CPSC 340:
Machine Learning and Data Mining

Ensemble Methods

Fall 2019

Admin

- Welcome to the course!

- Course webpage:
  - https://www.cs.ubc.ca/~schmidtm/Courses/340-F19/

- Assignment 1:
  - 2 late days to hand in tonight.

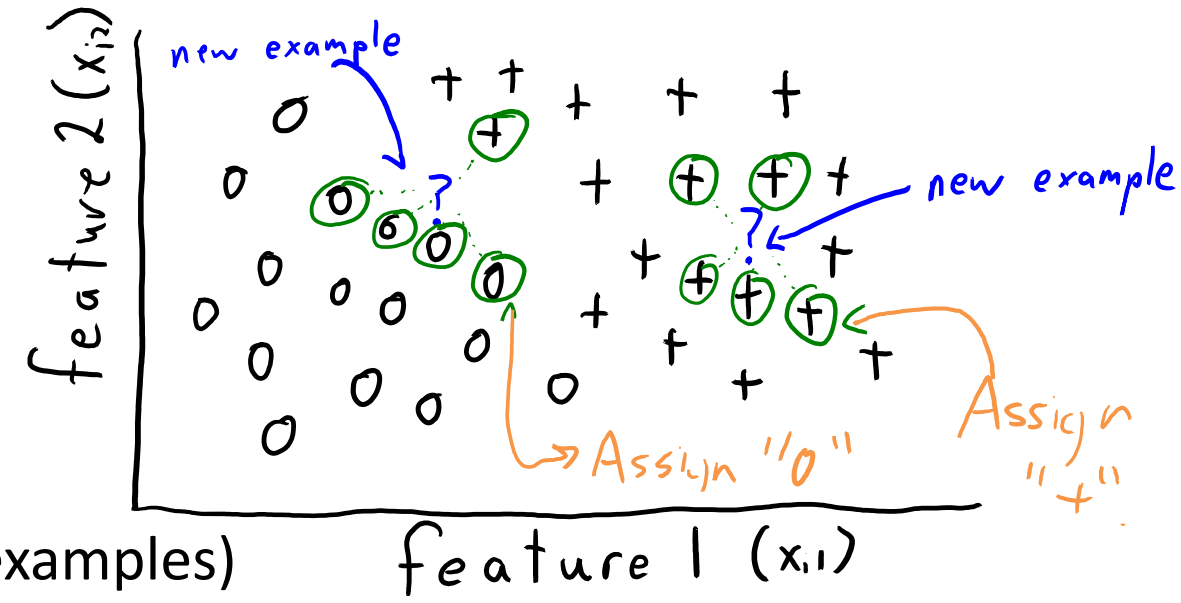
- Assignment 2 is out.
  - Due Friday of next week. It's long so start early.

Last Time: K-Nearest Neighbours (KNN)

- K-nearest neighbours algorithm for classifying $\tilde{x}_i$:
  - Find 'k' values of x_i that are most similar to $\tilde{x}_i$.
  - Use mode of corresponding y_i.

- Lazy learning:
  - To "train" you just store X and y.

- Non-parametric:
  - Size of model grows with 'n' (number of examples)
  - Nearly-optimal test error with infinite data.



- But high prediction cost and may need large 'n' if 'd' is large.

Defining "Distance" with "Norms"

- A common way to define the "distance" between examples:
  - Take the "norm" of the difference between feature vectors.

$$\|x_i - \tilde{x}_i\|_2 = \sqrt{\sum_{j=1}^{d} (x_{ij} - \tilde{x}_{ij})^2}$$

train example — test example — "L_2-norm"

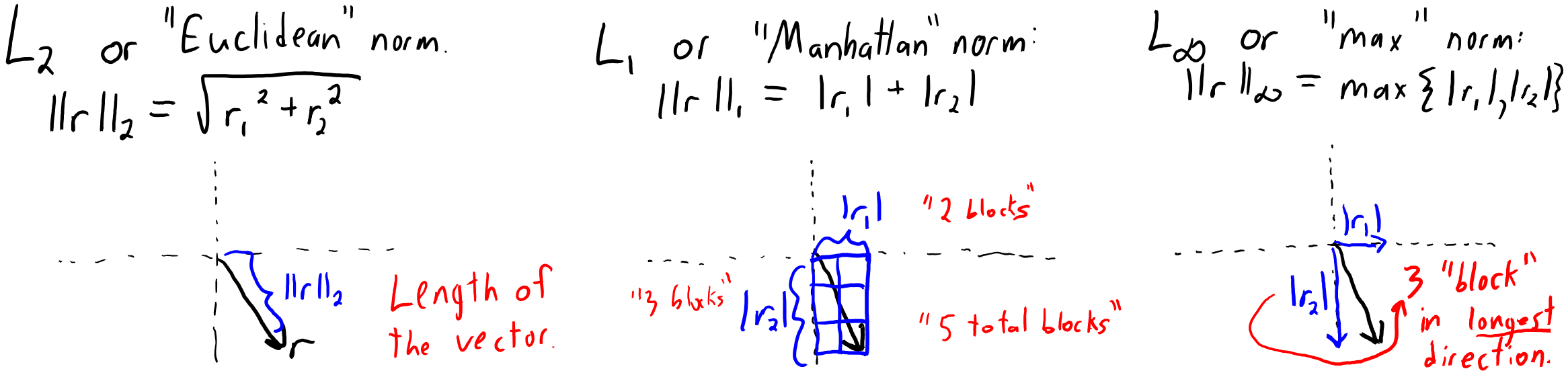
- Norms are a way to measure the "length" of a vector.
  - The most common norm is the "L2-norm" (or "Euclidean norm"):

$$\|r\|_2 = \sqrt{\sum_{j=1}^{d} r_j^2}$$

  - Here, the "norm" of the difference is the standard Euclidean distance.

L2-norm, L1-norm, and L∞-Norms.

- The three most common norms: L2-norm, L1-norm, and L∞-norm.
  - Definitions of these norms with two-dimensions:



  - Notation: we often leave out the "2" for the L2-norm: $\|r\| = \|r\|_2$

Norms in d-Dimensions

- We can generalize these common norms to d-dimensional vectors:

L_2: $\|r\|_2 = \sqrt{\sum_{j=1}^{d} r_j^2}$ L_1: $\|r\|_1 = \sum_{j=1}^{d} |r_j|$ L_∞: $\max_j \{|r_j|\}$

E.g., in 3-dimensions:

$\|r\|_2 = \sqrt{r_1^2 + r_2^2 + r_3^2}$

in 4-dimensions:

$\|r\|_2 = \sqrt{r_1^2 + r_2^2 + r_3^2 + r_4^2}$

Notation: $\|r\|_2^2 = (\|r\|_2)^2 = \left(\sqrt{\sum_{j=1}^{d} r_j^2}\right)^2 = \sum_{j=1}^{d} r_j^2 = r^T r$

Different ways to write the same thing

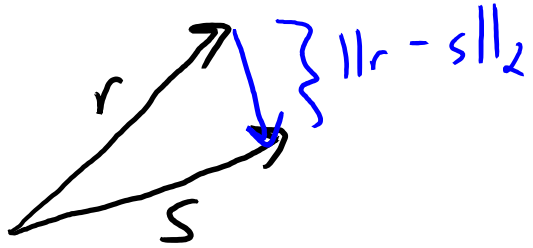
- These norms place different "weights" on large values:
  - L_1: all values are equal.
  - L_2: bigger values are more important (because of squaring).
  - L_∞: only biggest value is important.

