CSE419 – Artificial Intelligence and Machine Learning 2020

PhD Furkan Gözükara, Toros University

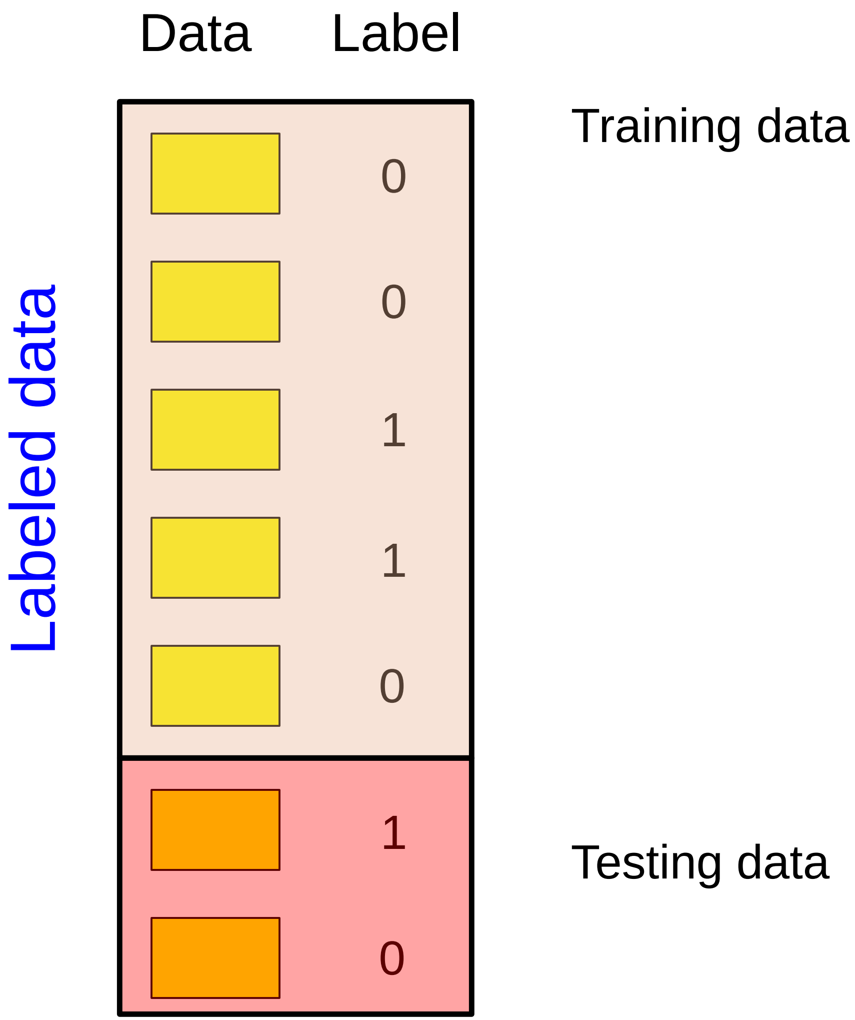
https://github.com/FurkanGozukara/CSE419-Artificial-Intelligence-and-Machine-Learning-2020

