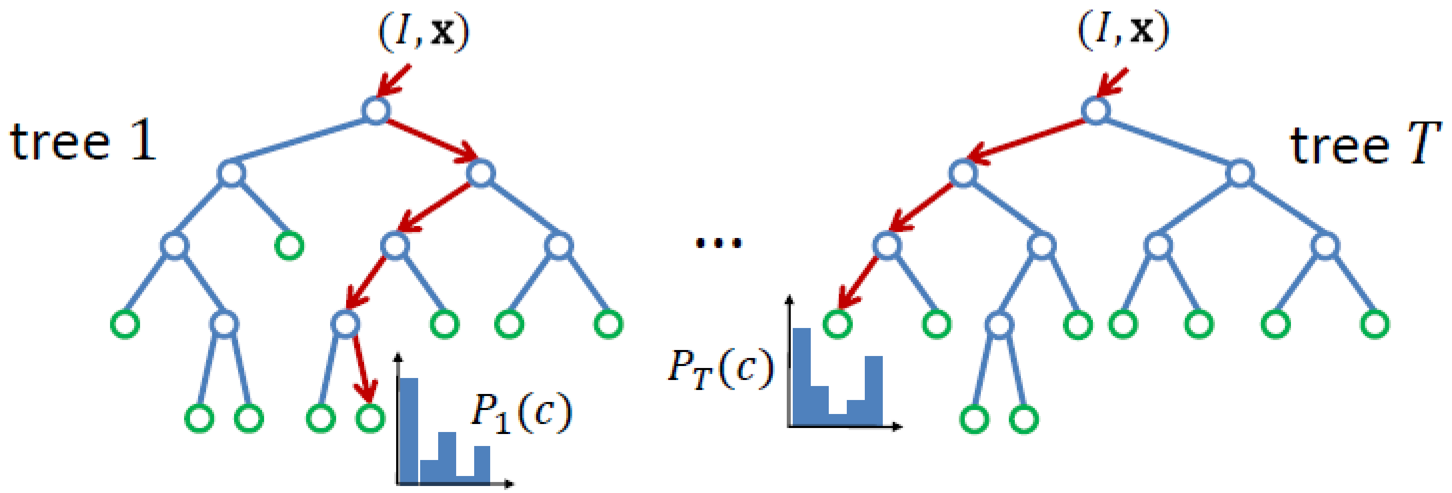
- Capture and sample 500K mocap frames of people kicking, driving, dancing, etc.
- Get 3D models for 15 bodies with a variety of weight, height, etc.
- Synthesize mocap data for all 15 body types

Body models



Part prediction with random forests

- Randomized decision forests: collection of independently trained trees
- Each tree is a classifier that predicts the likelihood of a pixel belonging to each part
  - Node corresponds to a thresholded feature
  - The leaf node that an example falls into corresponds to a conjunction of several features
  - In training, at each node, a subset of features is chosen randomly, and the most discriminative is selected

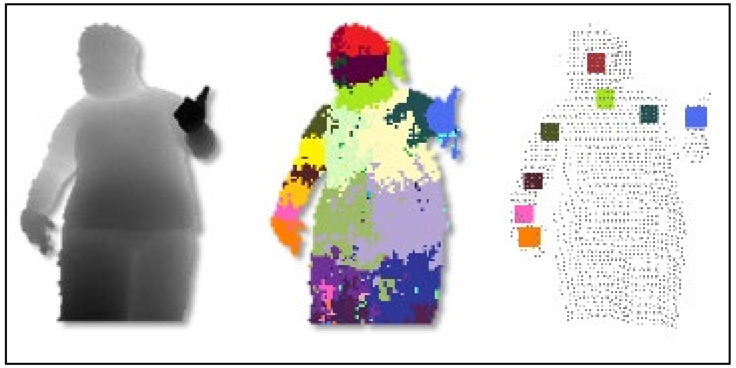


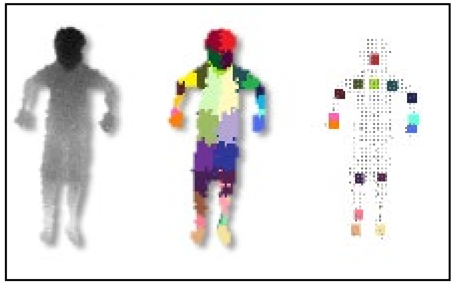
Joint estimation

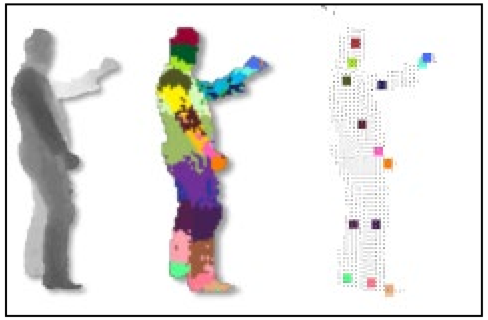
- Joints are estimated using mean-shift (a fast mode-finding algorithm)
- Observed part center is offset by pre-estimated value