Norms as Measures of Distance

- By taking norm of difference, we get a "distance" between vectors:

$$\|r - s\|_2 = \sqrt{(r_1 - s_1)^2 + (r_2 - s_2)^2}$$

$= \|r - s\|$ "Euclidean distance"



$$\|r - s\|_1 = |r_1 - s_1| + |r_2 - s_2|$$

"Number of blocks you need to walk to get from r to s."

$$\|r - s\|_\infty = \max\{|r_1 - s_1|, |r_2 - s_2|\}$$

"Most number of blocks in any direction you would have to walk."

[Infinite Series Video]

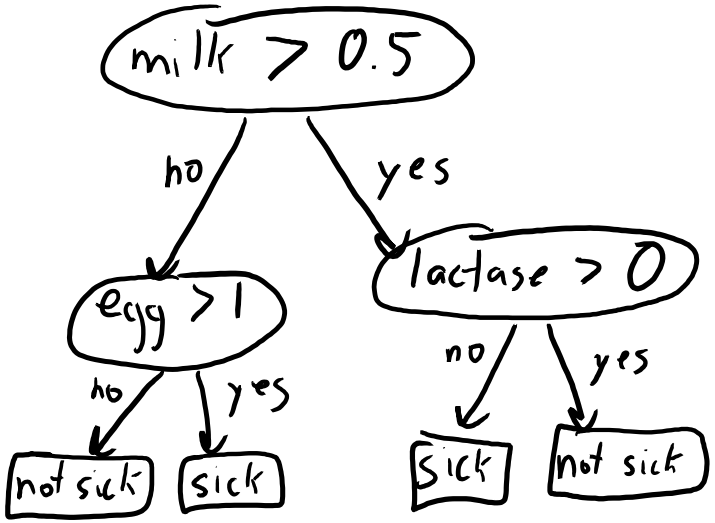
KNN Distance Functions

- Most common KNN distance functions: norm($x_i - x_j$).
  - L1-, L2-, and L∞-norm.
  - Weighted norms (if some features are more important): $\sum_{j=1}^{d} v_j |x_j|$
  - "Mahalanobis" distance (takes into account correlations).
    - See bonus slide for what functions define a "norm".

"weight" of feature j

- But we can consider other distance/similarity functions:
  - Jaccard similarity (if x_i are sets).
  - Edit distance (if x_i are strings).
  - Metric learning (*learn* the best distance function).

Decision Trees vs. Naïve Bayes vs. KNN

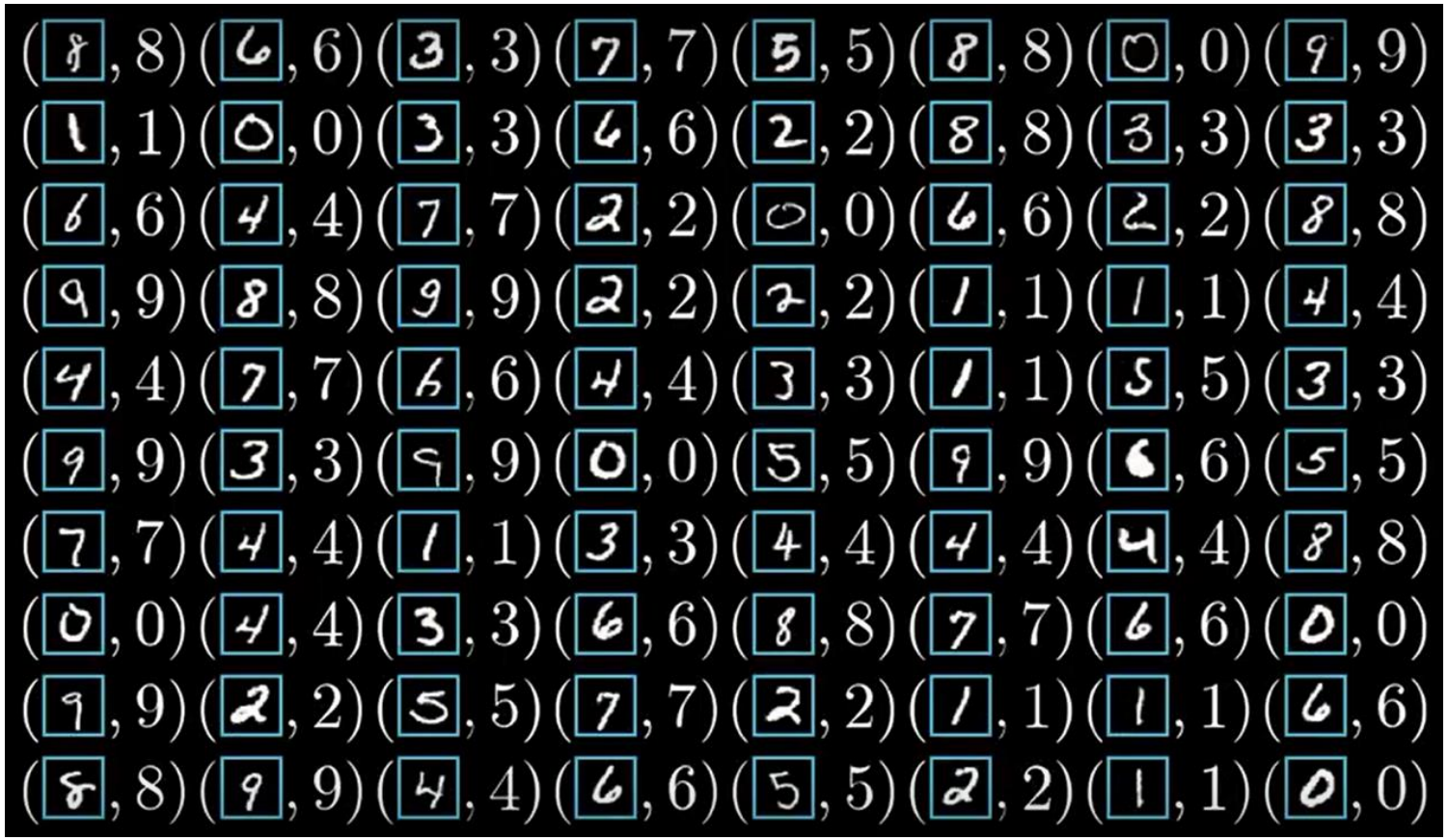


$$p(\text{sick} \mid \text{milk}, \text{egg}, \text{lactase}) \approx p(\text{milk} \mid \text{sick})\, p(\text{egg} \mid \text{sick})\, p(\text{lactase} \mid \text{sick})\, p(\text{sick})$$

(milk = 0.6, egg = 2, lactase = 0, ??) is close to
(milk = 0.7, egg = 2, lactase = 0, sick) so predict sick.

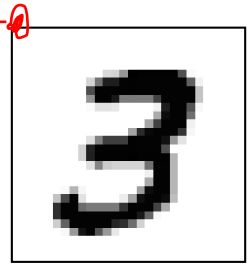
Application: Optical Character Recognition

- To scan documents, we want to turn images into characters:
  - "Optical character recognition" (OCR).



Application: Optical Character Recognition

- To scan documents, we want to turn images into characters:
  - "Optical character recognition" (OCR).