Lecture 9 Part 1

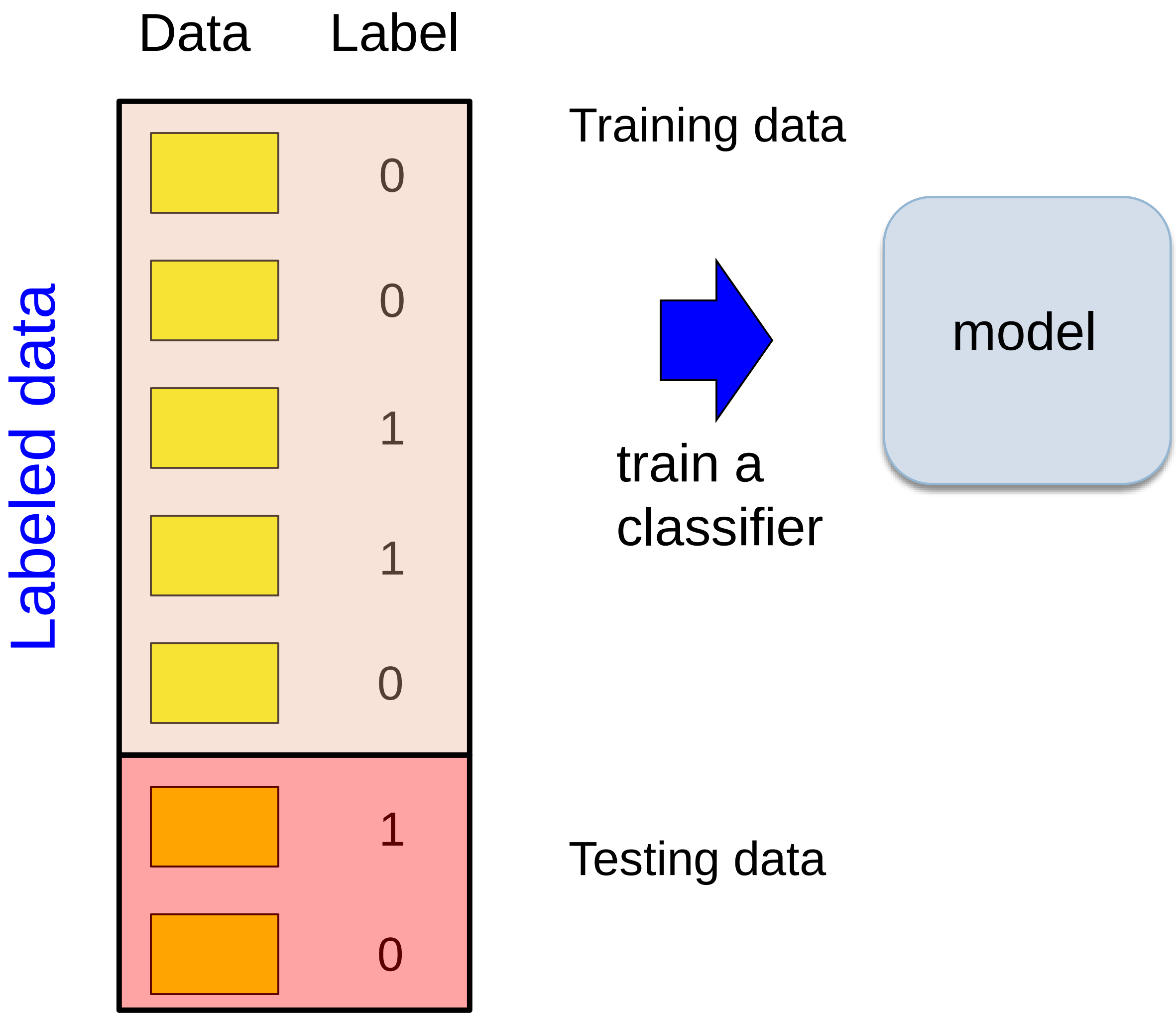
Evaluation

Based on Asst. Prof. Dr. David Kauchak (Pomona College) Lecture Slides

Supervised evaluation

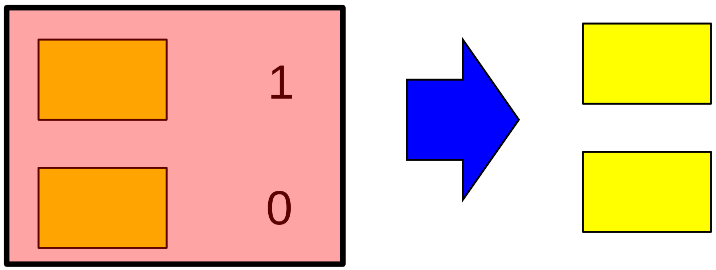


Supervised evaluation



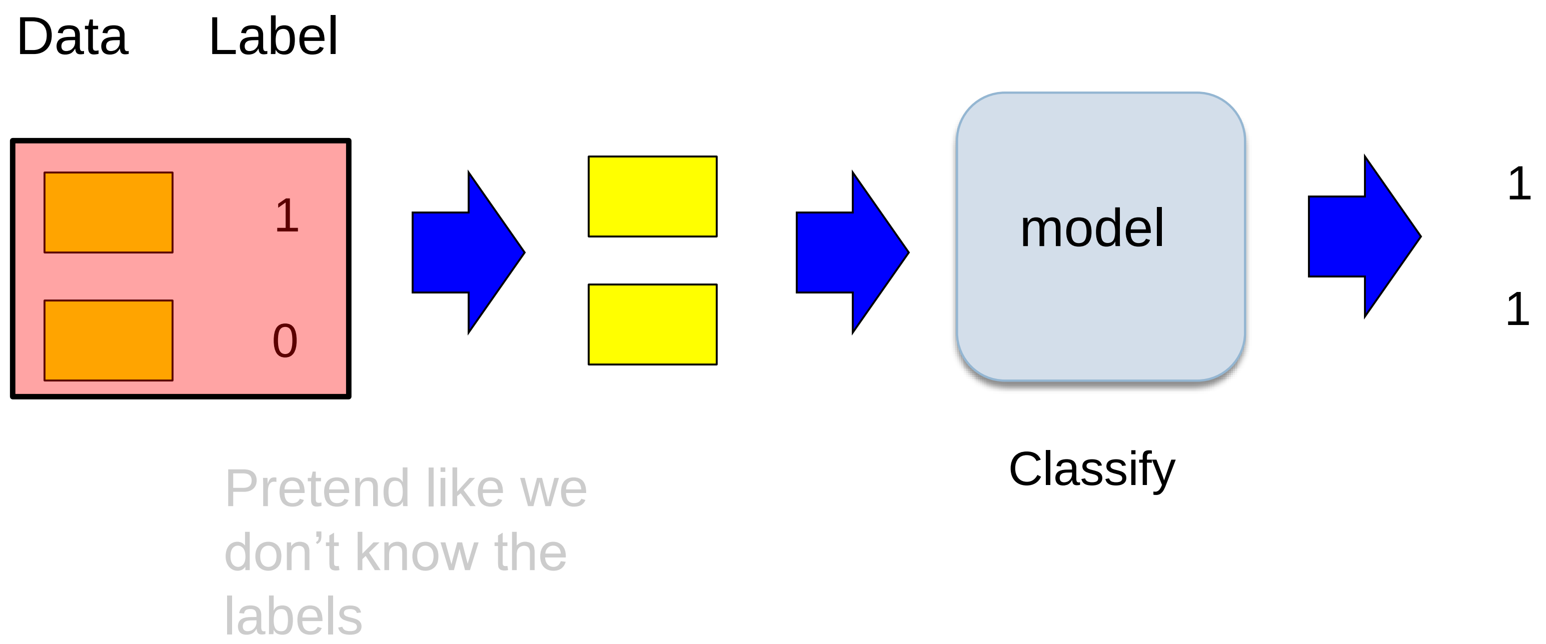
Supervised evaluation

Data Label

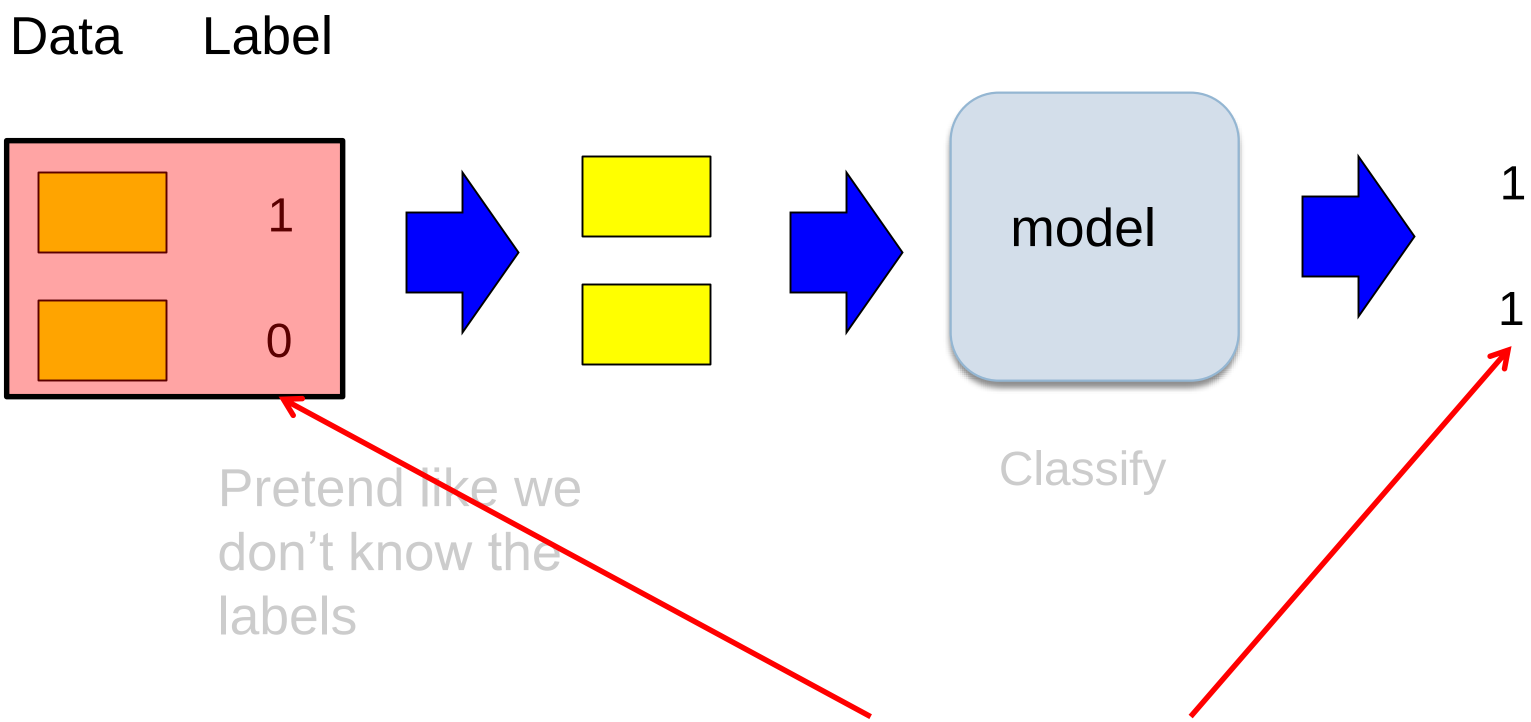


Pretend like we don’t know the labels

Supervised evaluation

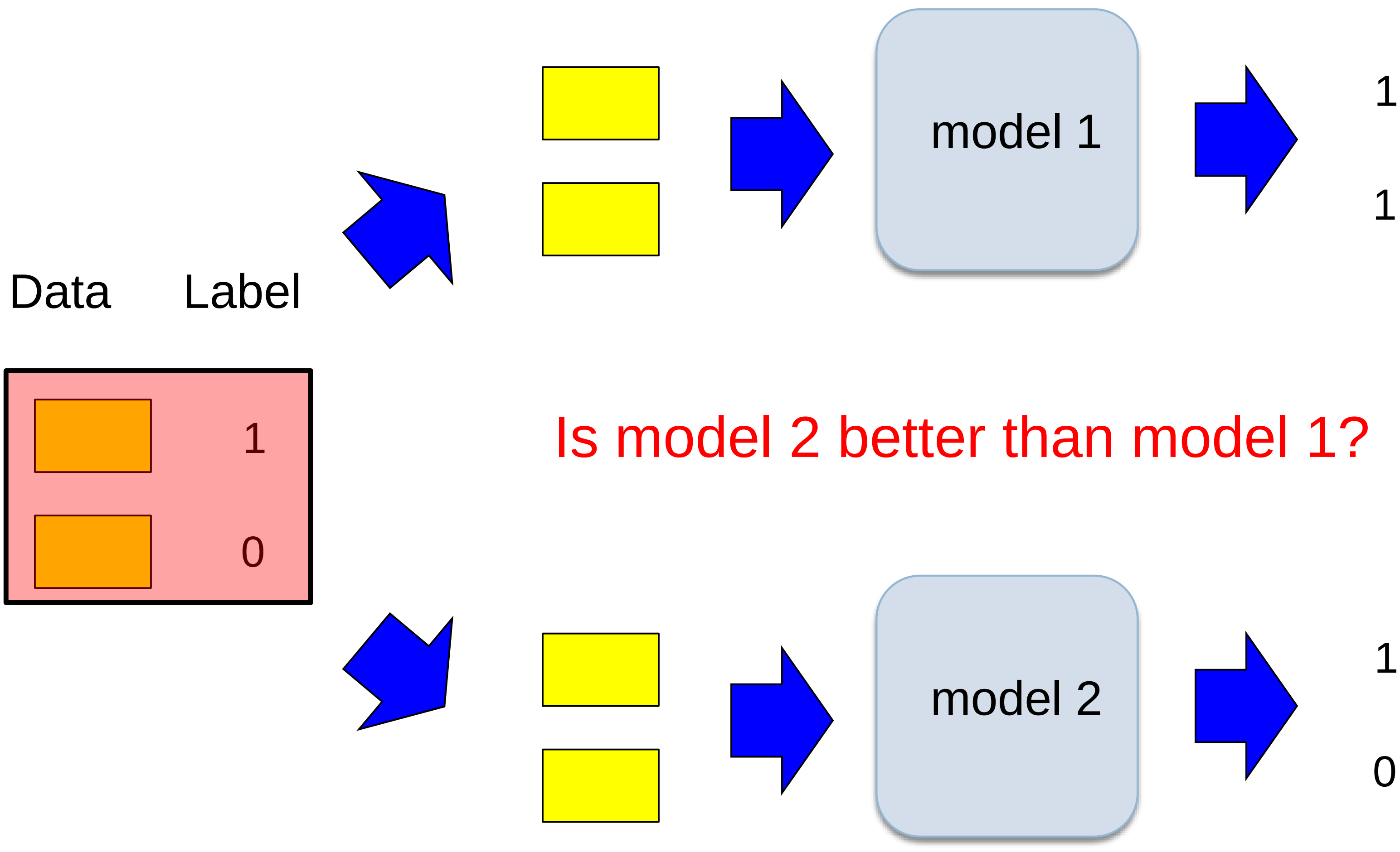


Supervised evaluation

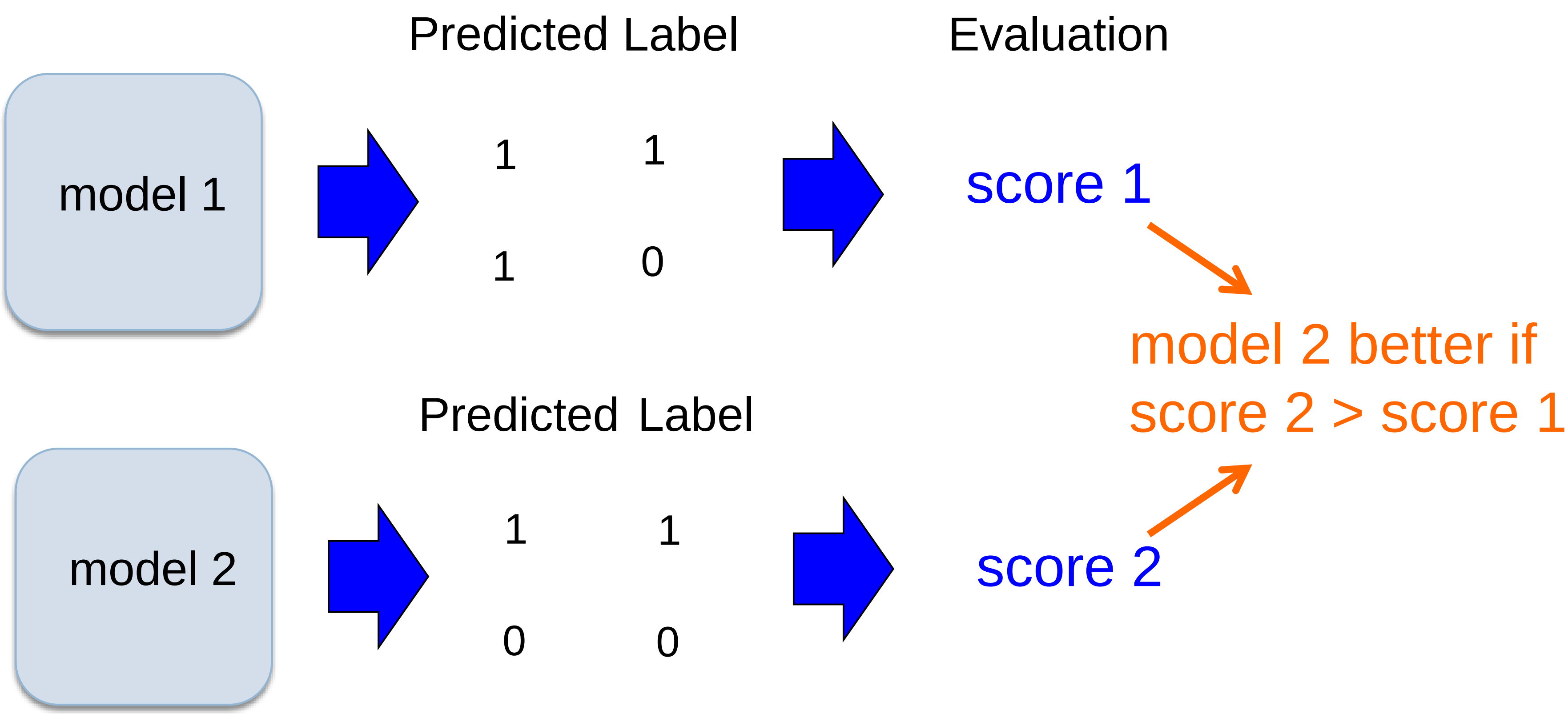


Compare predicted labels to actual labels

Comparing algorithms

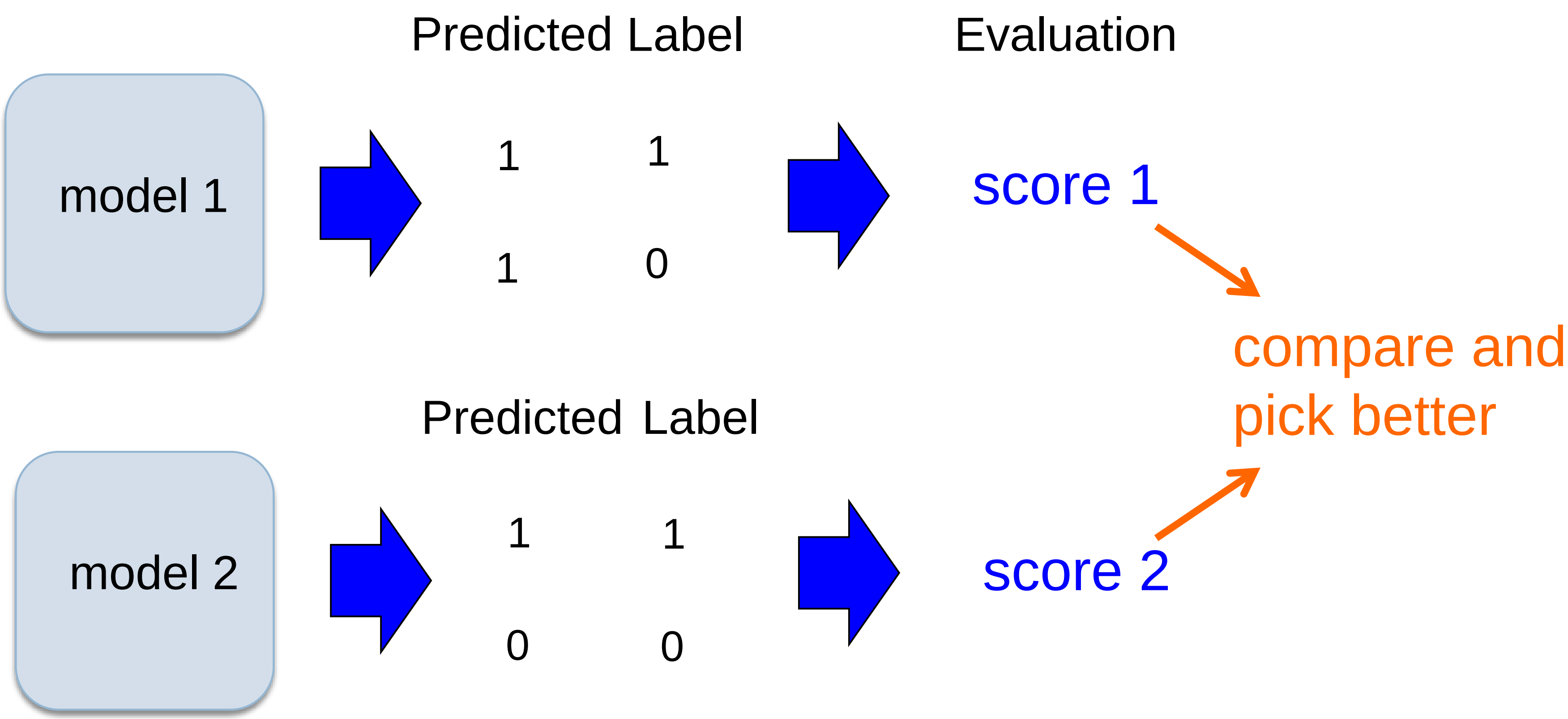


Idea 1



When would we want to do this type of comparison?

Idea 1



Any concerns?

Is model 2 better?

Model 1: 85% accuracy
Model 2: 80% accuracy

Model 1: 85.5% accuracy
Model 2: 85.0% accuracy

Model 1: 0% accuracy
Model 2: 100% accuracy

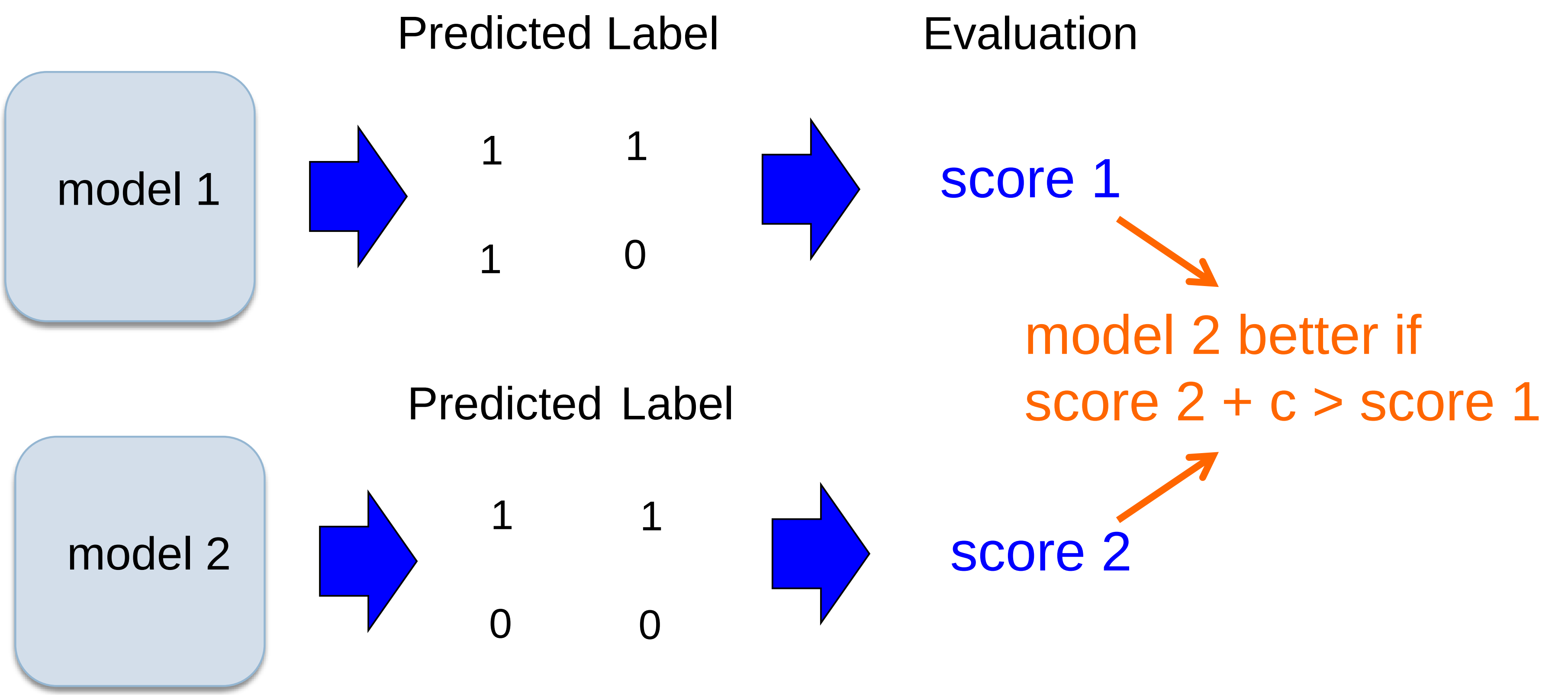
Comparing scores: significance

Just comparing scores on one data set isn't enough!

