CPSC 340:
Machine Learning and Data Mining

Ensemble Methods

Fall 2015

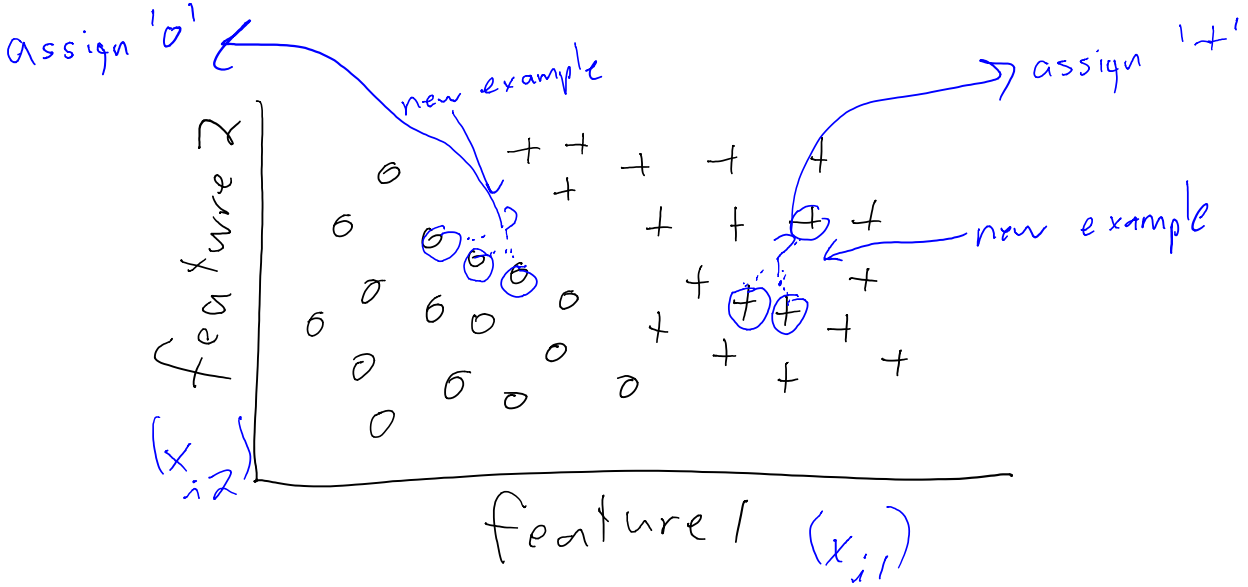
Admin

- Friday is last day to hand in Assignment 1.
  - Solutions posted after class Friday.
- Assignment 2 is up, due next Friday.

- We will have *standardized* tutorials every week.

K-Nearest Neighbours (KNN)

- K-nearest neighbours algorithm for classifying ‘x‘:
  - Find ‘k’ values of x_i that are most similar to x.
  - Use mode of corresponding y_i.



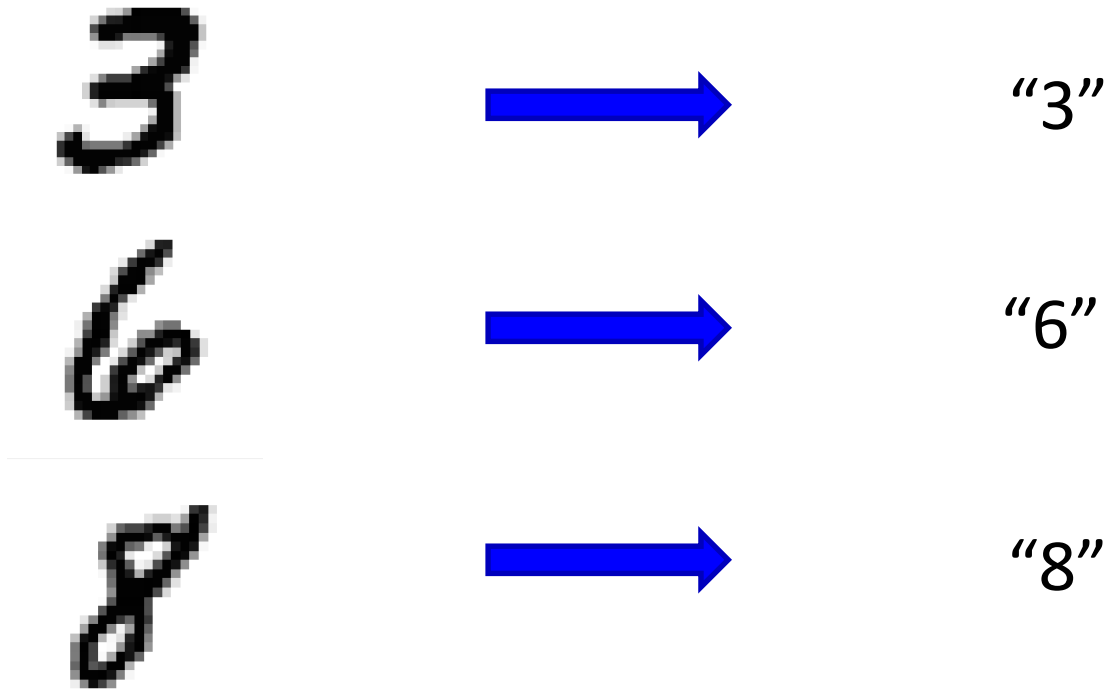
- Consistency:
  - Nearly-optimal test error with infinite data.