"3"

pixel by pixel

  - Turning this into a supervised learning problem (with 28 by 28 images):

X =

(1,1)	(2,1)	(3,1)	…	(28,1)	(1,2)	(2,2)	…	(14,14)	…	(28,28)
0	0	0		0	0	0		1		0
0	0	0		0	0	0		1		0
0	0	0		0	0	0		0		0
0	0	0		0	0	0		1		0

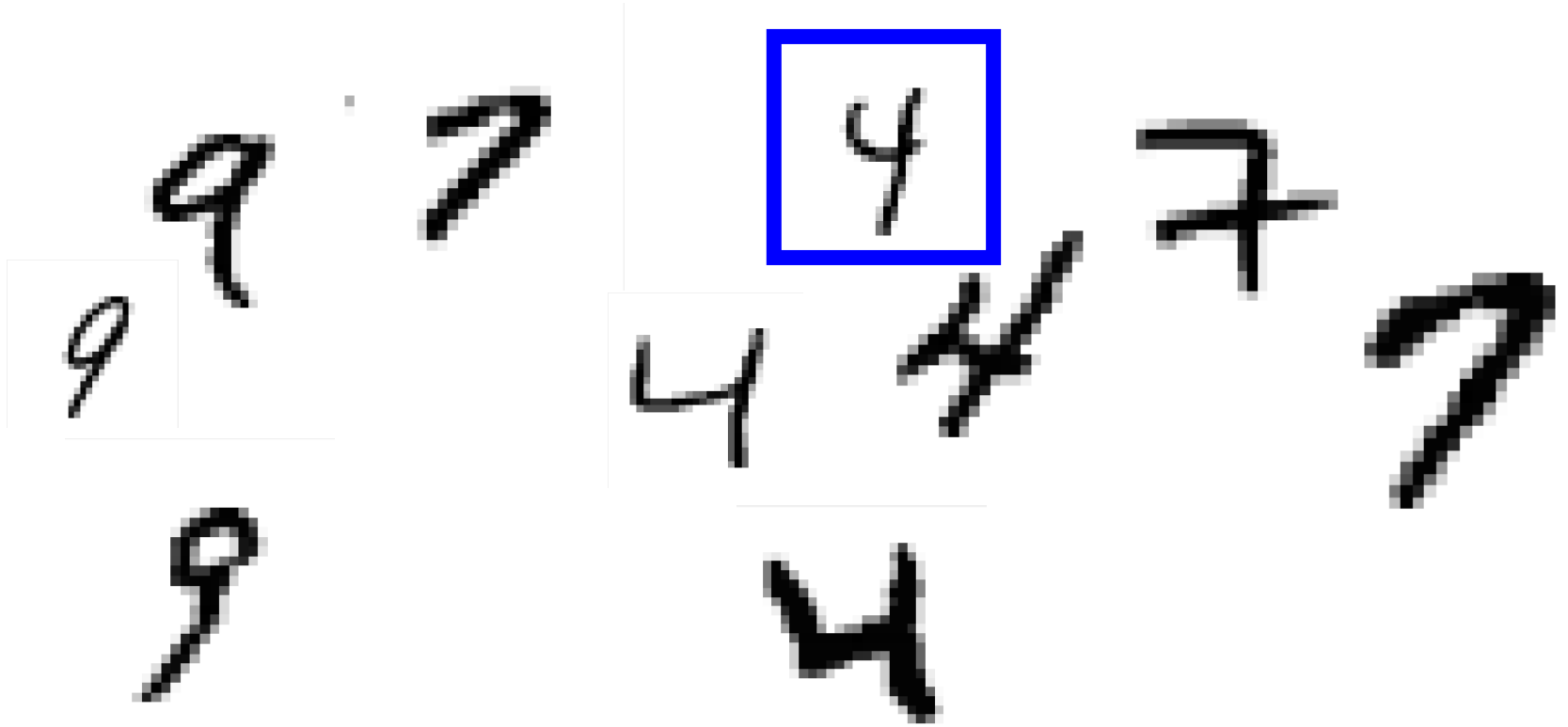
Each feature is grayscale intensity of one of the 784 pixels