We don't just want to know which system is better on ***this particular data***, we want to know if model 1 is better than model 2 ***in general***

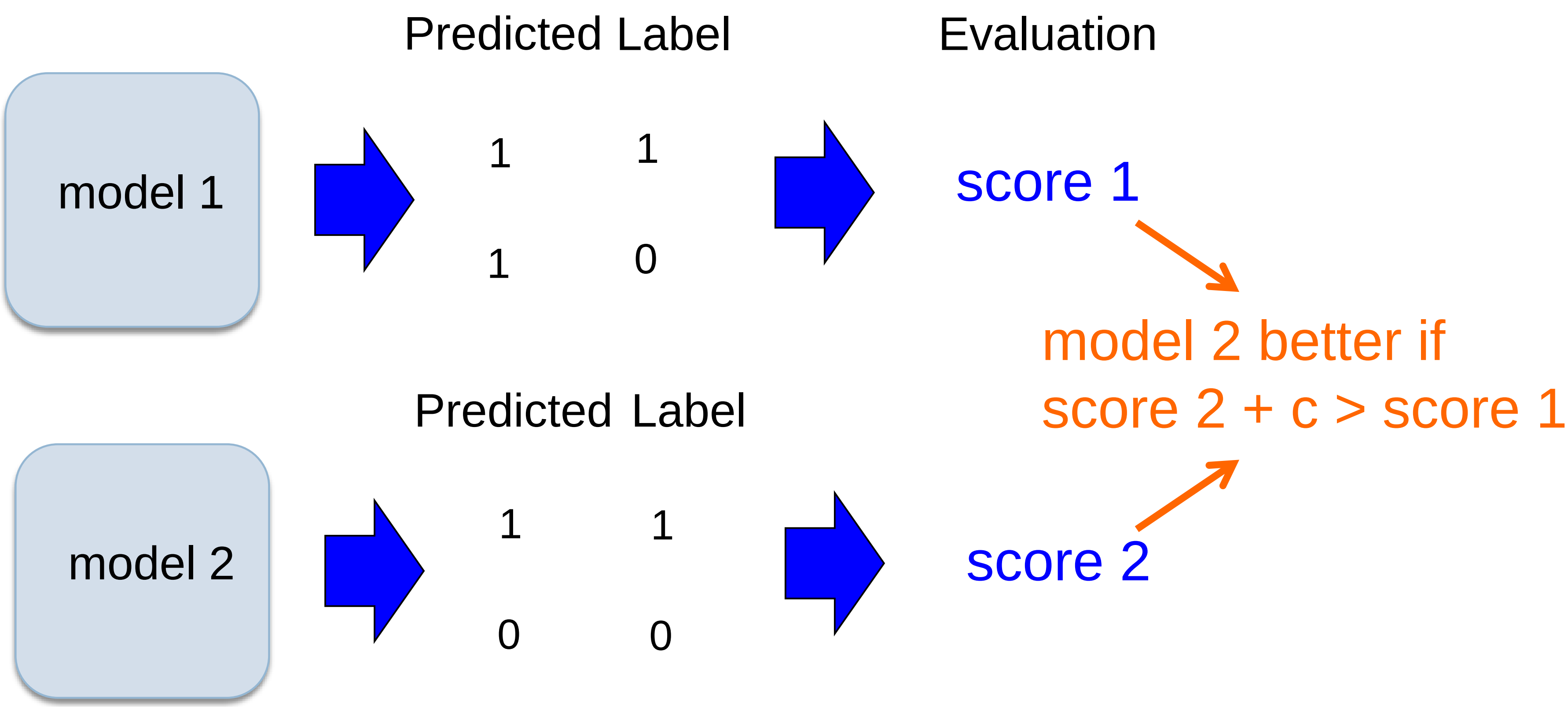
Put another way, we want to be confident that the difference is real and not just do to random chance

Idea 2



Is this any better?

Idea 2

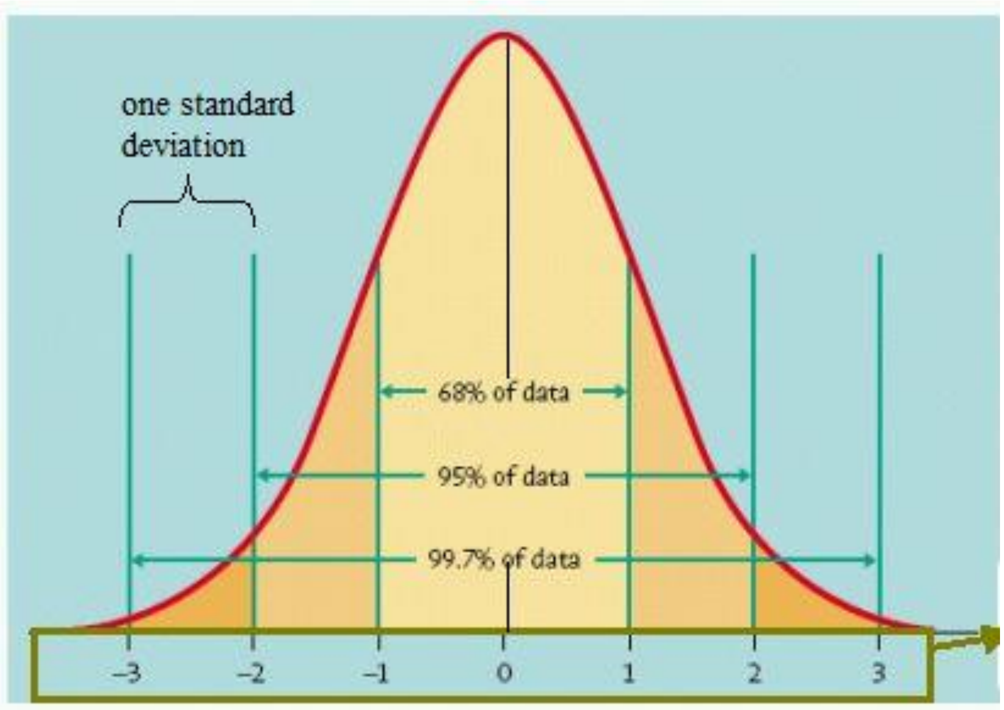


NO!

Key: we don’t know the variance of the output

Variance

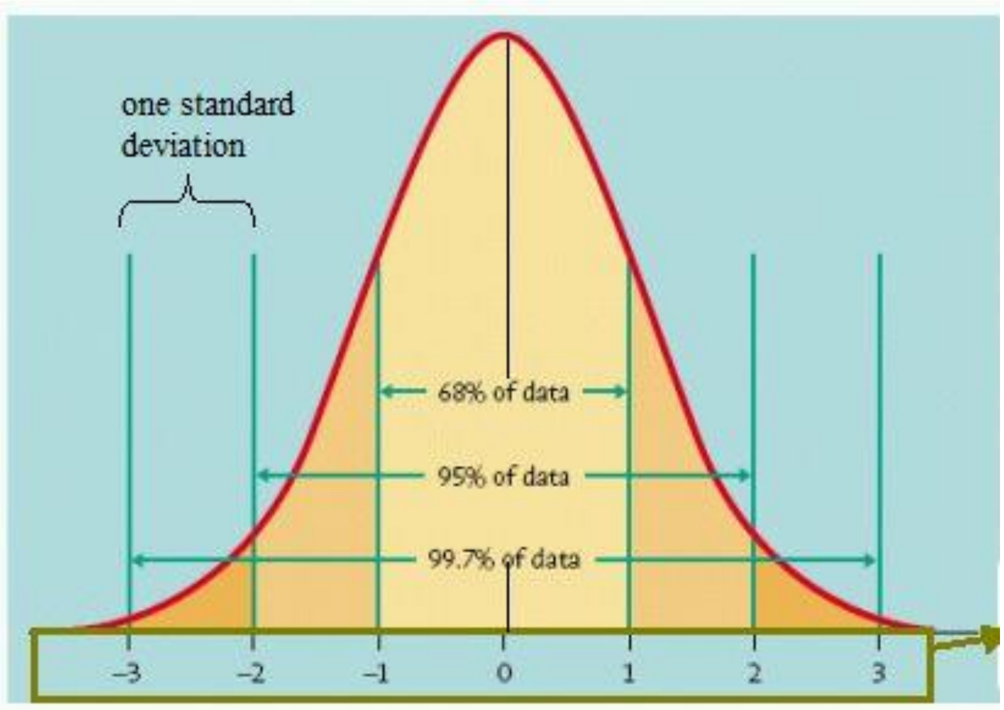
Recall that variance (or standard deviation) helped us predict how likely certain events are:



How do we know how variable a model's accuracy is?

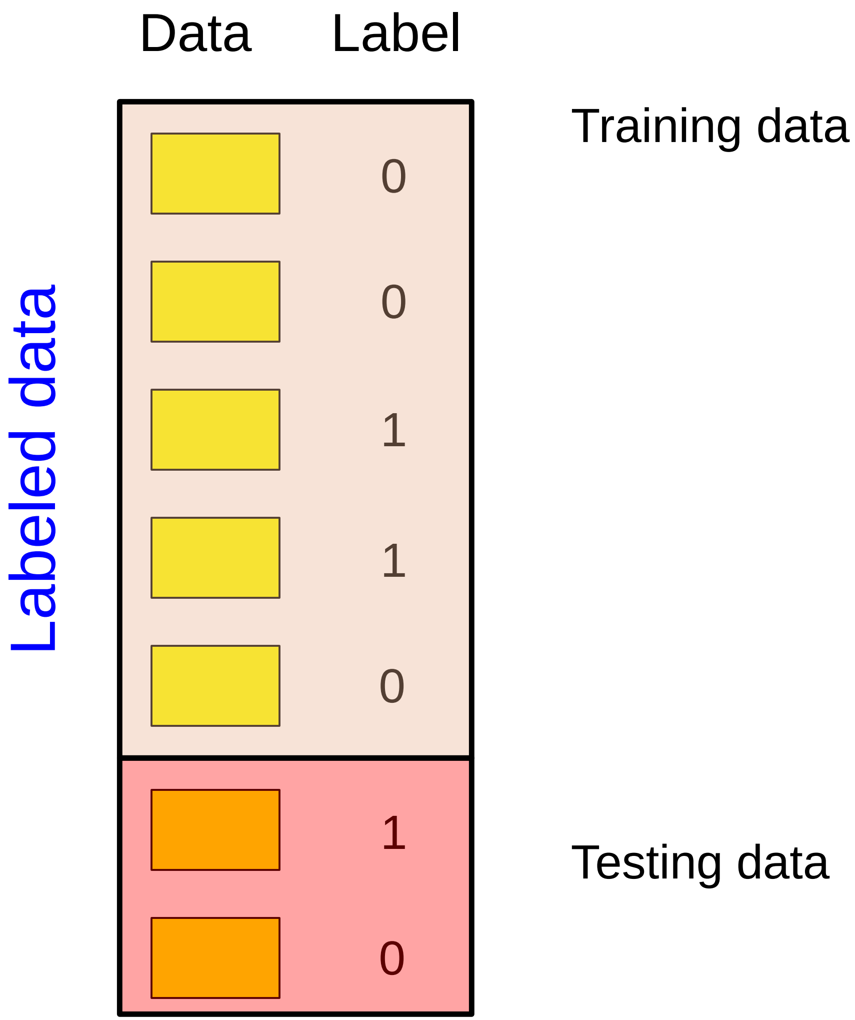
Variance

Recall that variance (or standard deviation) helped us predict how likely certain events are:



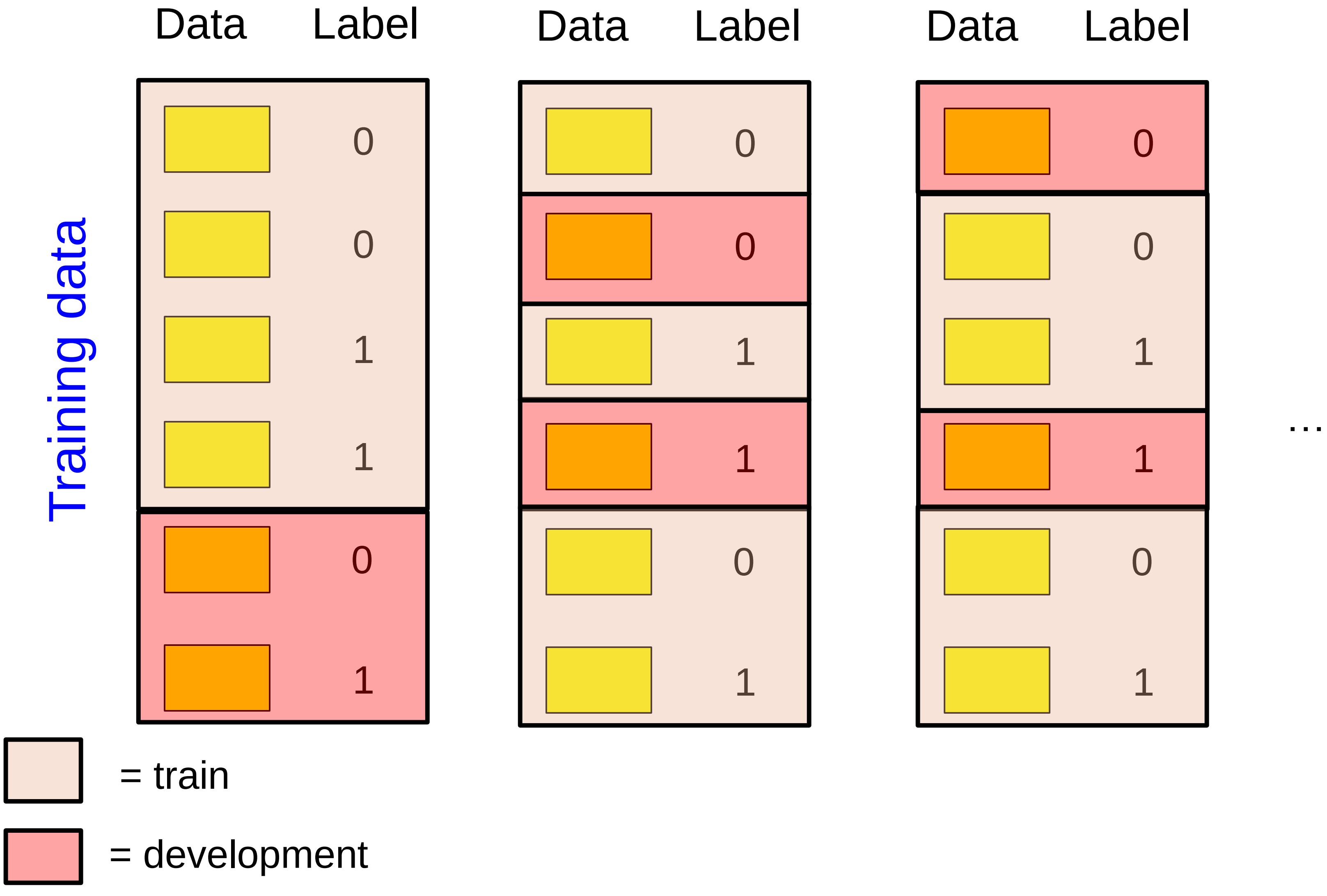
We need multiple accuracy scores! Ideas?