Results

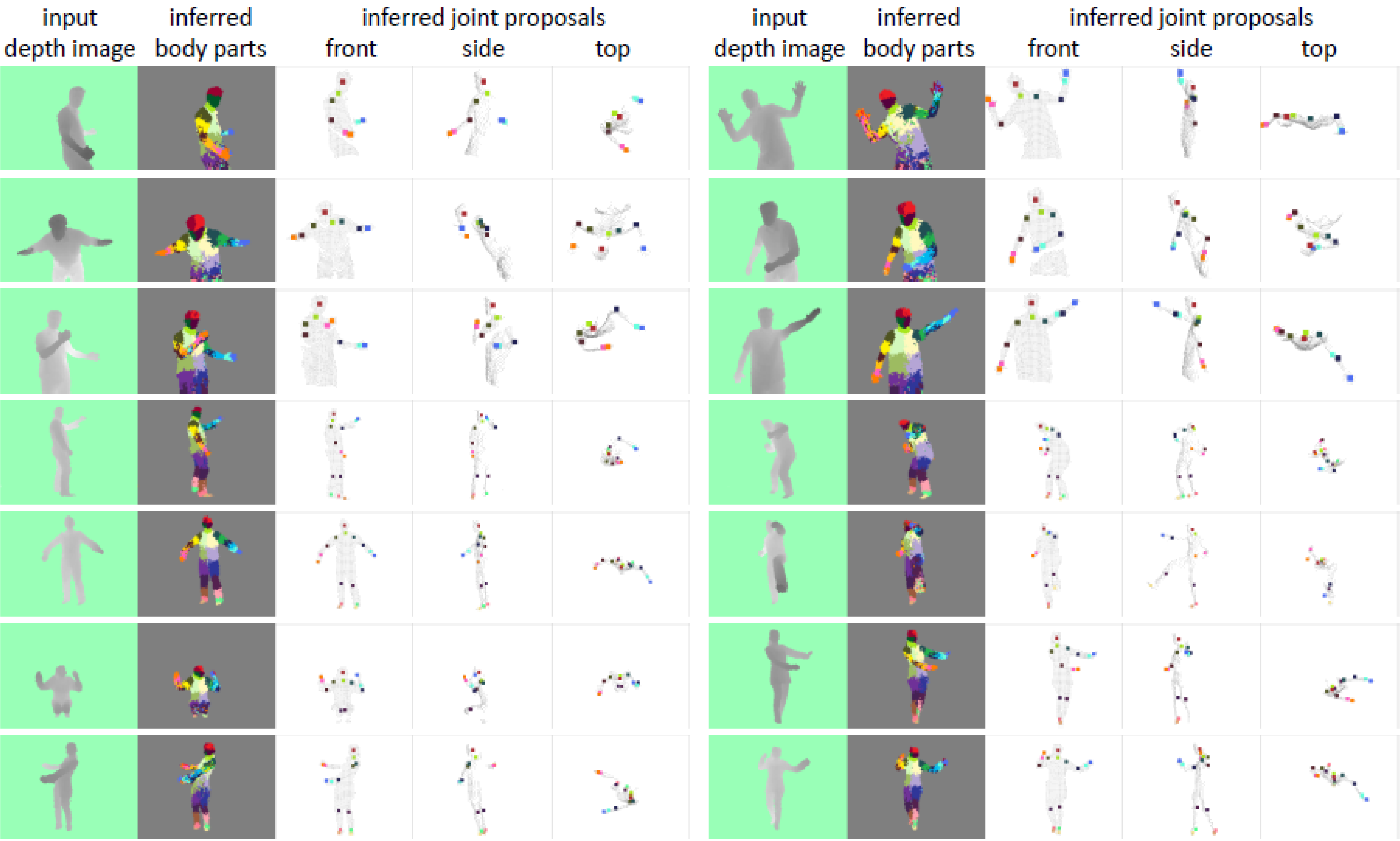
Ground Truth



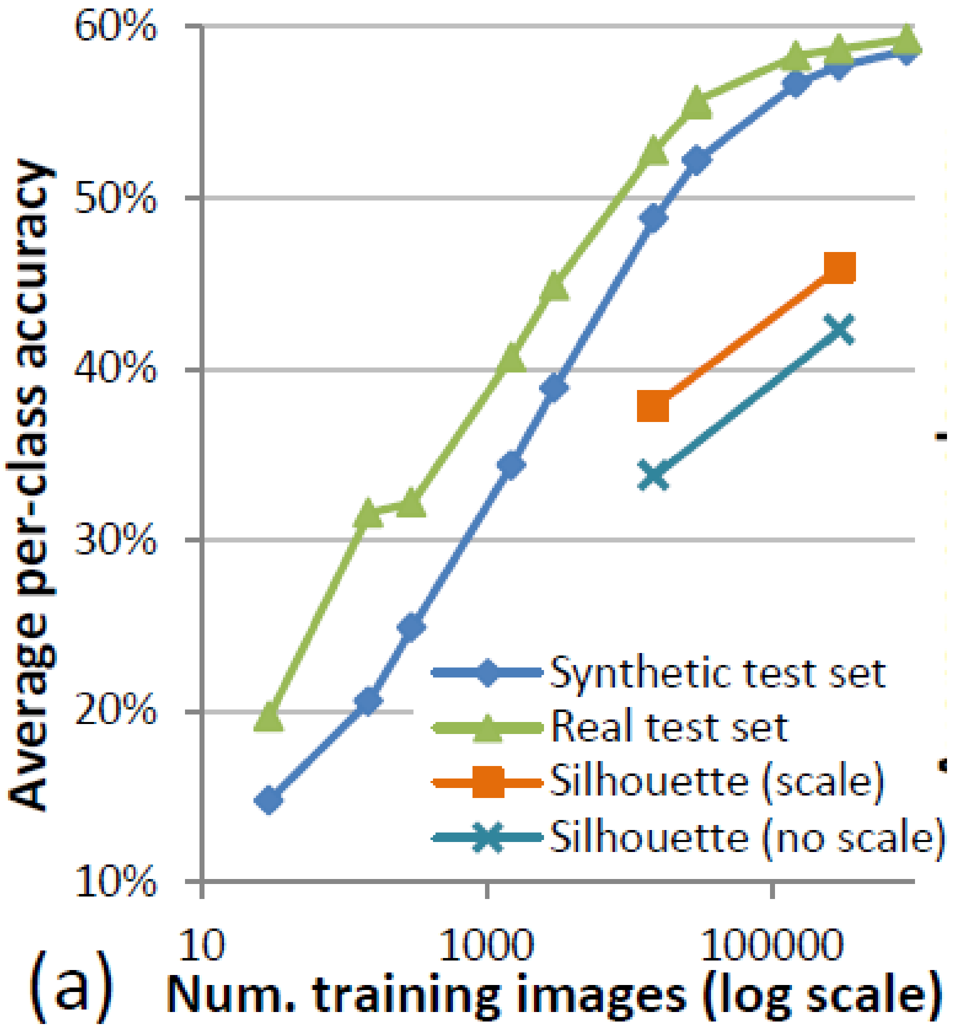




More results



Accuracy vs. Number of Training Examples



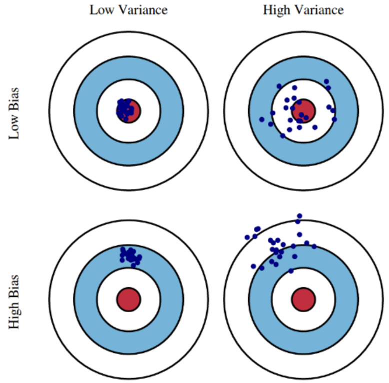
HW 4 (due April 1)

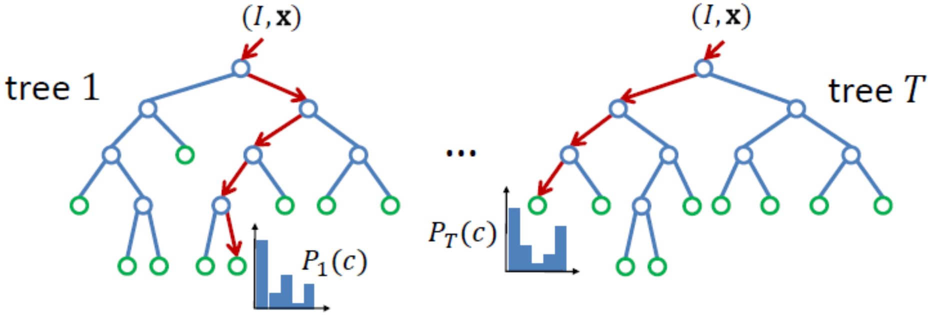
https://docs.google.com/document/d/1_9ZUFL7gi7Mq0-isQOcwDxhhmlDKVgdZg9mDaKokHEA/edit

Things to Remember

- Ensembles improve accuracy and confidence estimates by reducing bias and/or variance
- Boosted trees minimize bias by fixing previous mistakes
- Random forests minimize variance by averaging over multiple different trees
- Random forests and boosted trees are powerful classifiers and useful for a wide variety of problems

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x}) - y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2}$$





Thursday

- Stochastic Gradient Descent