y =

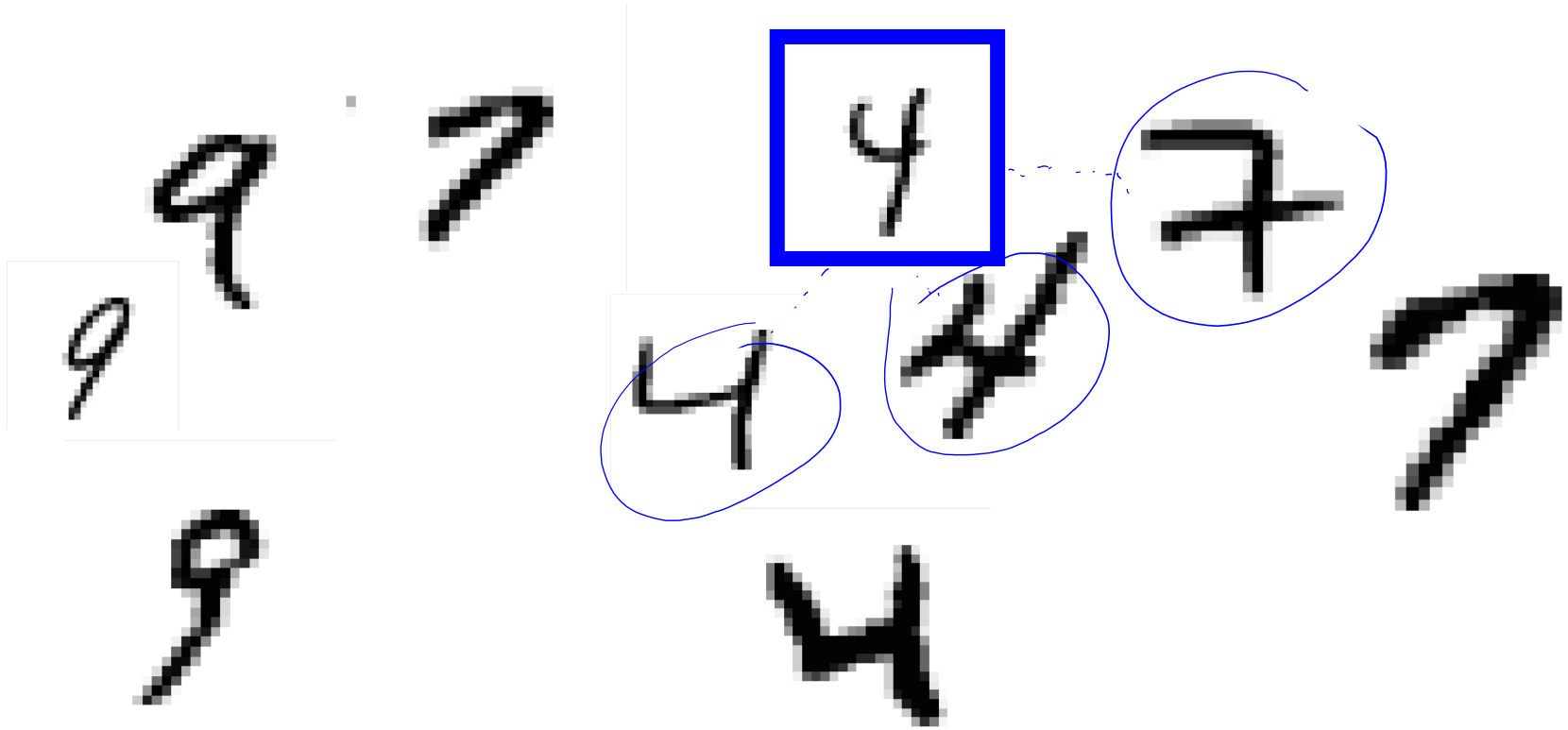
char
3
6
0
9

KNN for Optical Character Recognition

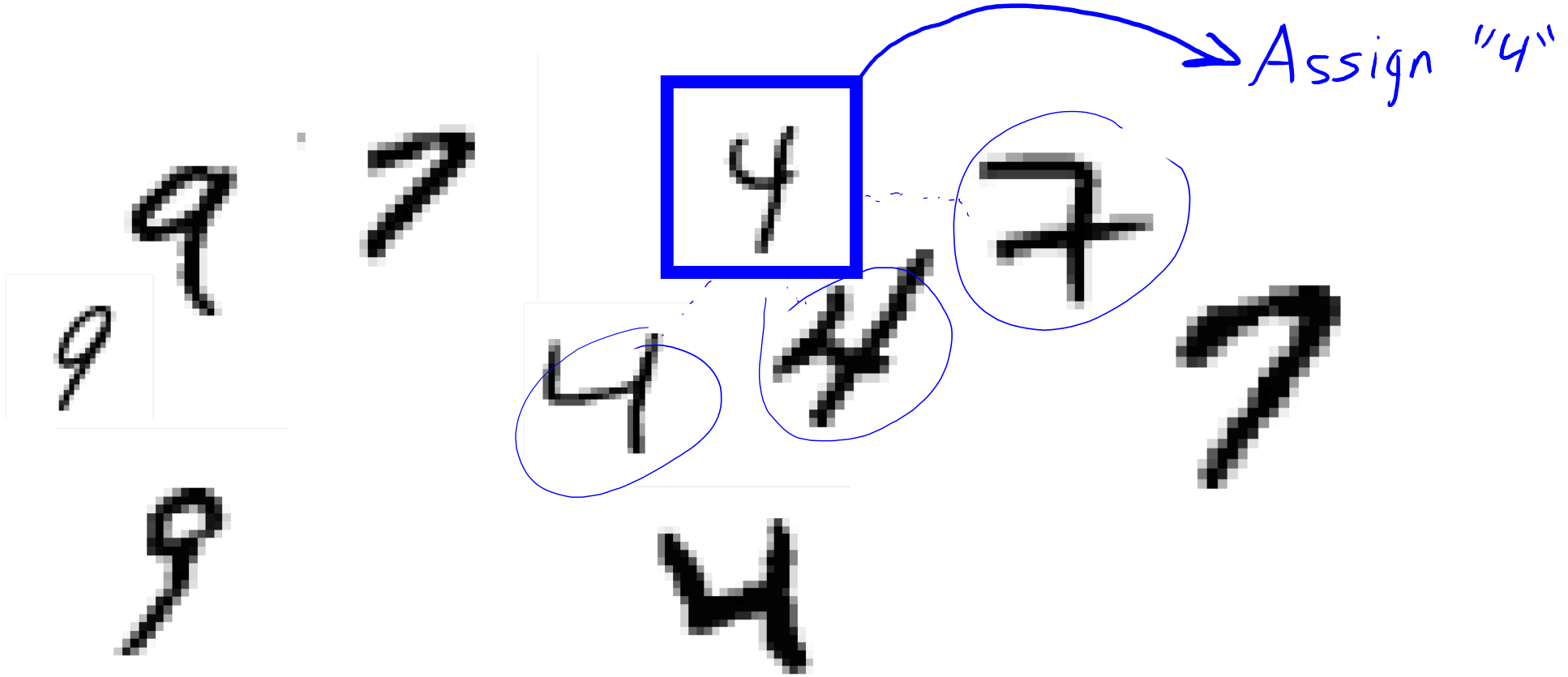
KNN for Optical Character Recognition



KNN for Optical Character Recognition



KNN for Optical Character Recognition



Human vs. Machine Perception

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

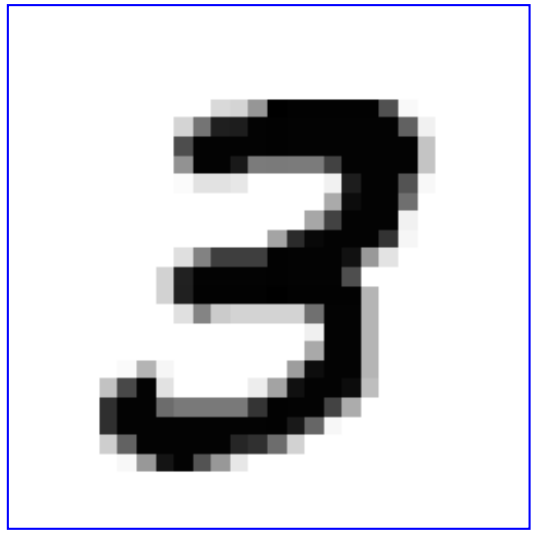
What we see:

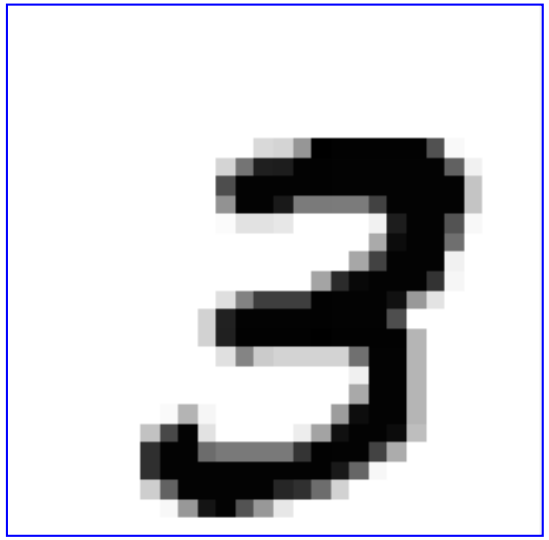
What the computer “sees”:

Actually, it’s worse:

What the Computer Sees

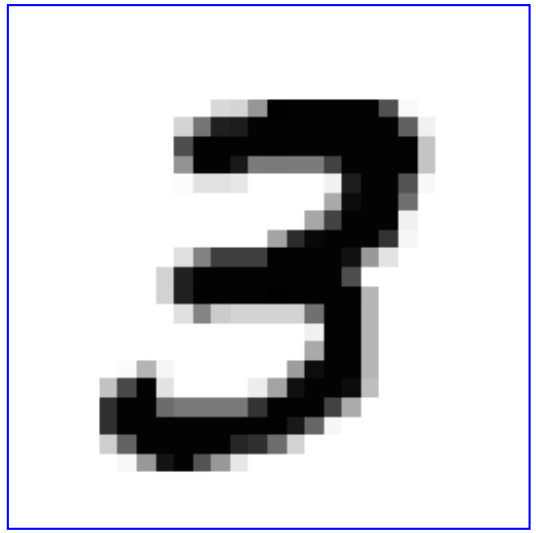
- Are these two images “similar”?