Repeated experimentation

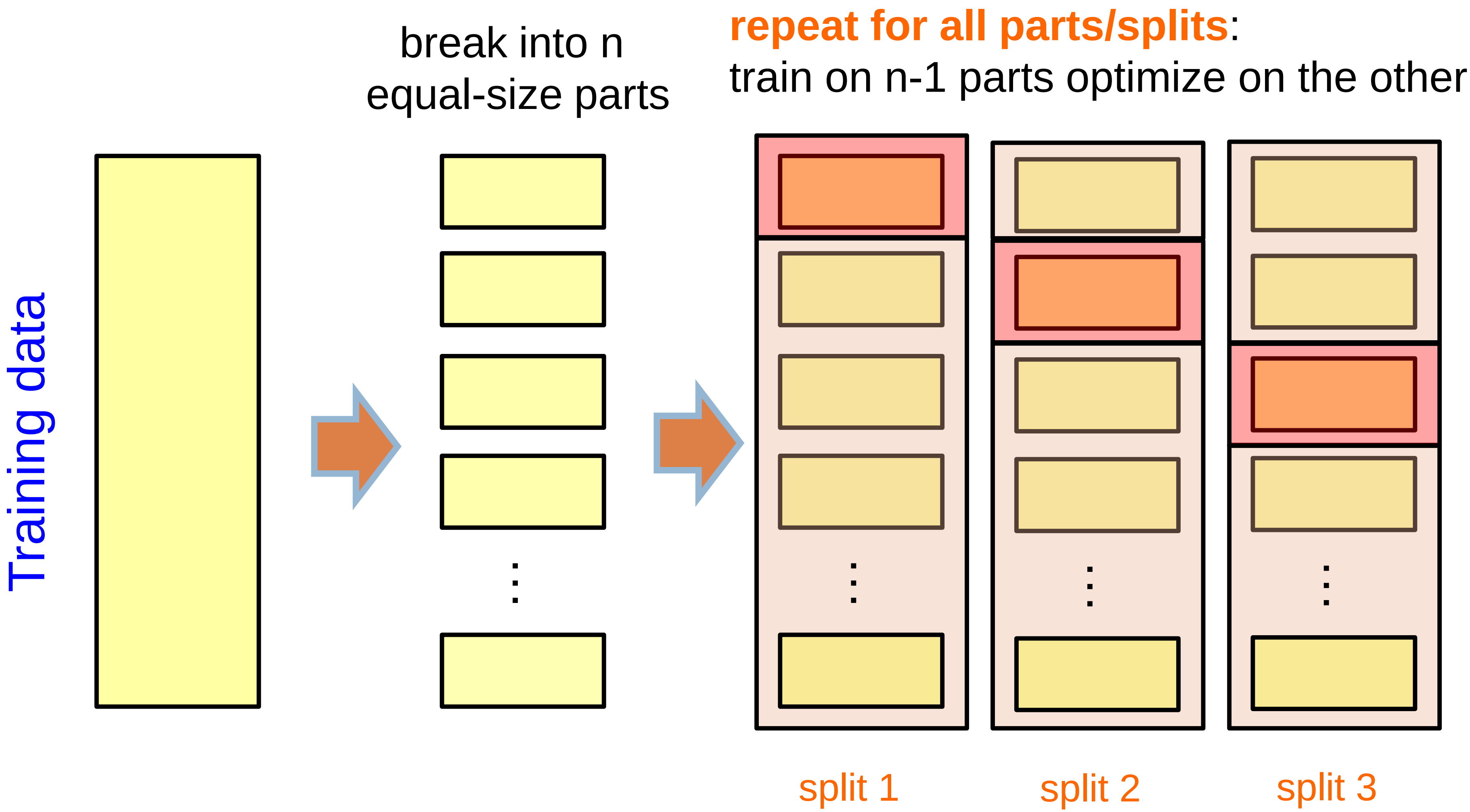


Rather than just splitting once, split multiple times

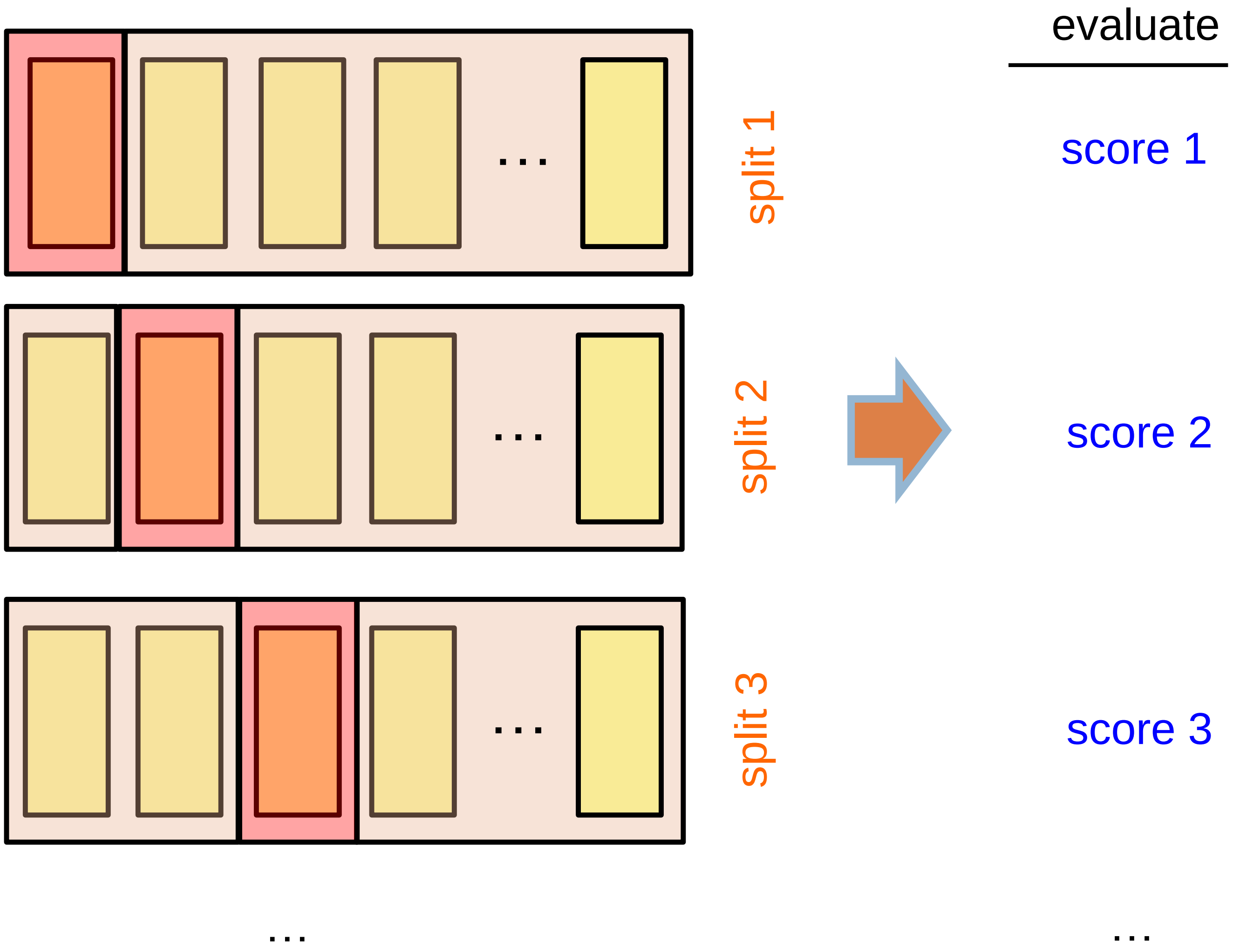
Repeated experimentation



n-fold cross validation



n-fold cross validation



n-fold cross validation

better utilization of labeled data

more robust: don’t just rely on one test/development set to evaluate the approach (or for optimizing parameters)

multiplies the computational overhead by n (have to train n models instead of just one)

10 is the most common choice of n

Leave-one-out cross validation

n-fold cross validation where n = number of examples

aka “jackknifing”

pros/cons?

when would we use this?

Leave-one-out cross validation

Can be very expensive if training is slow and/or if there are a large number of examples

Useful in domains with limited training data: *maximizes the data we can use for training*

Comparing systems: sample 1

split	model 1	model 2
1	87	88
2	85	84
3	83	84
4	80	79
5	88	89
6	85	85
7	83	81
8	87	86
9	88	89
10	84	85
average:	**85**	**85**

Is model 2 better than model 1?

Comparing systems: sample 2

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
average:	**82**	**85**

Is model 2 better than model 1?

Comparing systems: sample 3

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**

Is model 2 better than model 1?

Comparing systems

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
average:	**82**	**85**

What's the difference?

Comparing systems

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**
std dev	**2.3**	**1.7**

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
average:	**82**	**85**
std dev	**5.9**	**3.9**

Even though the averages are same, the variance is different!

Comparing systems: sample 4

split	model 1	model 2
1	80	82
2	84	87
3	89	90
4	78	82
5	90	91
6	81	83
7	80	80
8	88	89
9	76	77
10	86	88
average:	**83**	**85**
std dev	**4.9**	**4.7**

Is model 2 better than model 1?

Comparing systems: sample 4

split	model 1	model 2	model 2 – model 1
1	80	82	2
2	84	87	3
3	89	90	1
4	78	82	4
5	90	91	1
6	81	83	2
7	80	80	0
8	88	89	1
9	76	77	1
10	86	88	2
average:	**83**	**85**	
std dev	**4.9**	**4.7**	

Is model 2 better than model 1?

Comparing systems: sample 4

split	model 1	model 2	model 2 – model 1
1	80	82	2
2	84	87	3
3	89	90	1
4	78	82	4
5	90	91	1
6	81	83	2
7	80	80	0
8	88	89	1
9	76	77	1
10	86	88	2
average:	**83**	**85**	
std dev	**4.9**	**4.7**	

Model 2 is ALWAYS better

Comparing systems: sample 4

split	model 1	model 2	model 2 – model 1
1	80	82	2
2	84	87	3
3	89	90	1
4	78	82	4
5	90	91	1
6	81	83	2
7	80	80	0
8	88	89	1
9	76	77	1
10	86	88	2
average:	**83**	**85**	
std dev	**4.9**	**4.7**	

How do we decide if model 2 is better than model 1?

Statistical tests

Setup:

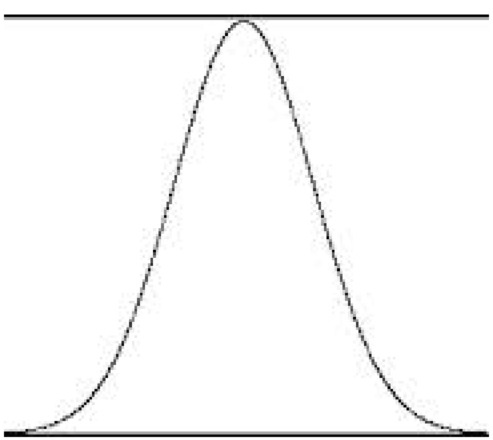
- Assume some default hypothesis about the data that you'd like to *disprove*, called the null hypothesis
- e.g. model 1 and model 2 are not statistically different in performance

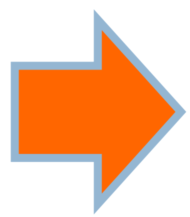
Test:

- Calculate a test statistic from the data (often assuming something about the data)
- Based on this statistic, with *some probability* we can **reject the null hypothesis**, that is, show that it does not hold

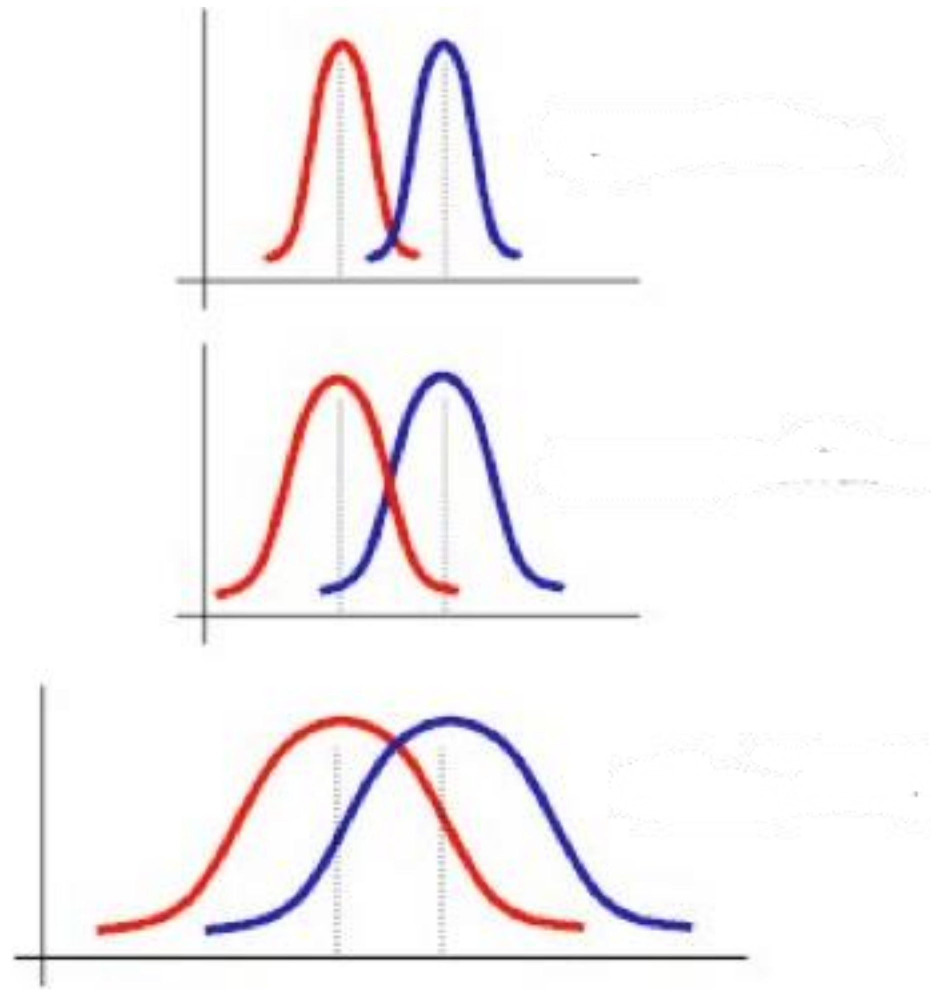
t-test

Determines whether two samples come from the same underlying distribution or not





?



t-test

Null hypothesis: model 1 and model 2 accuracies are no different, i.e. come from **the same** distribution

Assumptions: there are a number that often aren't completely true, but we're often not too far off

Result: probability that the difference in accuracies is due to random chance (low values are better)

Calculating t-test

For our setup, we’ll do what’s called a “pair t-test”

- The values can be thought of as pairs, where they were calculated under the same conditions
- In our case, the same train/test split
- Gives more power than the unpaired t-test (we have more information)

For almost all experiments, we’ll do a “two-tailed” version of the t-test

Can calculate by hand or in code, but why reinvent the wheel: use excel or a statistical package

http://en.wikipedia.org/wiki/Student's_t-test

http://www.statskingdom.com/160MeanT2pair.html

https://www.socscistatistics.com/tests/ttestdependent/Default2.aspx

p-value

The result of a statistical test is often a p-value

p-value: the probability that the null hypothesis holds. Specifically, if we re-ran this experiment multiple times (say on different data) what is the probability that we would reject the null hypothesis incorrectly (i.e. the probability we'd be wrong)

Common values to consider "significant": 0.05 (95% confident), 0.01 (99% confident) and 0.001 (99.9% confident)

Comparing systems: sample 1

split	model 1	model 2
1	87	88
2	85	84
3	83	84
4	80	79
5	88	89
6	85	85
7	83	81
8	87	86
9	88	89
10	84	85
average:	**85**	**85**

Is model 2 better than model 1?

They are the same with: $p = 1$

Comparing systems: sample 2

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
average:	**82**	**85**

Is model 2 better than model 1?

They are the same with: $p = 0.15$

Comparing systems: sample 3

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**

Is model 2 better than model 1?