Curse of Dimensionality

- "Curse of dimensionality" refers to problems associated with exponential space of high-dimensions:
  - Volume of space grows exponentially with dimension.
  - Need exponentially more points to 'fill' a high-dimensional volume.
  - Distances become less meaningful (all vectors may have similar distances).
  - Emergence of hubs:
    - some datapoints are neighbours to many more points than average.
- KNN is also problematic if range of some features is much larger than others.
- Nevertheless, KNN is often hard to beat!

Application: Optical Character Recognition

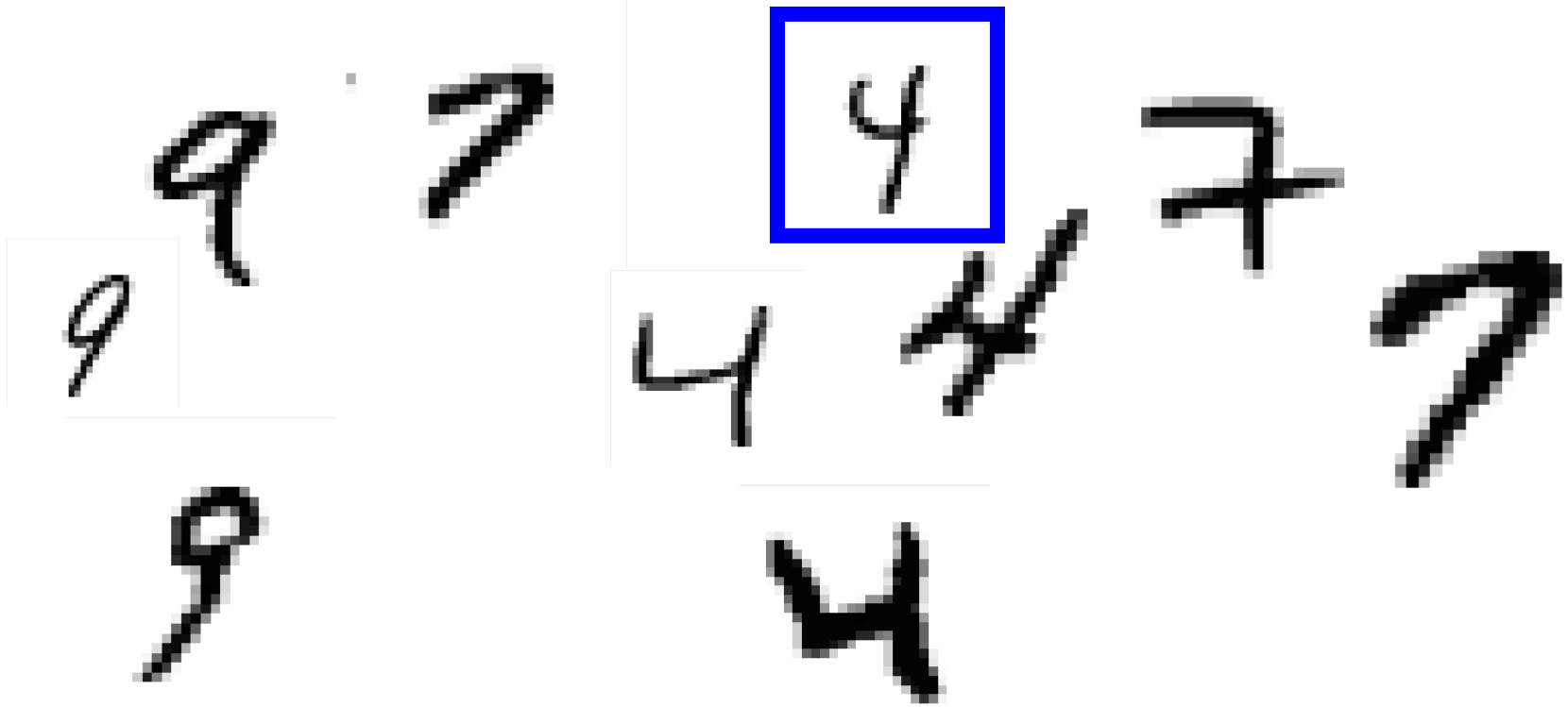
- We have collection of letter/digit images, and corresponding labels:



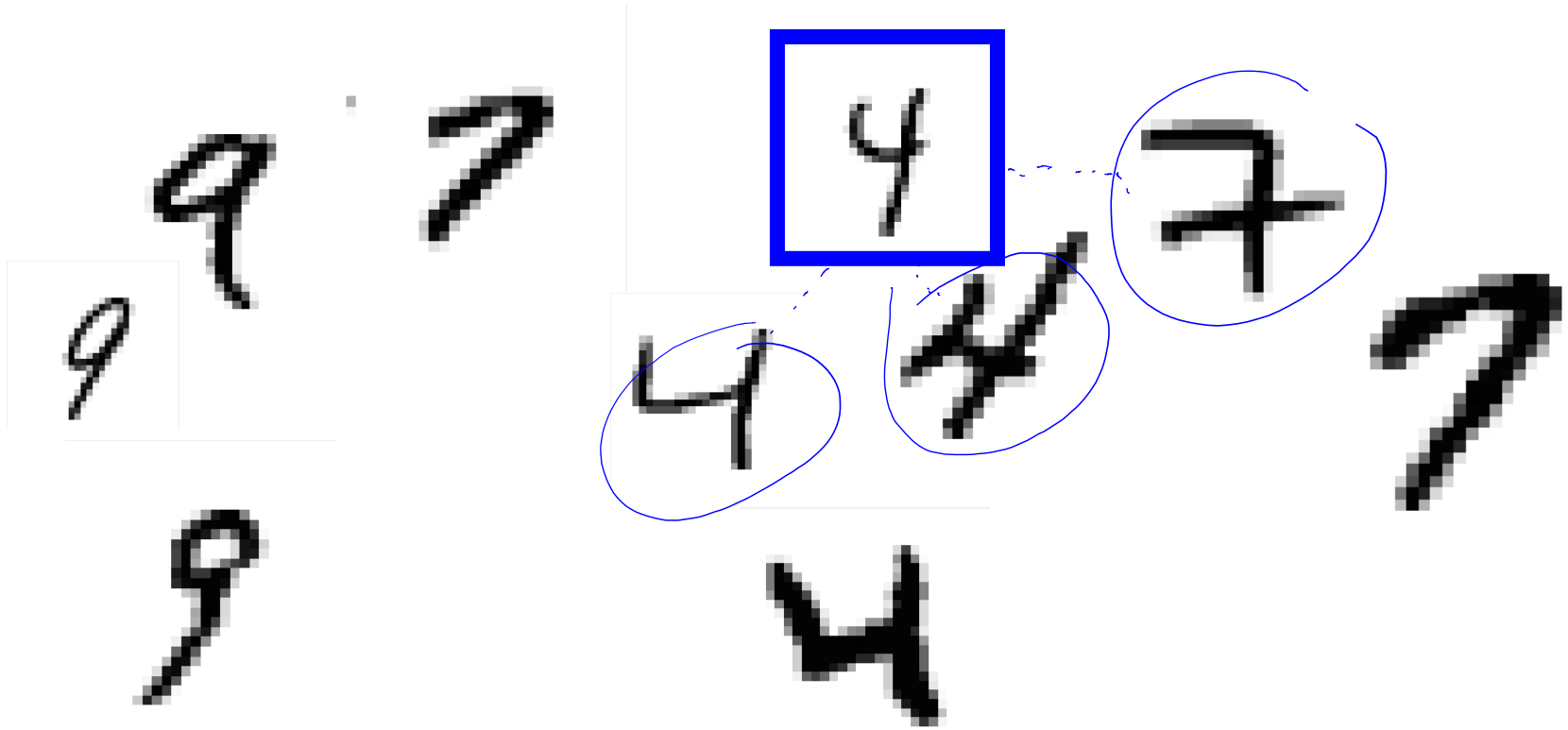
- Use supervised learning to automatically recognize letters/digits:
  - y_i could be the letter/digit, x_i could be the values of the pixels.