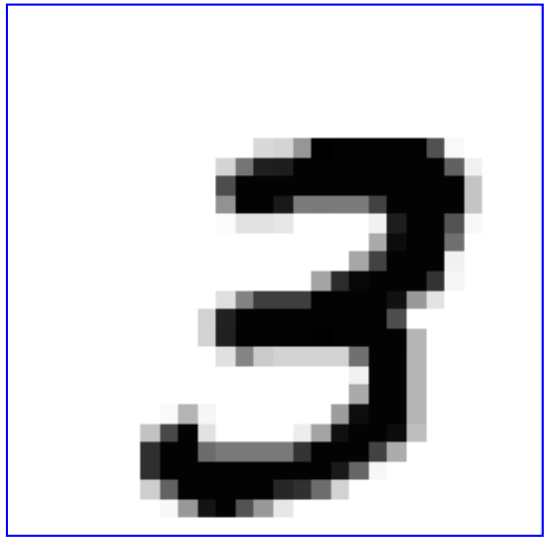




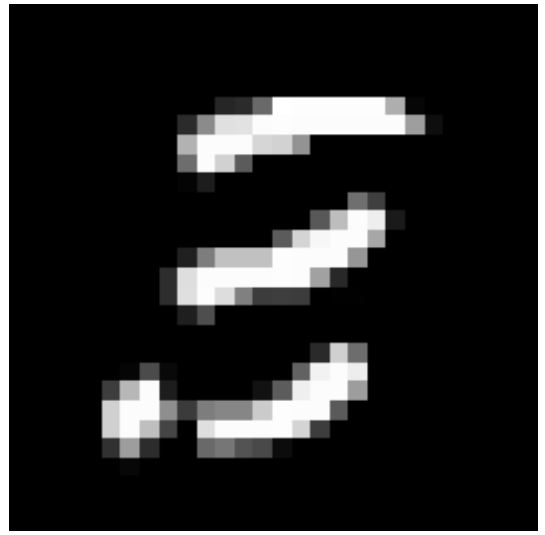
What the Computer Sees

- Are these two images “similar”?





Difference:

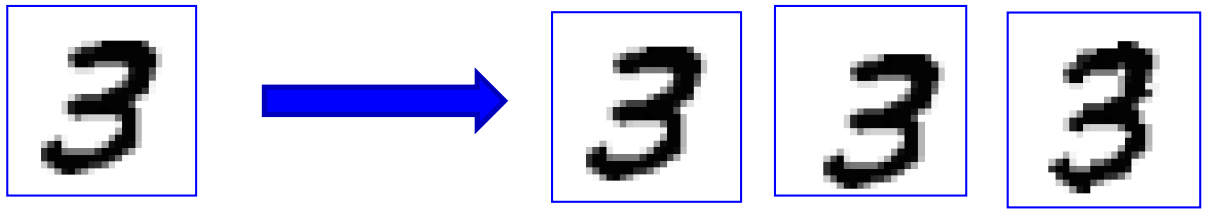


- KNN does not know that labels should be translation invariant.

Encouraging Invariance

- May want classifier to be invariant to certain feature transforms.
  - Images: 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.

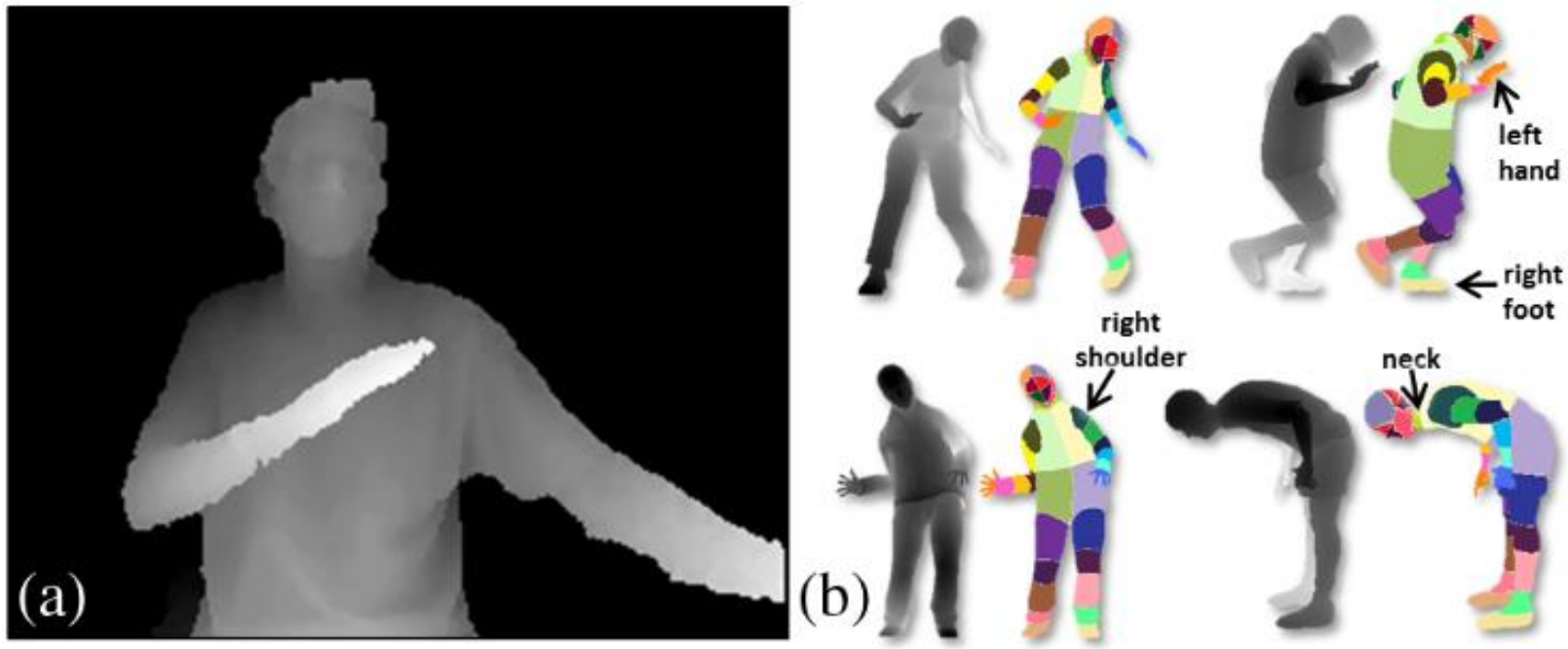
- The easy/fast way is to just add transformed data during training:
  - Add translated/rotate/resized/warped versions of training images.



  - Crucial part of many successful vision systems.
  - Also really important for sound (translate, change volume, and so on).

Application: Body-Part Recognition

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

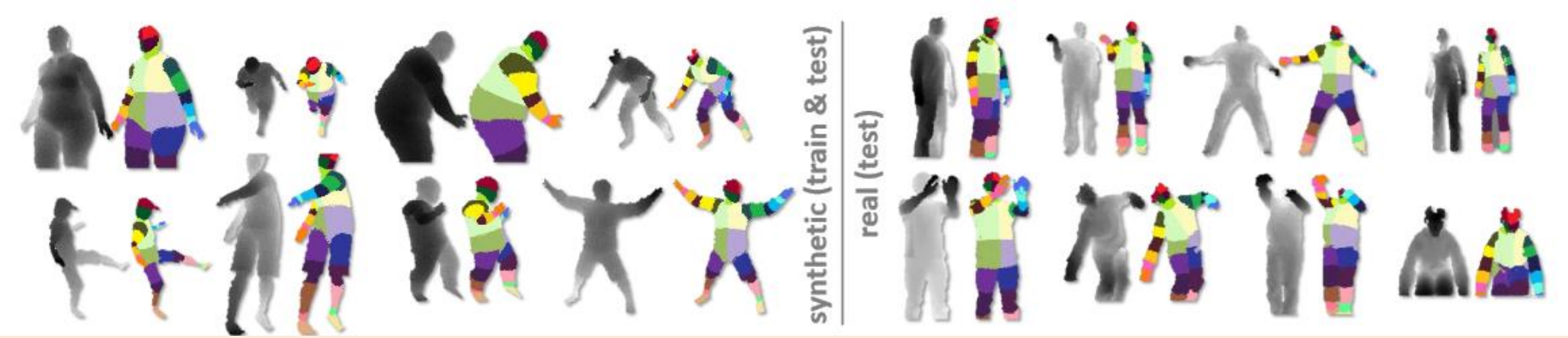


- How could we write a program to do this?