They are the same with: $p = 0.007$

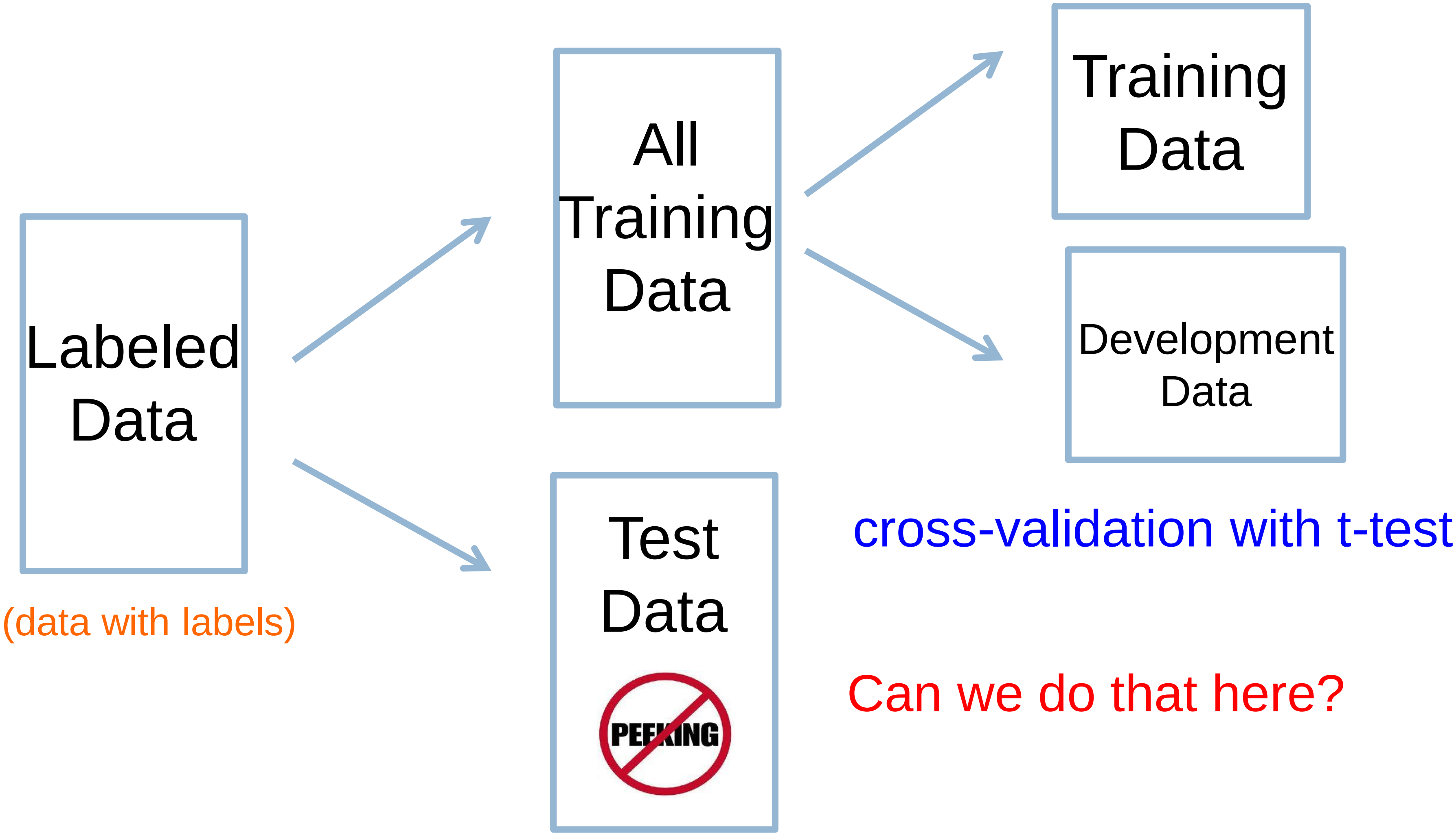
Comparing systems: sample 4

split	model 1	model 2
1	80	82
2	84	87
3	89	90
4	78	82
5	90	91
6	81	83
7	80	80
8	88	89
9	76	77
10	86	88
average:	**83**	**85**

Is model 2 better than model 1?

They are the same with: $p = 0.001$

Statistical tests on test data



Bootstrap resampling

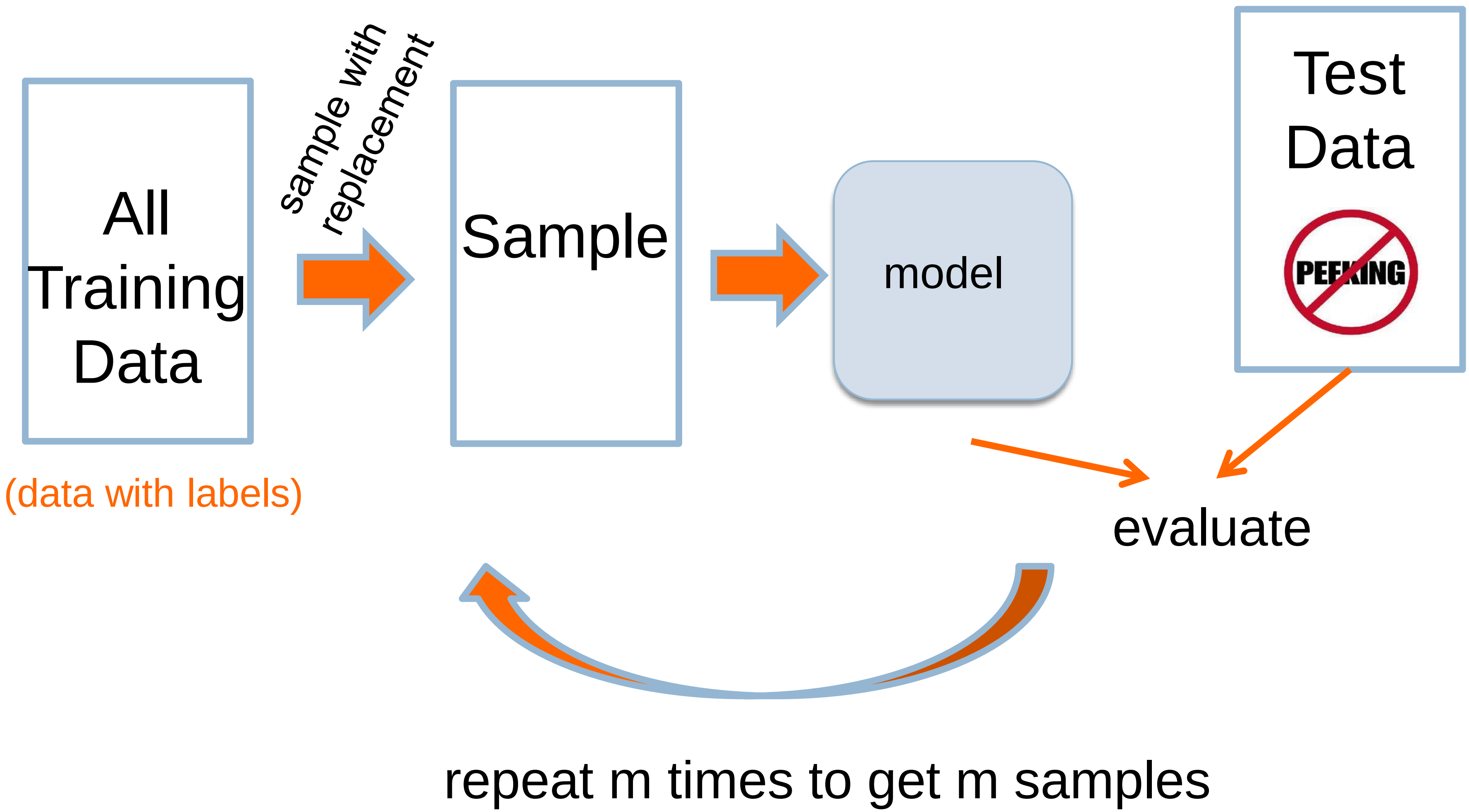
training set *t* with n samples

do *m* times:

- sample *n* examples **with replacement** from the training set to create a new training set *t'*
- train model(s) on *t'*
- calculate performance on test set

calculate t-test (or other statistical test) on the collection of *m* results

Bootstrap resampling



Experimentation good practices

Never look at your test data!

During development

- Compare different models/hyperparameters on development data
- use cross-validation to get more consistent results
- If you want to be confident with results, use a t-test and look for $p = 0.05$

For final evaluation, use bootstrap resampling combined with a t-test to compare final approaches

CSE419 – Artificial Intelligence and Machine Learning 2020

PhD Furkan Gözükara, Toros University

https://github.com/FurkanGozukara/CSE419-Artificial-Intelligence-and-Machine-Learning-2020

Lecture 9 Part 2

Multiclass

Based on Asst. Prof. Dr. David Kauchak (Pomona College) Lecture Slides

Multiclass classification

examples

label

apple

orange

apple

banana

banana

pineapple

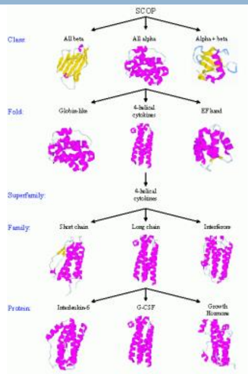
Same setup where we have a set of features for each example

Rather than just two labels, now have 3 or more

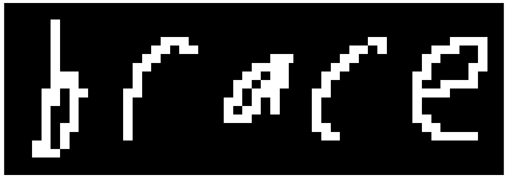
real-world examples?

Real world multiclass classification

document classification



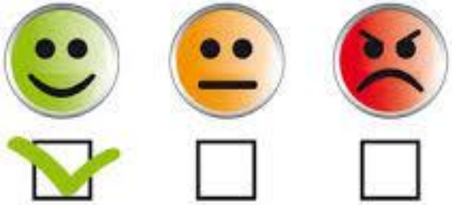
protein classification



handwriting recognition

face recognition

most real-world applications tend to be multiclass

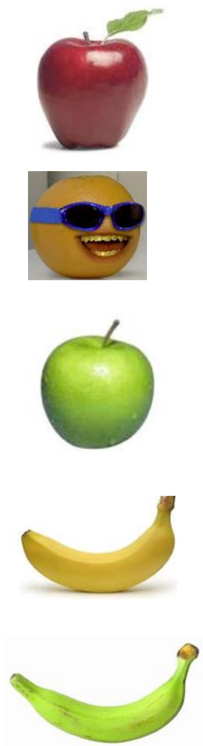


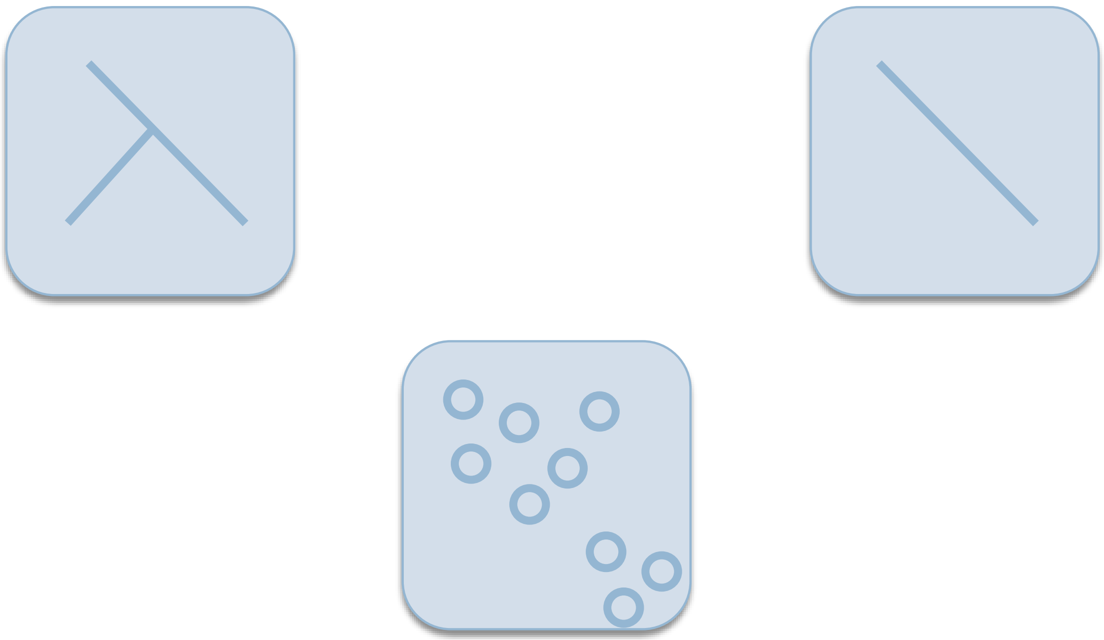
sentiment analysis

autonomous vehicles

emotion recognition

Multiclass: current classifiers





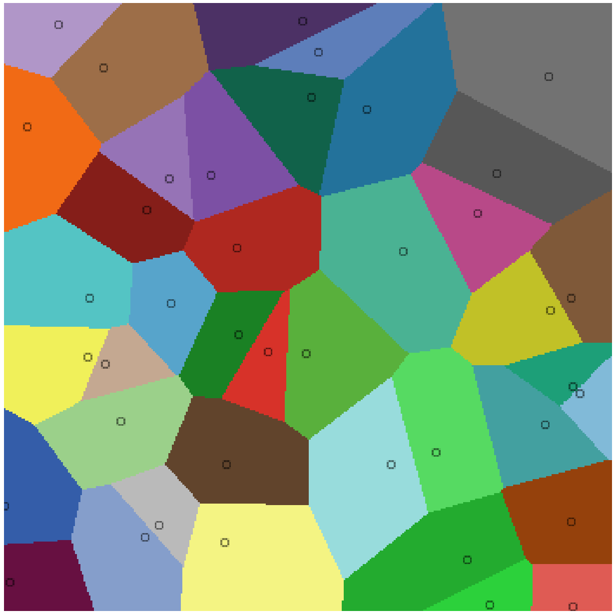
Any of these work out of the box?
With small modifications?

k-Nearest Neighbor (k-NN)

To classify an example ***d***:

- Find ***k*** nearest neighbors of ***d***
- Choose as the label the majority label within the ***k*** nearest neighbors

No algorithmic changes!