KNN for Optical Character Recognition

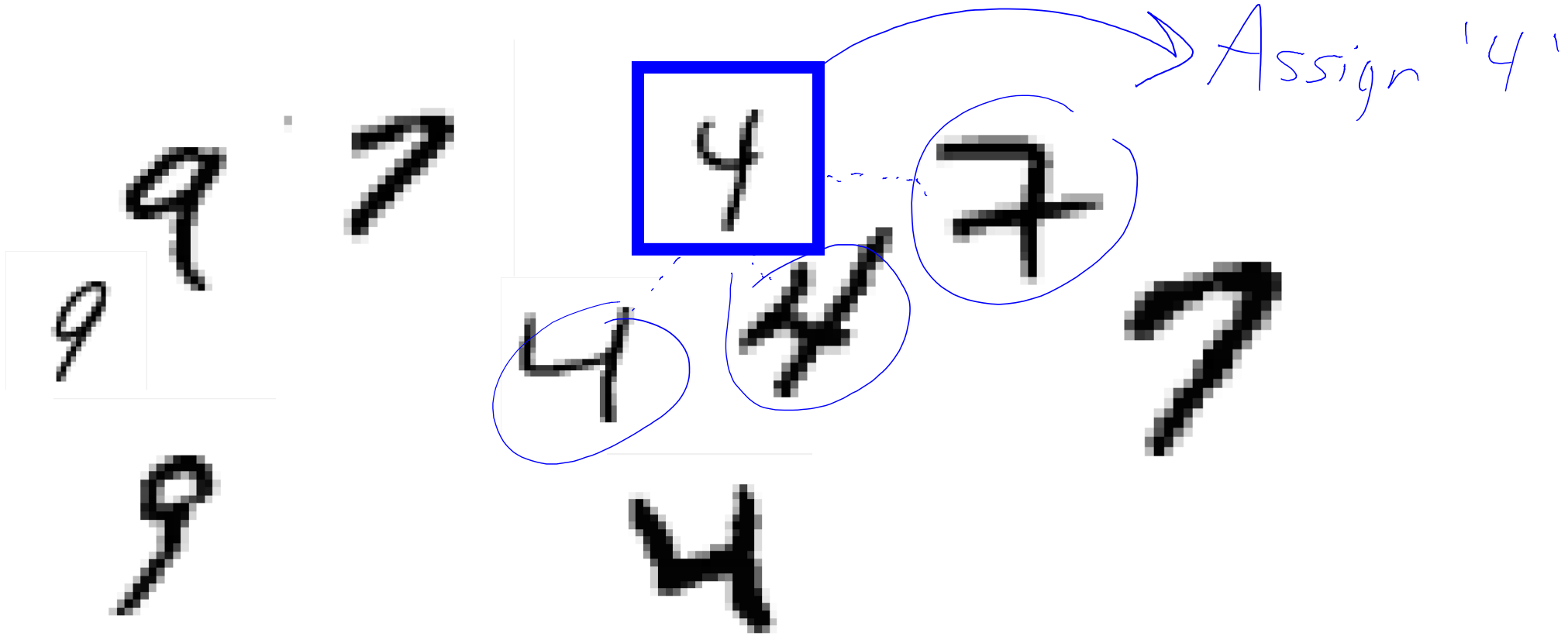
KNN for Optical Character Recognition



KNN for Optical Character Recognition



KNN for Optical Character Recognition



What the Computer Sees

- There is huge difference between what we see and what KNN sees:

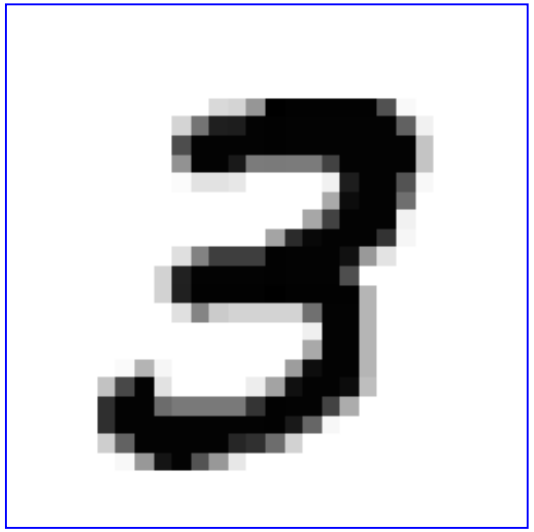
What we see:

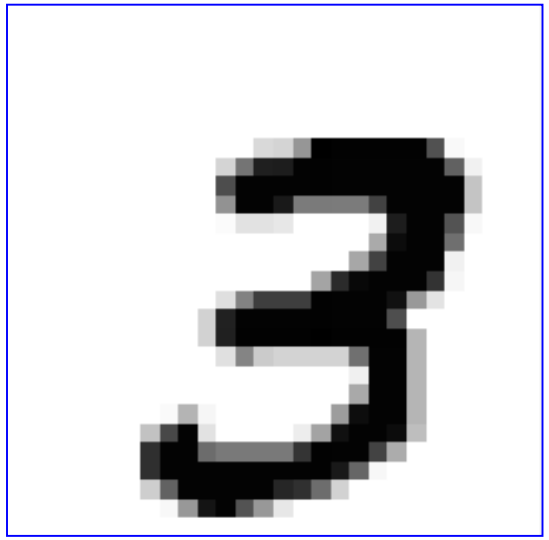
What the computer “sees”:

Actually, it’s worse:

What the Computer Sees

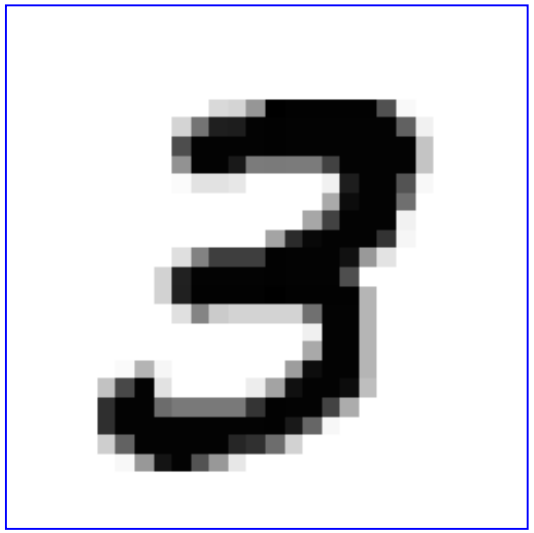
- Are these two images ‘similar’?