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, age, shape, clothing, and crop.
2. Build a simulator that fills space of images 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 body part of a 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 steps are important, but we'll focus on the 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”.
- They have the best names:
  - Averaging.
  - Boosting.
  - Bootstrapping.
  - Bagging.
  - Cascading.
  - Random Forests.
  - Stacking.
- Ensemble methods often have higher accuracy than input classifiers.

Ensemble Methods

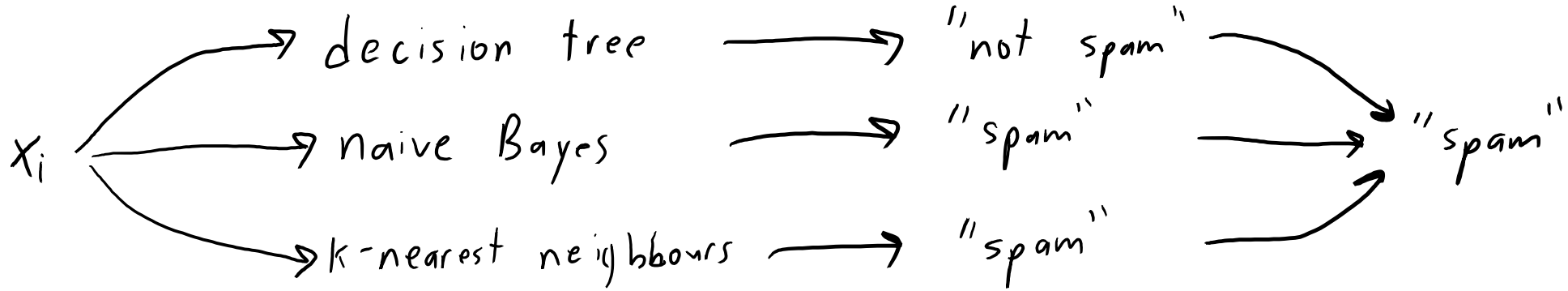
- Remember the fundamental trade-off:
  1. E_{train}: How small you can make the training error.

     vs.

  2. E_{approx}: 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.
- This suggests two types of ensemble methods:
  1. Boosting: improves training error of classifiers with high E_{train}.
  2. Averaging: improves approximation error of classifiers with high E_{approx}.

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.
- Simple model averaging:
  - Take the mode of the predictions (or average probabilities if probabilistic).



Why can Averaging Work?

- Consider 3 binary classifiers, each independently correct with probability 0.80:

- The ensemble will be correct if the mode of the is right ("3 right" or "2 right").
  - P(all 3 right) = 0.8^3 = 0.512.
  - P(2 rights, 1 wrong) = $3*0.8^2(1-0.8)$ = 0.384.
  - P(1 right, 2 wrongs) = $3*(1-0.8)^2 0.8$ = 0.096.
  - P(all 3 wrong) = $(1-0.8)^3$ = 0.008.
  - So ensemble is right with probability 0.896 (which is 0.512+0.384).

- Notes:
  - For averaging to work, classifiers need to be at least somewhat independent.
  - You also want the probability of being right to be > 0.5, otherwise it will do much worse.
  - Probabilities also shouldn't be to different (otherwise, it might be better to take most accurate).

Averaging

- Consider a set of classifiers that make these predictions:
  - Classifier 1: “spam”.
  - Classifier 2: “spam”.
  - Classifier 3: “spam”.
  - Classifier 4: “not spam”.
  - Classifier 5: “spam”.
  - Classifier 6: “not spam”.
  - Classifier 7: “spam”.
  - Classifier 8: “spam”.
  - Classifier 9: “spam”.
  - Classifier 10: “spam”.
- If these independently get 80% accuracy, mode will be close to 100%.
  - In practice errors won’t be completely independent (due to noise in labels).

Why can Averaging Work?

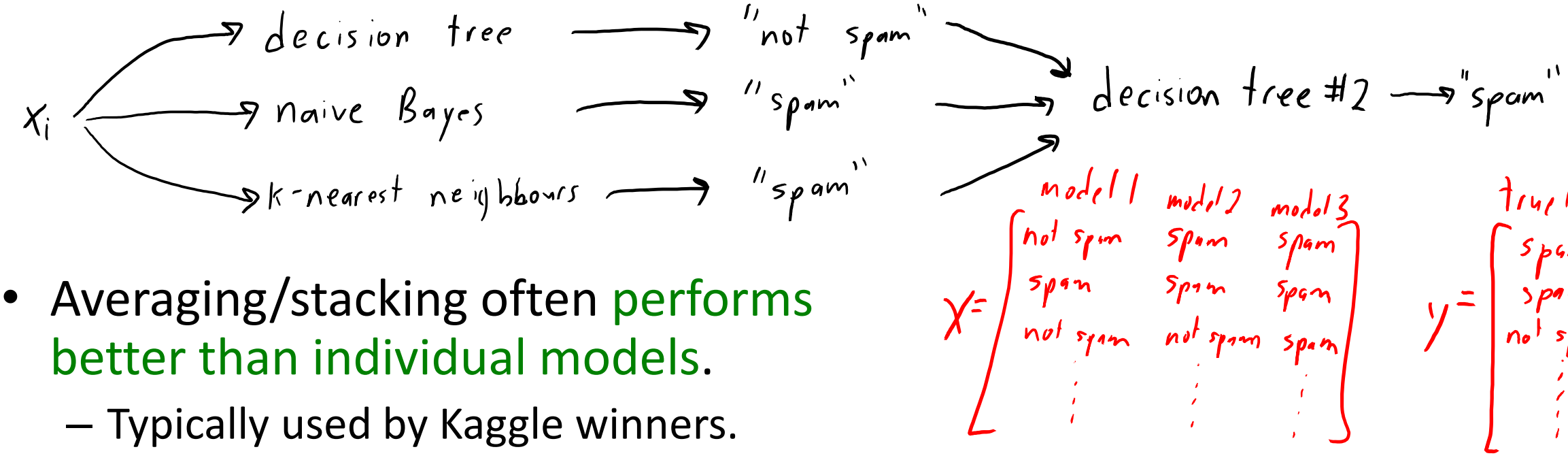
- Why can averaging lead to better results?

- Consider classifiers that overfit (like deep decision trees):
  - If they all overfit in exactly the same way, averaging does nothing.

- But if they make independent errors:
  - Probability that “average” is wrong can be lower than for each classifier.
  - Less attention to specific overfitting of each classifier.

Learning to Average: Stacking

- Stacking:
  - Fit another classifier that uses the predictions as features.