Decision Tree learning

Base cases:

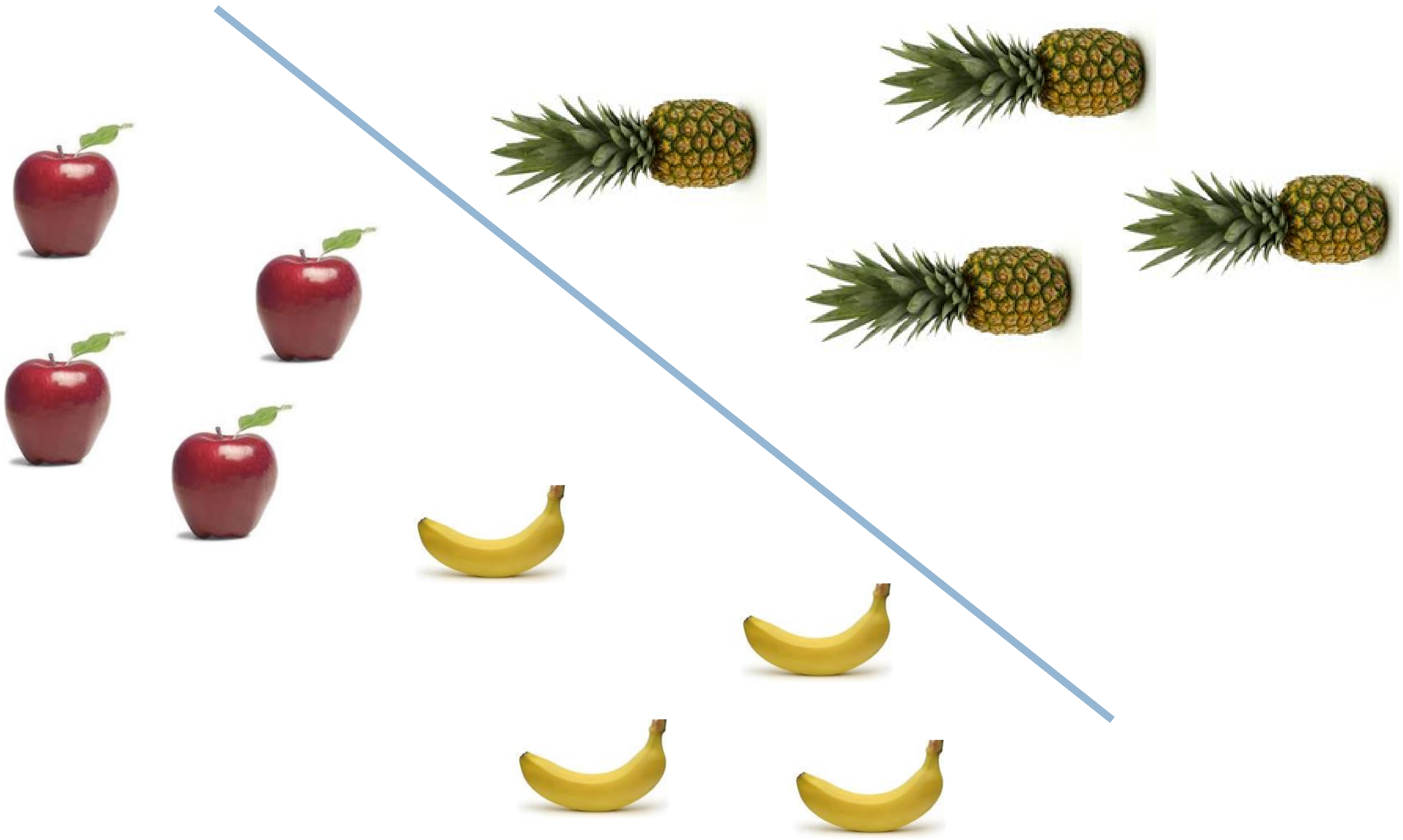
1. If all data belong to the same class, pick that label
2. If all the data have the same feature values, pick majority label
3. If we're out of features to examine, pick majority label
4. If the we don't have any data left, pick majority label of *parent*
5. *If some other stopping criteria* exists to avoid overfitting, pick majority label

Otherwise:

- calculate the "score" for each feature if we used it to split the data
- pick the feature with the highest score, partition the data based on that data value and call recursively

No algorithmic changes!

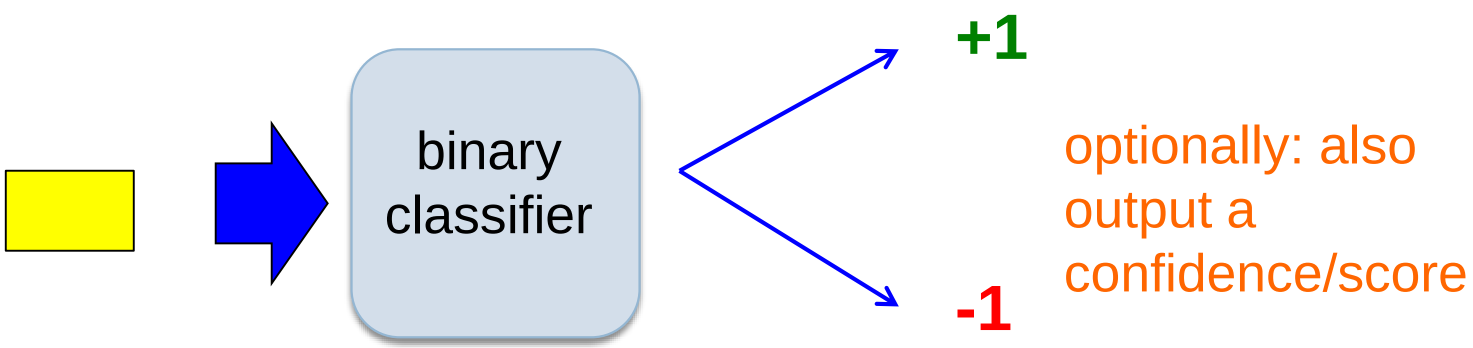
Perceptron learning



Hard to separate three classes with just one line ☹

Black box approach to multiclass

Abstraction: we have a generic binary classifier, how can we use it to solve our new problem



Can we solve our multiclass problem with this?

Multiclass classification

examples

label

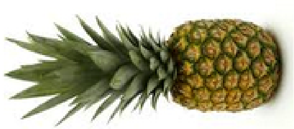

apple


orange


apple


banana


banana

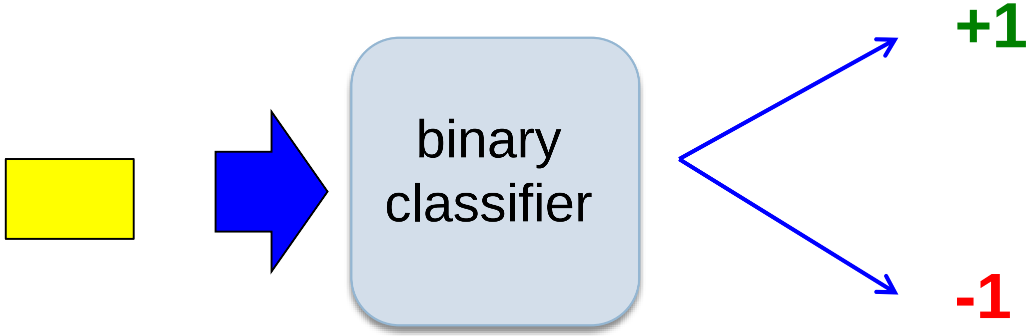

pineapple

Same setup where we have a set of features for each example

Rather than just two labels, now have 3 or more

Black box approach to multiclass

Abstraction: we have a generic binary classifier, how can we use it to solve our new problem



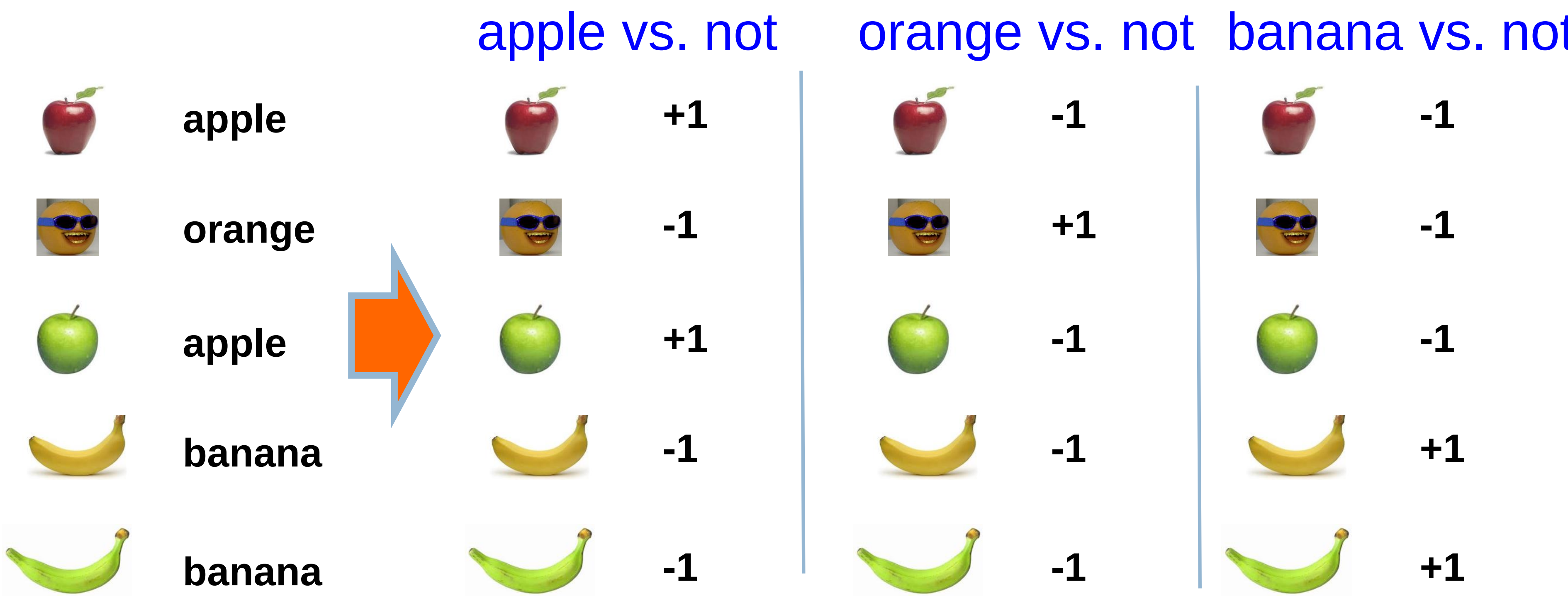
optionally: also output a confidence/score

Can we solve our multiclass problem with this?

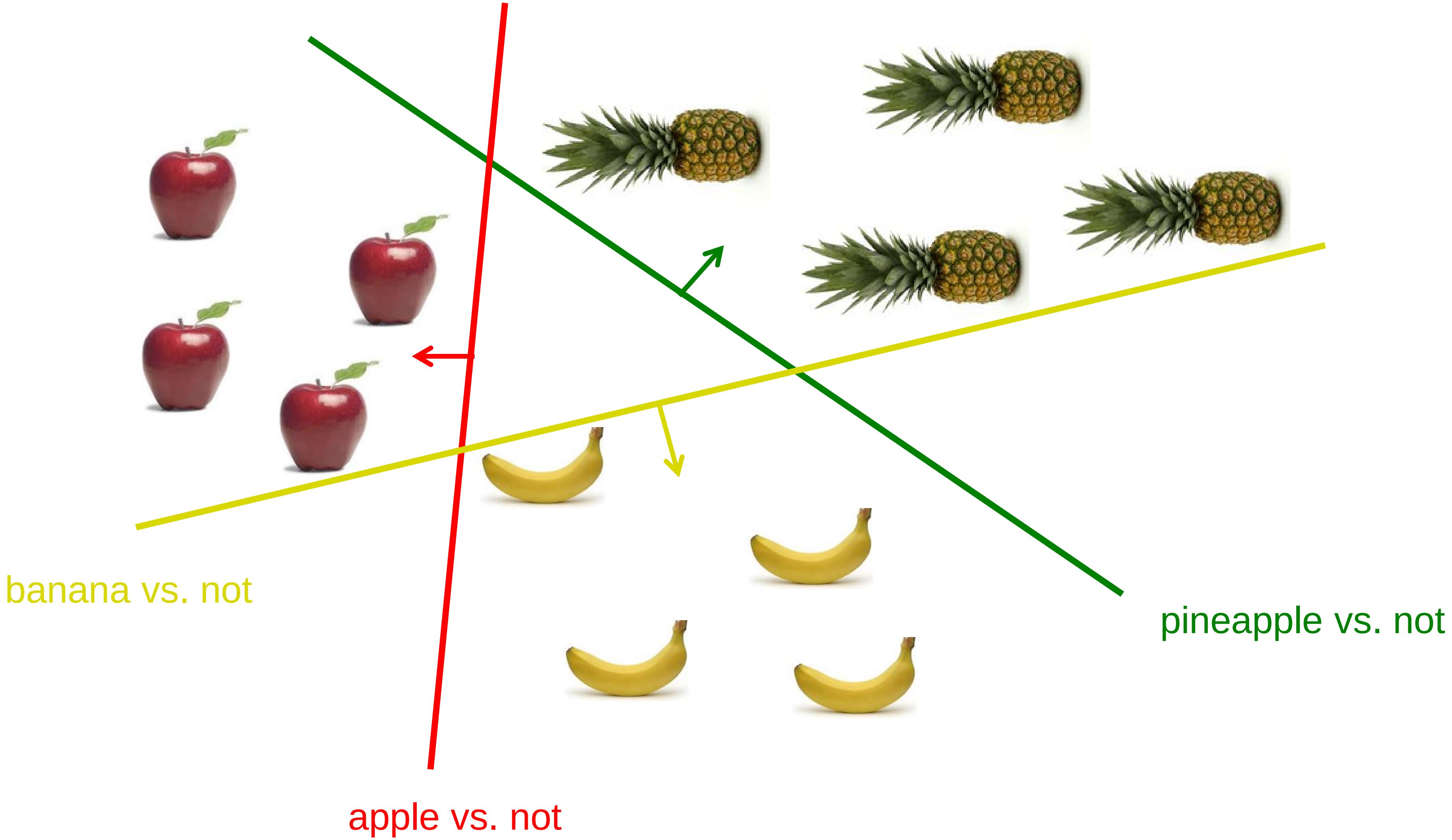
Approach 1: One vs. all (OVA)