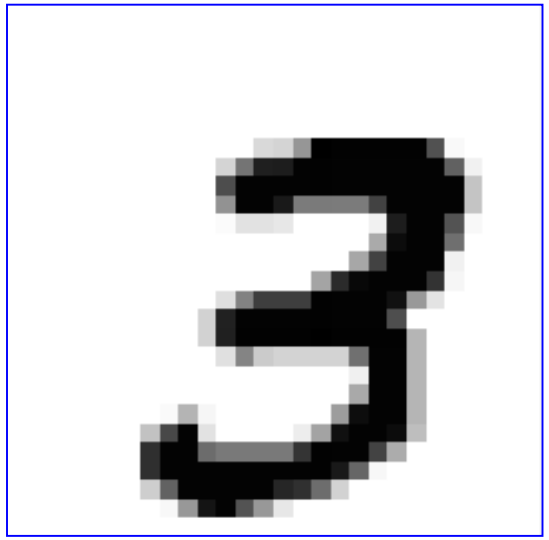




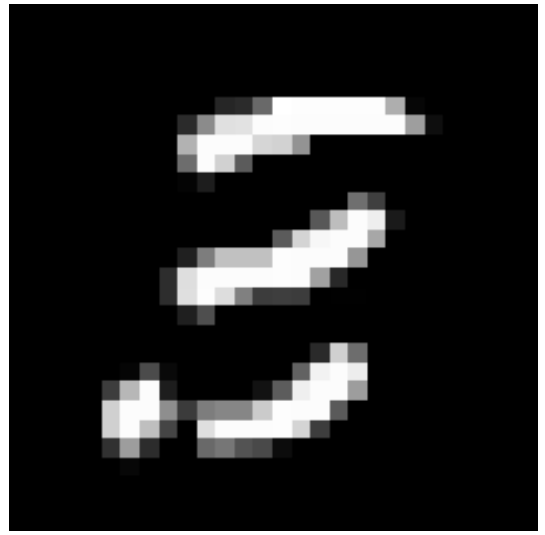
What the Computer Sees

- Are these two images ‘similar’?





Difference:

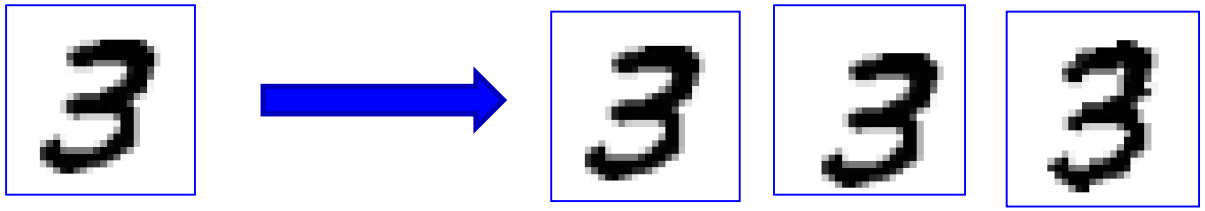


- KNN does not know that labels should be translation invariant.

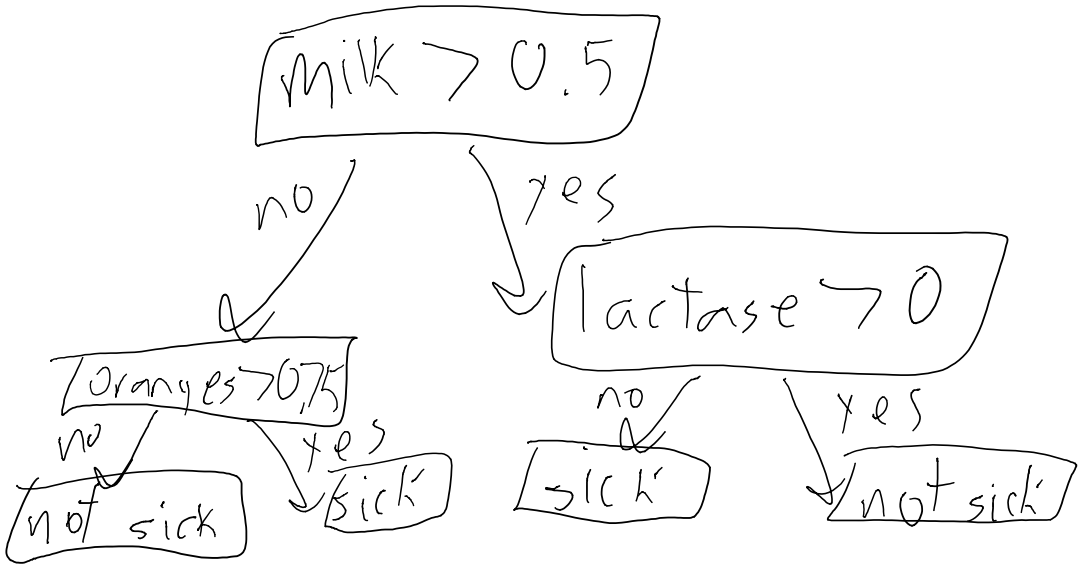
Encouraging Invariance

- May want classifier to be invariant to certain feature transforms.
  - Digits: translations, small rotations, changes in size, mild warping,…

- The hard/slow way is to modify your distance function:
  - Find neighbours that require the ‘smallest’ transformation of image.
- The easy/fast way is to just add transformed data during training:
  - Add translated/rotate/resized/warped versions of training images.
  - Important part of many successful vision systems.