- Averaging/stacking often performs better than individual models.
  - Typically used by Kaggle winners.
  - E.g., Netflix $1M user-rating competition winner was stacked classifier.

Random Forests

- Random forests average a set of deep decision trees.
  - Tend to be one of the best "out of the box" classifiers.
    - Often close to the best performance of any method on the first run.
  - And predictions are very fast.

- Do deep decision trees make independent errors?
  - No: with the same training data you'll get the same decision tree.

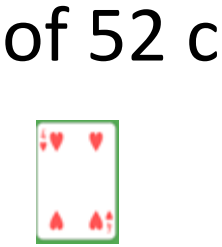
- Two key ingredients in random forests:
  - Bootstrapping.
  - Random trees.

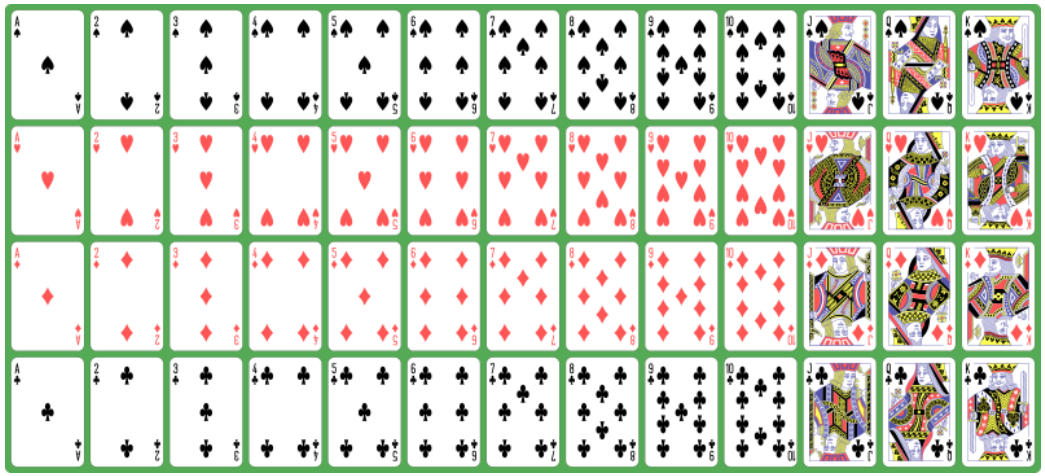
Boostrap Sampling

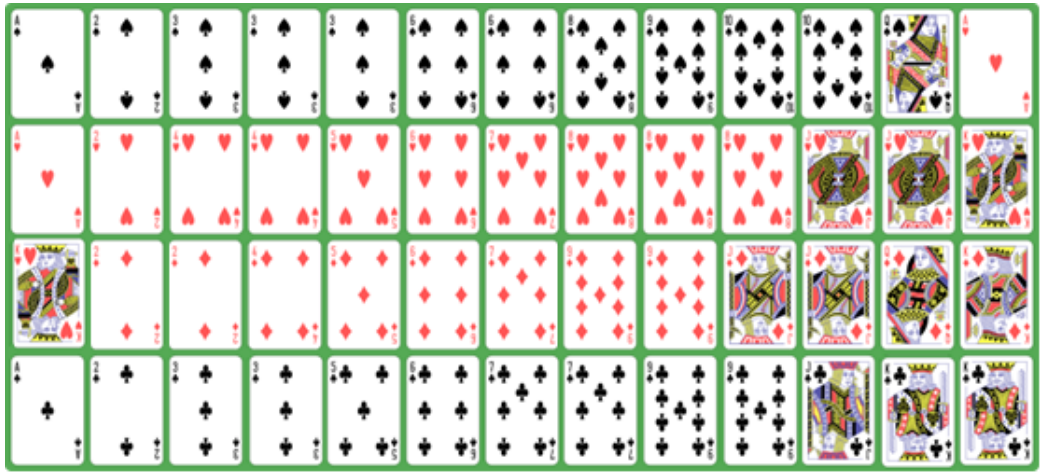
- Start with a standard deck of 52 cards:
  1. Sample a random card:
     (put it back in the deck)
  2. Sample a random card:
     (put it back in the deck)
  3. Sample a random card:
     (put it back in the deck)

  – …

  52. Sample a random card:
      (which may be a repeat)
- We now have a new deck of 52 cards:



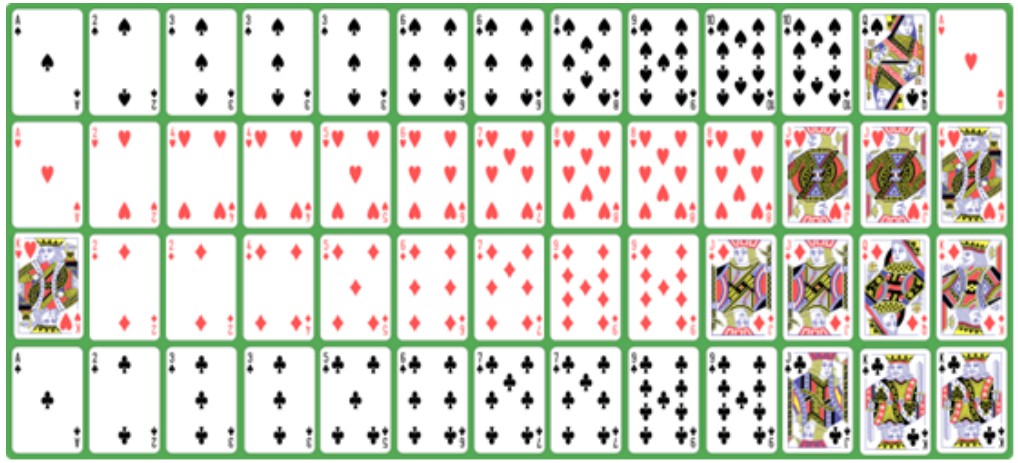




https://commons.wikimedia.org/wiki/File:English_pattern_playing_cards_deck.svg

Bootstrap Sampling

- New 52-card deck is called a "bootstrap sample":



  - Some cards will be missing, and some cards will be duplicated.
    - So calculations on the bootstrap sample will give different results than original data.
  - However, the bootstrap sample roughly maintains trends:
    - Roughly 25% of the cards will be diamonds.
    - Roughly 3/13 of the cards will be "face" cards.
    - There will be roughly four "10" cards.
  - Common use: compute a statistic based on several bootstrap samples.
    - Gives you an idea of how the statistic varies as you vary the data.

Random Forest Ingredient 1: Bootstrap

- Bootstrap sample of a list of 'n' examples:
  - A new set of size 'n' chosen independently with replacement.

```
for i in 1:n
    j = rand(1:n)   # pick a random number from {1,2,...,n}
    X_bootstrap[i,:] = X[j,:]   # use the random sample
```

  - Gives new dataset of 'n' examples, with some duplicated and some missing.
    - For large 'n', approximately 63% of original examples are included.

- Bagging: using bootstrap samples for ensemble learning.
  - Generate several bootstrap samples of the examples (x_i, y_i).
  - Fit a classifier to each bootstrap sample.
  - At test time, average the predictions.

} Decision trees will make different splits.