Training: for each label *L*, pose as a binary problem

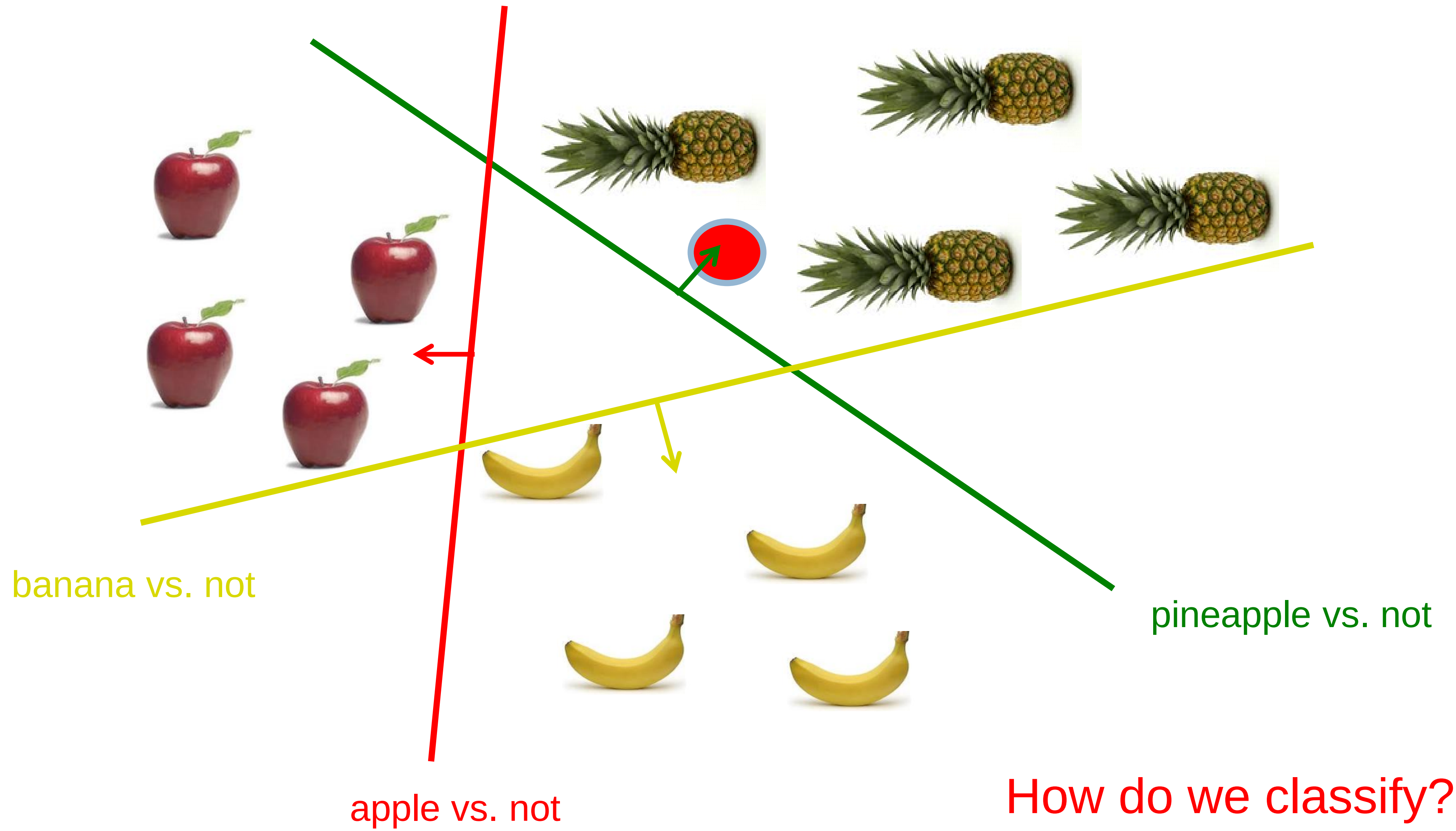
- all examples with label *L* are positive
- all other examples are negative



OVA: linear classifiers (e.g. perceptron)

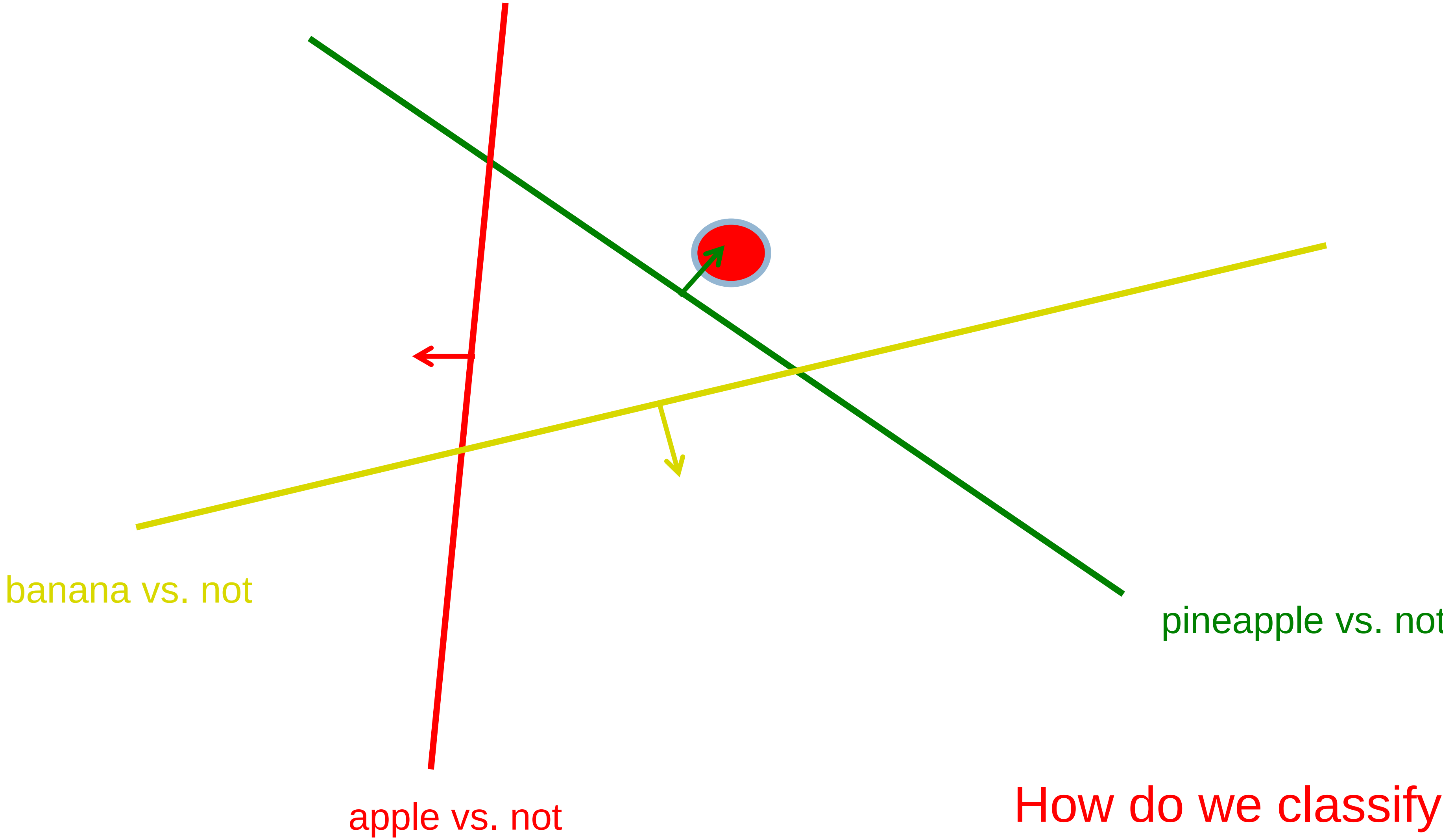


OVA: linear classifiers (e.g. perceptron)



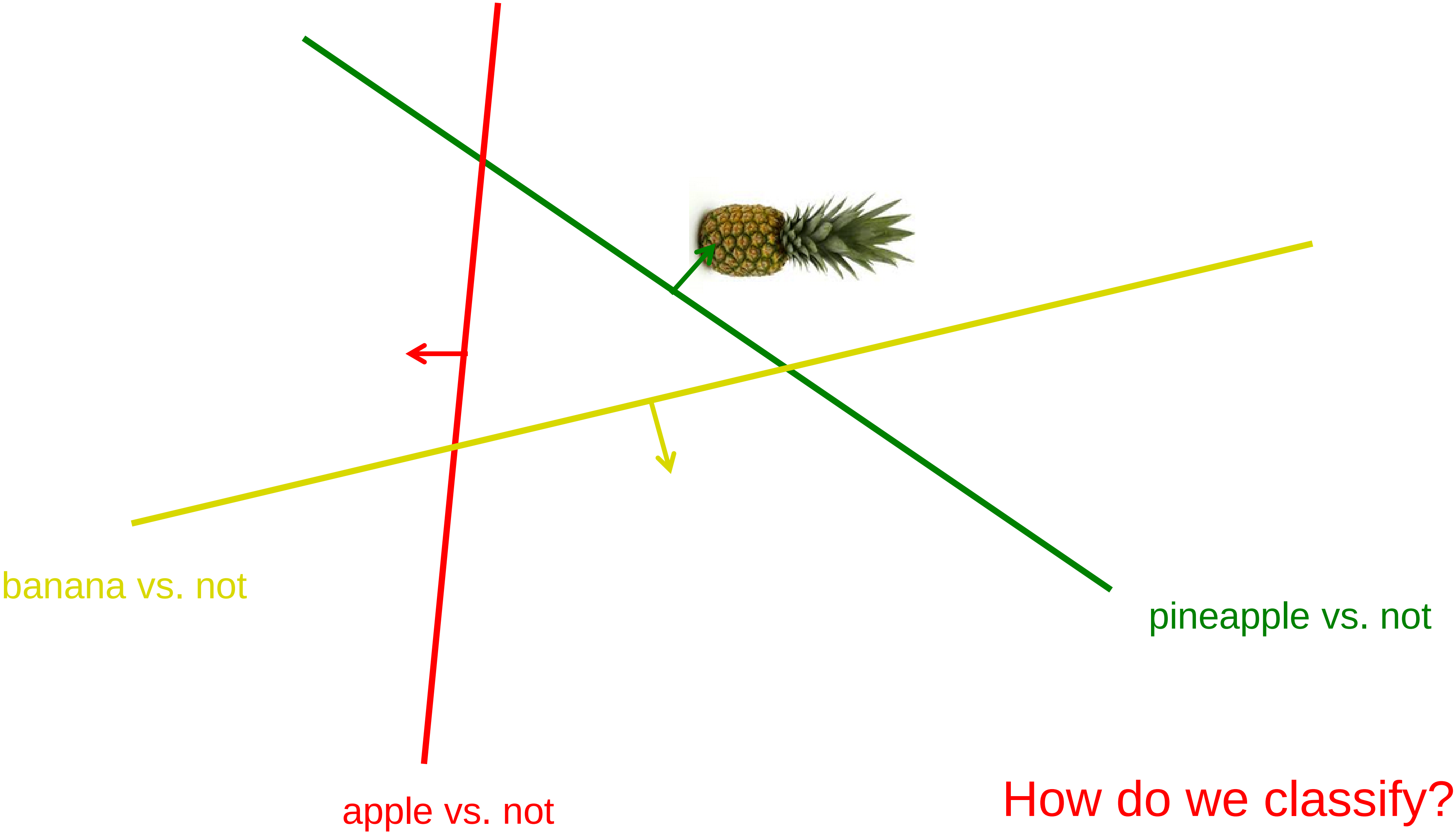
How do we classify?

OVA: linear classifiers (e.g. perceptron)



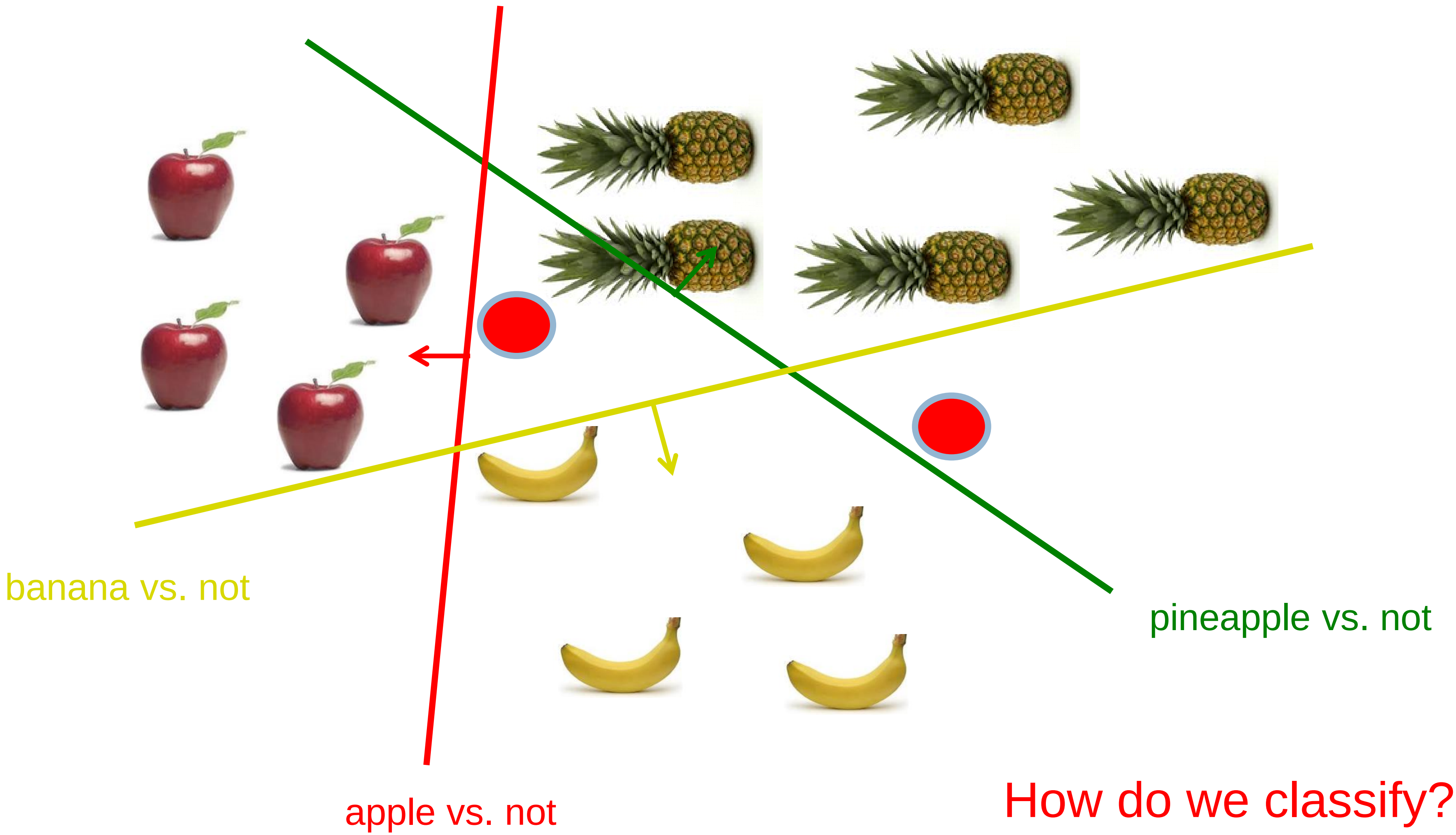
How do we classify?