Decision Trees vs. Naïve Bayes vs. KNN

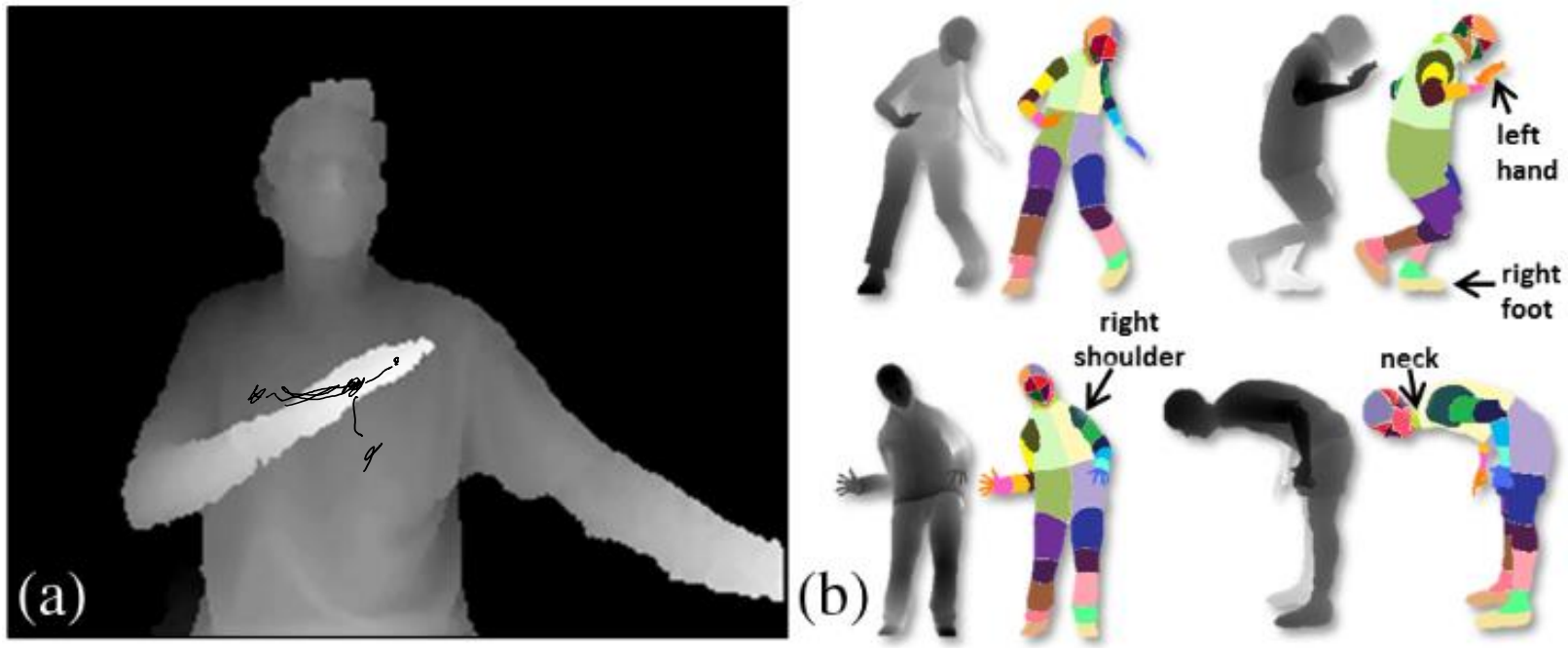


$$p(\text{sick} \mid \text{milk}, \text{oranges}, \text{lactase})$$
$$\propto p(\text{milk}, \text{oranges}, \text{lactase} \mid \text{sick})\, p(\text{sick})$$
$$\approx p(\text{milk} \mid \text{sick})\, p(\text{oranges} \mid \text{sick})\, p(\text{lactase} \mid \text{sick})\, p(\text{sick})$$

(milk = 0.6, oranges = 0.2, lactase = 0, ?) is close to
(milk = 0.7, oranges = 0.3, lactase = 0, sick), so predict sick.

Body-Part Recognition

- Microsoft Kinect:
  - Real-time recognition of 31 body parts from laser depth data.

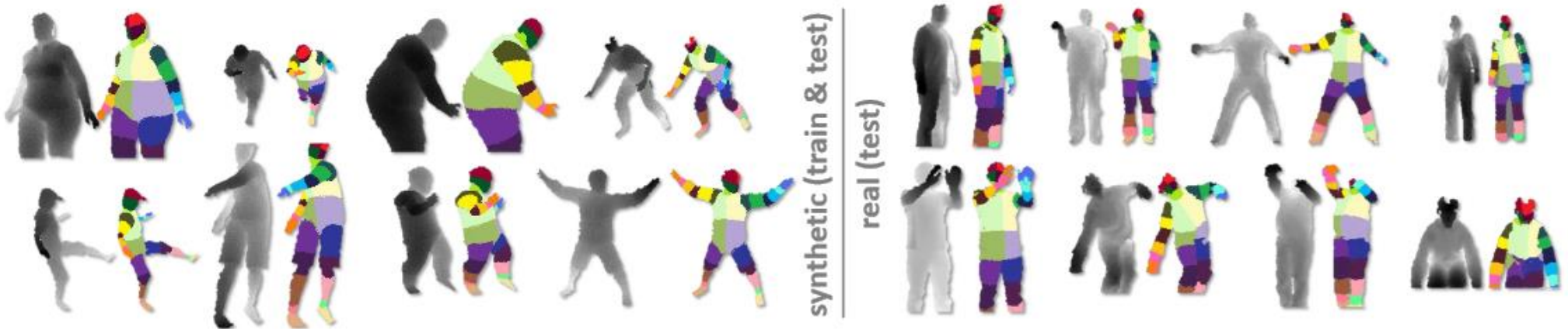


- How could we write a program to do this?

http://research.microsoft.com/pubs/158806/CriminisiForests_FoundTrends_2011.pdf

Some Ingredients of Kinect

1. Collect hundreds of thousands of labeled images (motion capture).
   - Variety of pose, shape, clothing, and crop.
2. Build a simulator that increases variety by making even more images.