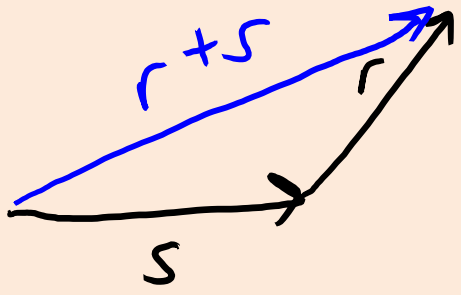
Summary

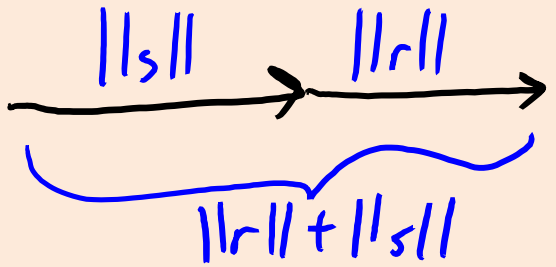
- Encouraging invariance:
  - Add transformed data to be insensitive to the transformation.
- Ensemble methods take classifiers as inputs.
  - Try to reduce either E_{train} or E_{approx} without increasing the other much.
  - "Boosting" reduces E_{train} and "averaging" reduces E_{approx}.
- Averaging:
  - Improves predictions of multiple classifiers if errors are independent.
- Random forests:
  - Averaging of deep randomized decision trees.
  - One of the best "out of the box" classifiers.

- Next time:
  - We start unsupervised learning.

3 Defining Properties of Norms

- A “norm” is any function satisfying the following 3 properties:
  1. Only ‘0’ has a ‘length’ of zero.
  2. Multiplying ‘r’ by constant ‘α’ multiplies length by |α|
     - “If be will twice as long if you multiply by 2”: ||αr|| = |α|•||r||.
     - Implication is that norms cannot be negative.
  3. Length of ‘r+s’ is not more than length of ‘r’ plus length of ‘s’:
     - “You can’t get there faster by a detour”.
     - “Triangle inequality”: ||r + s|| ≤ ||r|| + ||s||.





Squared/Euclidean-Norm Notation

We're using the following conventions:

The subscript after the norm is used to denote the p-norm, as in these examples:

$$\|x\|_2 = \sqrt{\sum_{j=1}^{d} w_j^2}.$$
$$\|x\|_1 = \sum_{j=1}^{d} |w_j|.$$

If the subscript is omitted, we mean the 2-norm:

$$\|x\| = \|x\|_2.$$

If we want to talk about the *squared* value of the norm we use a superscript of "2":

$$\|x\|_2^2 = \sum_{j=1}^{d} w_j^2.$$
$$\|x\|_1^2 = \left(\sum_{j=1}^{d} |w_j|\right)^2.$$

If we omit the subscript and have a superscript of "2", we're taking about the squared L2-norm:

$$\|x\|^2 = \sum_{j=1}^{d} w_j^2.$$

Lp-norms

- The L_1-, L_2-, and L_∞-norms are special cases of Lp-norms:

$$\|x\|_p = \left(\sum_{j=1}^{d} x_j^p \right)^{1/p}$$

- This gives a norm for any (real-valued) $p \geq 1$.
  - The L_∞-norm is limit as 'p' goes to ∞.

- For $p < 1$, not a norm because triangle inequality not satisfied.

Extremely-Randomized Trees

- Extremely-randomized trees add an extra level of randomization:
  1. Each tree is fit to a bootstrap sample.
  2. Each split only considers a random subset of the features.
  3. Each split only considers a random subset of the possible thresholds.

- So instead of considering up to ‘n’ thresholds, only consider 10 or something small.
  - Leads to different partitions so potentially more independence.

Why does Bootstrapping select approximately 63%?

- Probability of an arbitrary x_i being selected in a bootstrap sample:

p(selected at least once in 'n' trials)

$= 1 -$ p(not selected in any of 'n' trials)

$= 1 - ($p(not selected in one trial)$)^n$

$= 1 - (1 - 1/n)^n$

$\approx 1 - 1/e$

≈ 0.63

(trials are independent)

(prob = $\frac{n-1}{n}$ for choosing any of the n-1 other samples)

($(1 - 1/n)^n \to e^{-1}$ as $n \to \infty$)

Why Random Forests Work

- Consider ‘k’ independent classifiers, whose errors have a variance of σ^2.
- If the errors are IID, the variance of the average is σ^2/k.
  - So the more classifiers you average, the more you decrease error variance. (And the more the training error approximates the test error.)
- Generalization to case where classifiers are not independent is:

$$c\sigma^2 + \frac{(1-c)}{k}\sigma^2$$

  - Where ‘c’ is the correlation.
- So the less correlation you have the closer you get to independent case.
- Randomization in random forests decreases correlation between trees.
  - See also “Sensitivity of Independence Assumptions”.

How these concepts often show up in practice

- Here is a recent e-mail related to many ideas we've recently covered:
  - "However, the performance did not improve while the model goes deeper and with augmentation. The best result I got on validation set was 80% with LeNet-5 and NO augmentation (LeNet-5 with augmentation I got 79.15%), and later 16 and 50 layer structures both got 70%~75% accuracy.

    In addition, there was a software that can use mathematical equations to extract numerical information for me, so I trained the same dataset with nearly 100 features on random forest with 500 trees. The accuracy was 90% on validation set.

    I really don't understand that how could deep learning perform worse as the number of hidden layers increases, in addition to that I have changed from VGG to ResNet, which are theoretically trained differently. Moreover, why deep learning algorithm cannot surpass machine learning algorithm?"
- Above there is data augmentation, validation error, effect of the fundamental trade-off, the no free lunch theorem, and the effectiveness of random forests.

Bayesian Model Averaging

- Recall the key observation regarding ensemble methods:
  - If models overfit in "different" ways, averaging gives better performance.
- But should all models get equal weight?
  - E.g., decision trees of different depths, when lower depths have low training error.
  - E.g., a random forest where one tree does very well (on validation error) and others do horribly.
  - In science, research may be fraudulent or not based on evidence.
- In these cases, naïve averaging may do worse.

Bayesian Model Averaging

- Suppose we have a set of 'm' probabilistic binary classifiers w_j.
- If each one gets equal weight, then we predict using:

$$p(y_i \mid x_i) = \frac{1}{m} p(y_i \mid w_1, x_i) + \frac{1}{m} p(y_i \mid w_2, x_i) + \cdots + \left(\frac{1}{m}\right) p(y_i \mid w_m, x_i)$$

- Bayesian model averaging treats model 'w_j' as a random variable:

Assume $w_j \perp x_i$

$$p(y_i \mid x_i) = \sum_{j=1}^{m} p(y_i, w_j \mid x_i) = \sum_{j=1}^{m} p(y_i \mid w_j, x_j) p(w_j \mid x_j) = \sum_{j=1}^{m} p(y_i \mid w_j, x_j) p(w_j)$$

- So we should weight by probability that w_j is the correct model:
  - Equal weights assume all models are equally probable.

Bayesian Model Averaging

Again, assuming $w_j \perp X$

- Can get better weights by conditioning on training set:

$$p(w_j \mid X, y) \propto p(y \mid w_j, X)\, p(w_j \mid X) = p(y \mid w_j, X)\, p(w_j)$$

- The 'likelihood' $p(y \mid w_j, X)$ makes sense:
  - We should give more weight to models that predict 'y' well.
  - Note that hidden denominator penalizes complex models.
- The 'prior' $p(w_j)$ is our 'belief' that w_j is the correct model.
- This is how rules of probability say we should weigh models.
  - The 'correct' way to predict given what we know.
  - But it makes some people unhappy because it is subjective.