3. Extract features of each location, that are cheap enough for real-time calculation (depth differences between pixel and pixels nearby.)
4. Treat classifying an individual pixel as a supervised learning problem.
5. Run classifier in parallel on all pixels using graphical processing unit (GPU).

http://research.microsoft.com/pubs/145347/BodyPartRecognition.pdf

Supervised Learning Step

- ALL of those steps are important, but we'll focus on learning step.

- Do we have any classifiers that are accurate and run in real time?
  - Decision trees and naïve Bayes are fast, but often not very accurate.
  - KNN is often accurate, but not very fast.

- Deployed system uses an ensemble method called random forests.

Ensemble Methods

- Ensemble methods are classifiers that have classifiers as input.
- Also called ‘meta-learning’.
- Final meta-classifier can have much higher accuracy than inputs.
- They have the best names:
  - Averaging.
  - Boosting.
  - Bootstrapping.
  - Bagging.
  - Cascading.
  - Random Forests.
  - Stacking.

Ensemble Methods

- Remember the fundamental trade-off:
  1. How small you can make the training error.

     vs.

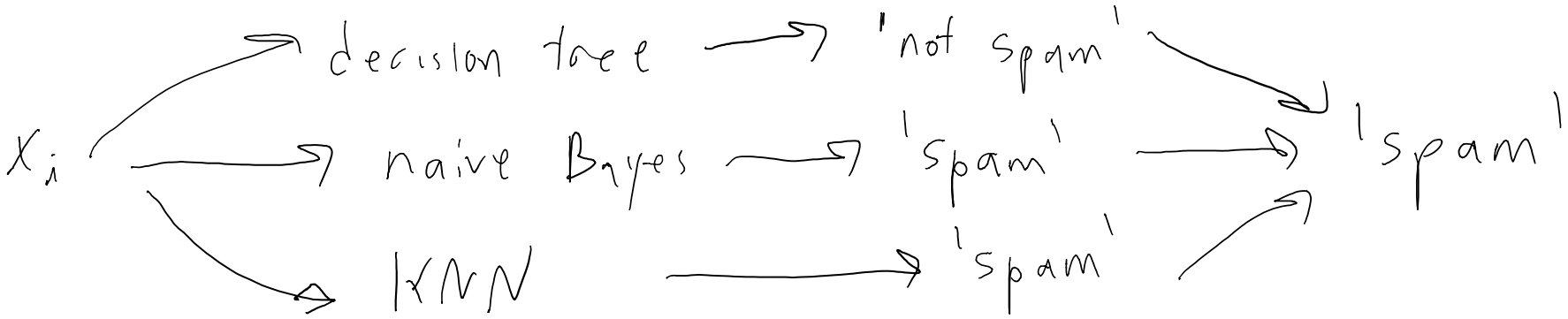
  2. How well training error approximates the test error.
- Goal of ensemble methods is that meta-classifier:
  - Does much better on one of these than individual classifiers.
  - Doesn't do too much worse on the other.
- There are many variations, but this gives roughly two types:
  a) Boosting: take simple classifier that underfits, improve its training error.
  b) Averaging: take complex classifier that overfits, improve its test error.

Boosting

- Input to boosting is classifier that:
  - Can obtain >50% accuracy on weighted training data.
  - And is simple enough that the training error approximates the test error.
- Example: decision stumps or low-depth decision trees.
- Basic steps:
  1. Fit a classifier on the training data.
  2. Give a higher weight to examples that the classifier got wrong.
  3. Fit a classifier on the weighted training data.
  4. Go back to 2.
- Final prediction: weighted vote of individual classifier predictions.
- Boosted decision trees are very fast/accurate classifiers.

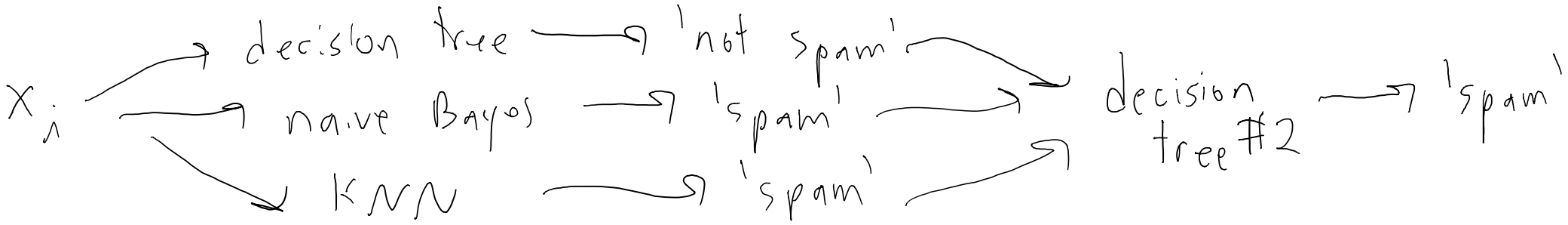
Averaging

- Input to averaging is the predictions of a set of models:
  - Decision trees make one prediction.
  - Naïve Bayes makes another prediction.
  - KNN makes another prediction.
- Model averaging:
  - Take the mode of the predictions (or average if probabilistic).



Averaging

- Input to averaging is the predictions of a set of models:
  - Decision trees make one prediction.
  - Naïve Bayes makes another prediction.
  - KNN makes another prediction.
- Model averaging:
  - Take the mode of the predictions (or average if probabilistic).
- Stacking:
  - Fit another classifier that uses the predictions as features.



Averaging

- Input to averaging is the predictions of a set of models:
  - Decision trees make one prediction.
  - Naïve Bayes makes another prediction.
  - KNN makes another prediction.
- Model averaging:
  - Take the mode of the predictions (or average if probabilistic).
- Stacking:
  - Fit another classifier that uses the predictions as features.
- These often perform better than individual models:
  - Some sort of averaging is typically used in Kaggle-winning systems.
  - E.g., Netflix $1M user-rating competition winner was stacked classifier.

Averaging

- Why does averaging work?
- Consider a set of classifiers that tend to overfit:
  - For example, a set of deep decision trees.
- If they all overfit in exactly the same way, averaging does nothing.
- If they overfit in *different* ways, averaging can improve test error.
  - While keeping similar training error.
- Pays less attention to specific overfitting done by each classifier.

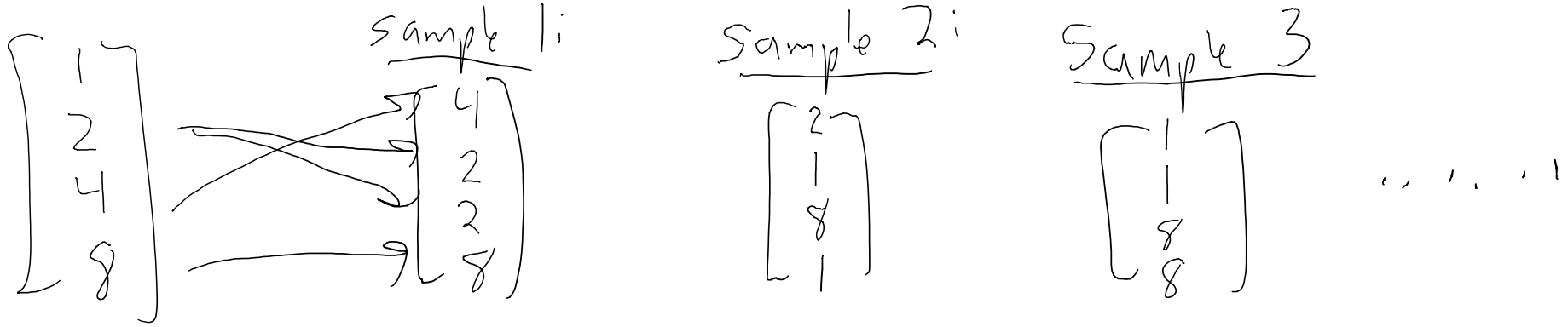
Random Forests

- Random forests average a set of deep decision trees.
  - Tend to be one of the best 'out of the box' classifiers.
  - Usually, get close to the best performance of any method on the first run.

- How do we ensure deep decision trees overfit in different ways?
  - If just fit a decision tree repeatedly, all trees will be the same.
- Two key ingredients in random forests:
  - Bootstrap sampling.
  - Random splits.

Random Forest Ingredient 1: Bagging

- Bootstrap sample of a list of ‘n’ elements:
  - A set of ‘n’ elements, chosen independently from list with replacement.



  - Each sample contains ~63% of original samples, re-weighted.
  - Usually, it used to estimate how sensitive a statistic is to the data.
- Bootstrap aggregation (bagging):
  - Fit a classifier on a boostrap sampling of the object (x_i, y_i).
  - Average results.

Random Forest Ingredient 2: Random Trees

- When fitting each decision stump ton construct deep decision tree:
  - Do not consider all features when searching for optimal rule.
  - Instead, only consider a small number of randomly-chosen features.
- These random trees will tend to be very different from each other.
- They will still overfit, but in *different* ways.
- The average will tend to have a much lower test error.
  - Bonus: fitting the decision trees is faster!

- I often say that random forests are one of the 'best' classifiers.
- Fernandez-Delgado et al. [2014]:
  - Compared 179 classifiers on 121 datasets.
  - Classifiers most likely to be the best are random forests.

Summary

1. Adding transformed data can lead to more variety in training set and invariance in classifier.
2. Ensemble methods take classifiers as inputs.
3. Boosting turns ‘weak’ classifiers into ‘strong’ classifiers.
4. Averaging predictions often improves performance.
5. Random forests incorporate randomization and averaging of deep decision trees to give strong empirical performance.

- Next time:
  - We start unsupervised learning.