OVA: linear classifiers (e.g. perceptron)



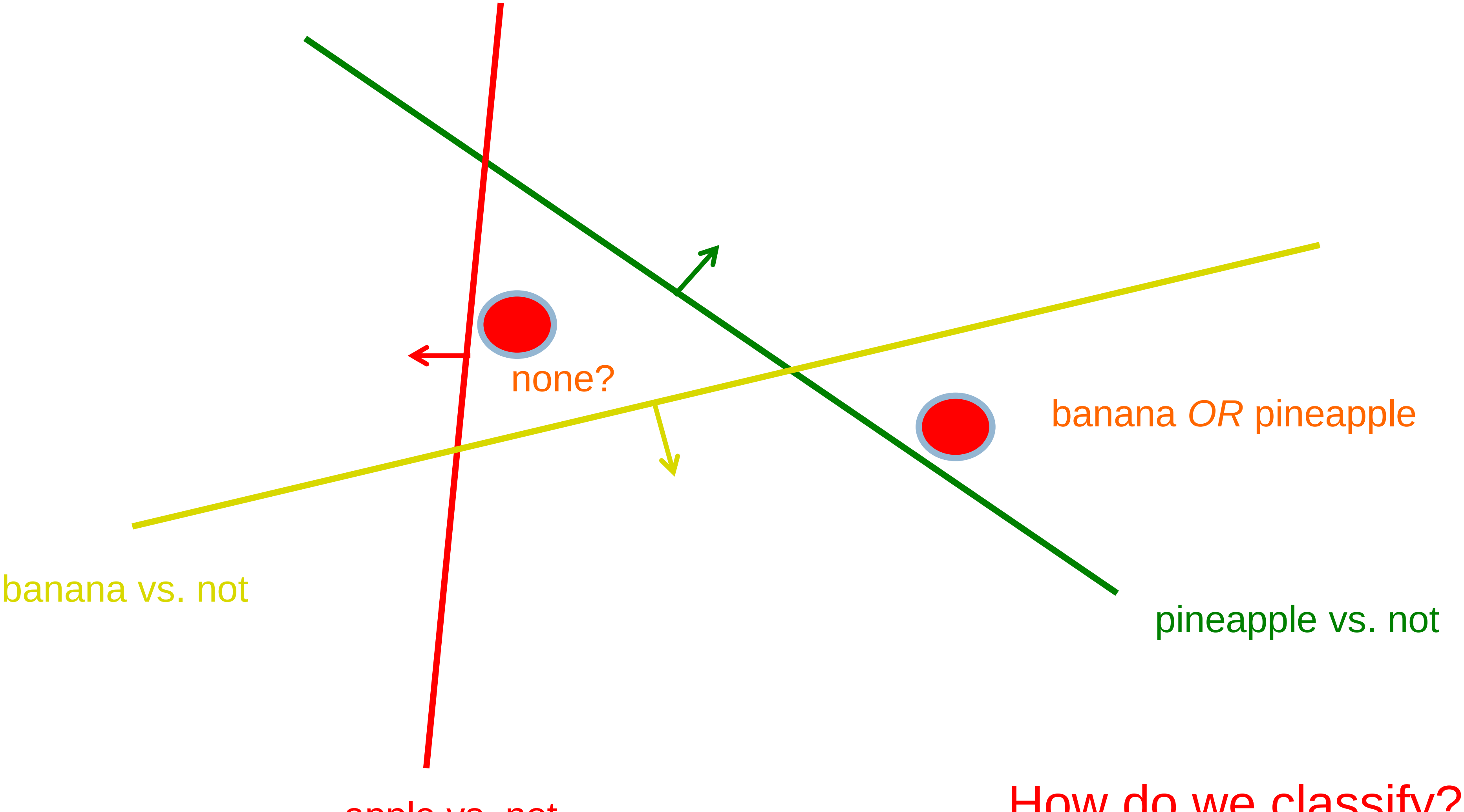
How do we classify?

OVA: linear classifiers (e.g. perceptron)



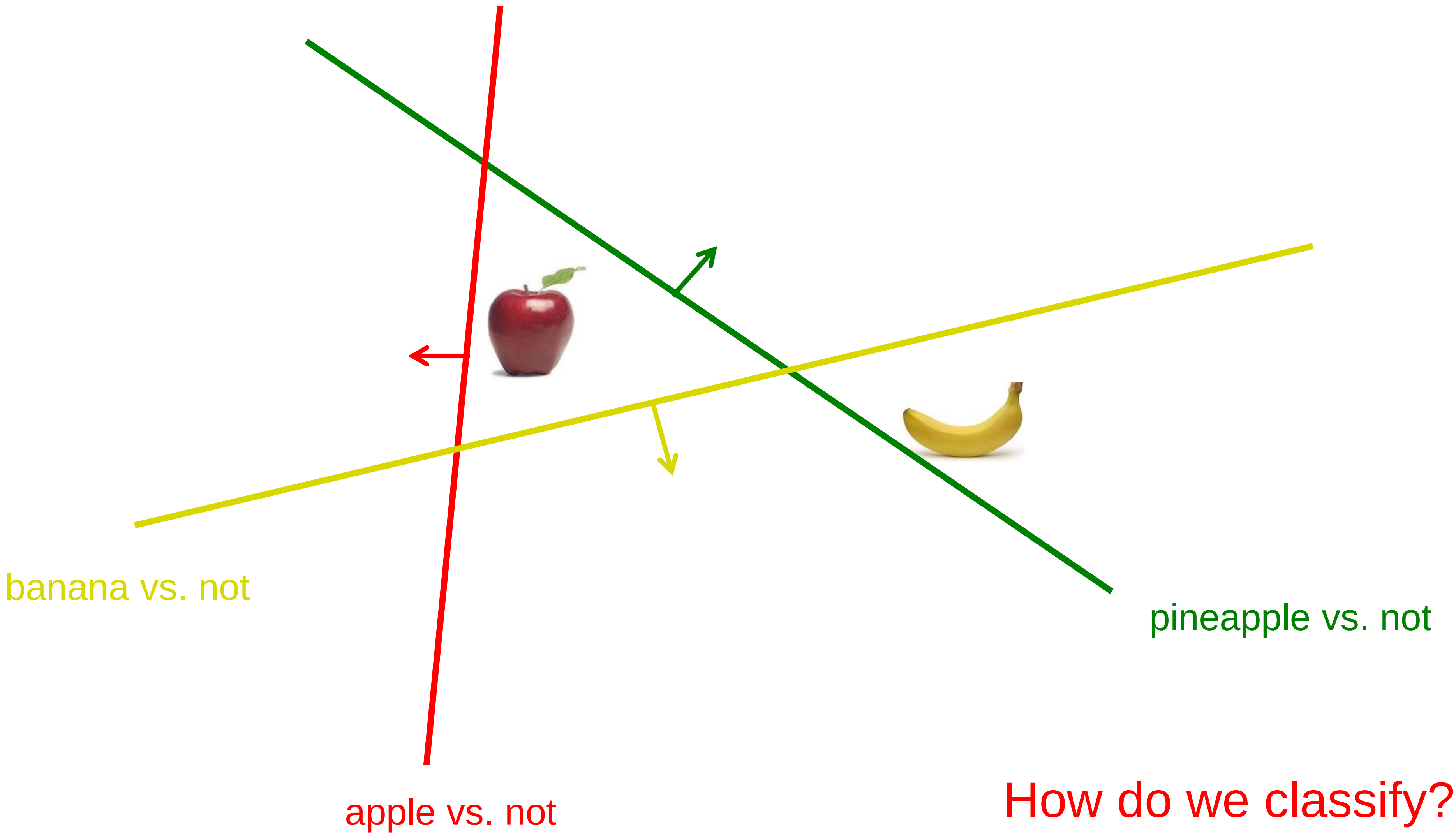
How do we classify?

OVA: linear classifiers (e.g. perceptron)



How do we classify?

OVA: linear classifiers (e.g. perceptron)



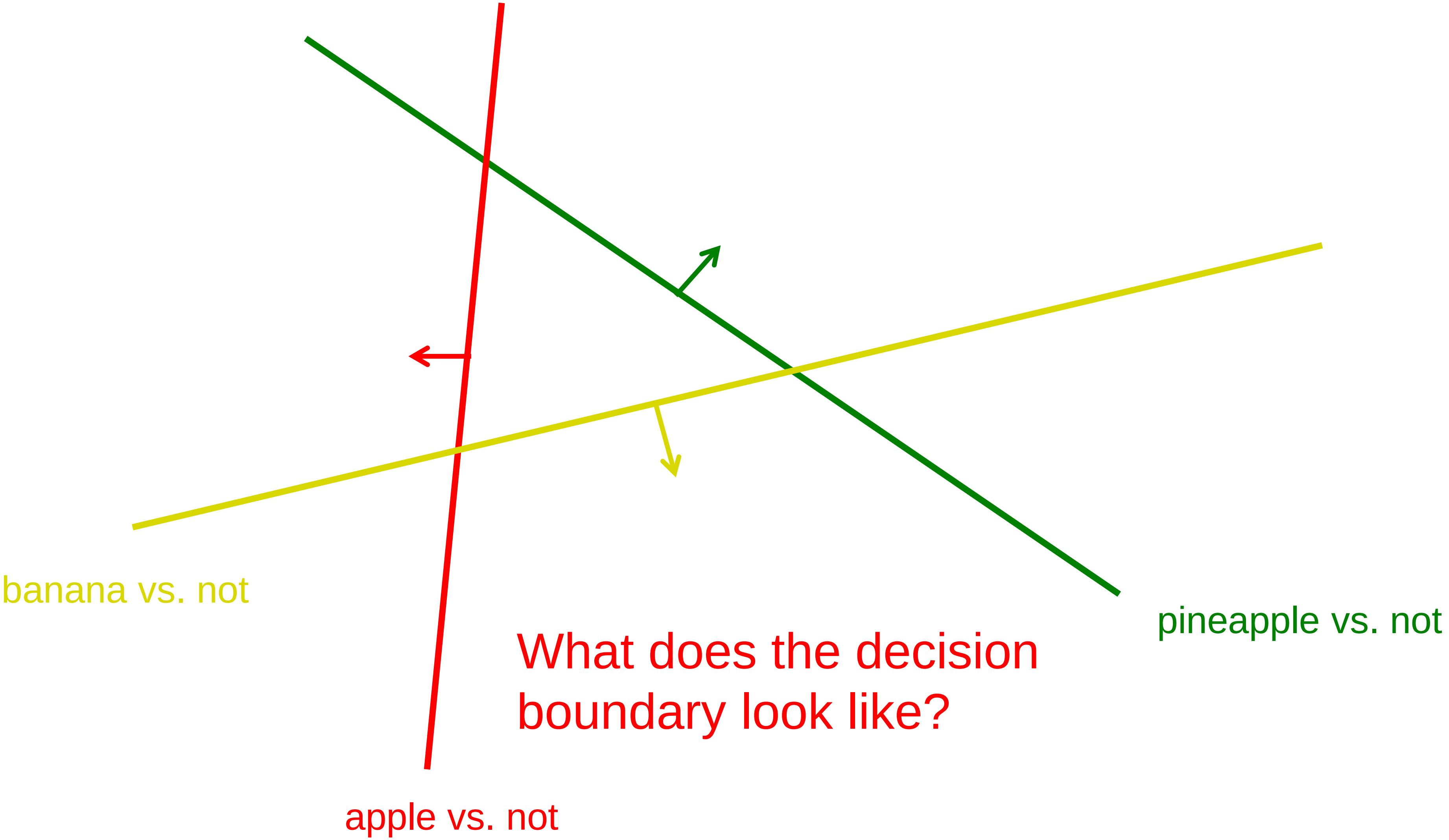
How do we classify?

OVA: classify

Classify:

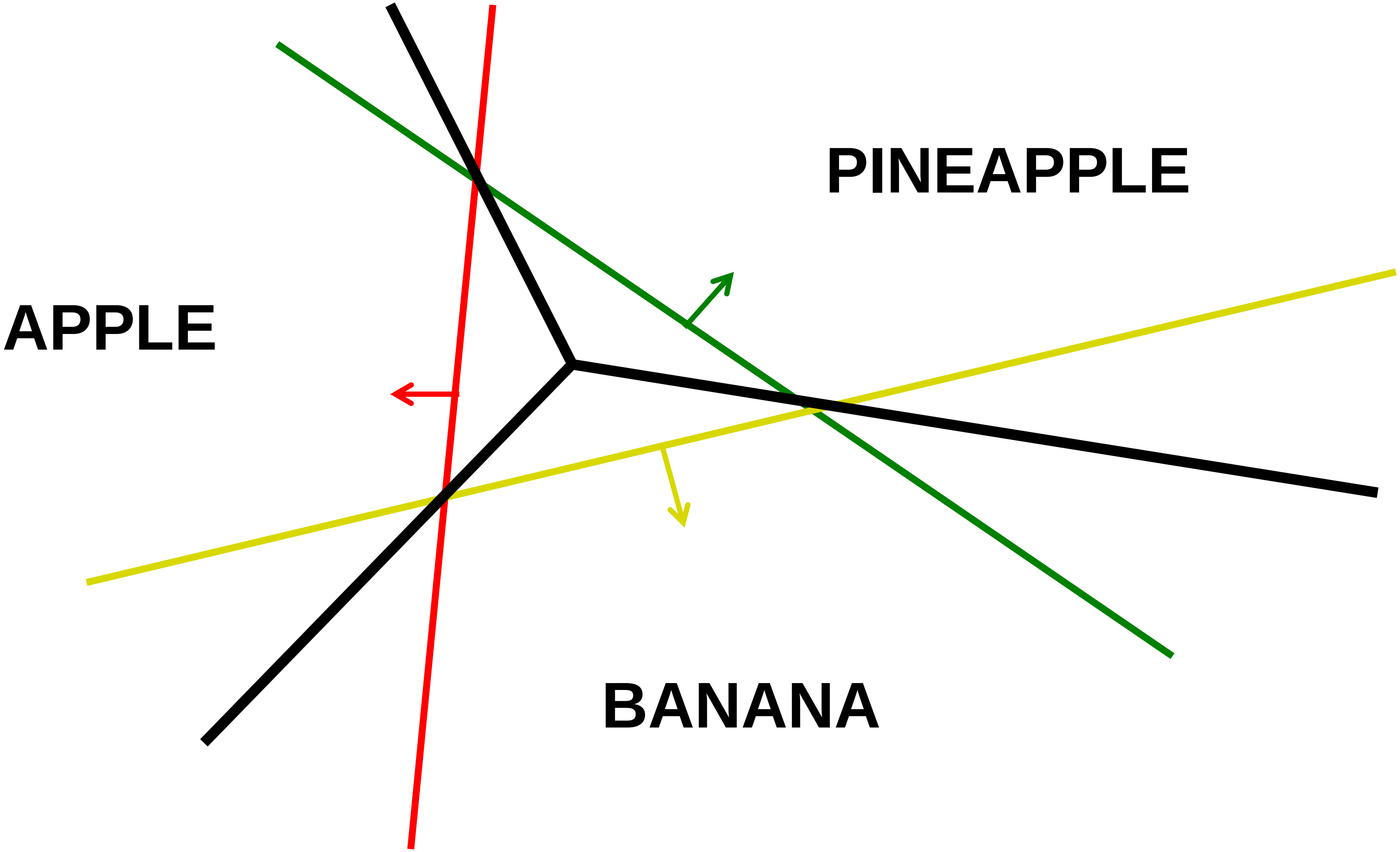
- If classifier doesn't provide confidence (this is rare) and there is ambiguity, pick one of the ones in conflict
- Otherwise:
  - pick the most confident positive
  - if none vote positive, pick *least* confident negative

OVA: linear classifiers (e.g. perceptron)



What does the decision boundary look like?

OVA: linear classifiers (e.g. perceptron)



OVA: classify, perceptron

Classify:

- If classifier doesn't provide confidence (this is rare) and there is ambiguity, pick majority in conflict
- Otherwise:
  - pick the most confident positive
  - if none vote positive, pick *least* confident negative

How do we calculate this for the perceptron?

OVA: classify, perceptron

Classify:

- If classifier doesn't provide confidence (this is rare) and there is ambiguity, pick majority in conflict
- Otherwise:
  - pick the most confident positive
  - if none vote positive, pick *least* confident negative

$$prediction = b + \sum_{i=1}^{n} w_i f_i$$

Distance from the hyperplane

Approach 2: All vs. all (AVA)

Training:

For each pair of labels, train a classifier to distinguish between them

for *i* = 1 to number of labels:

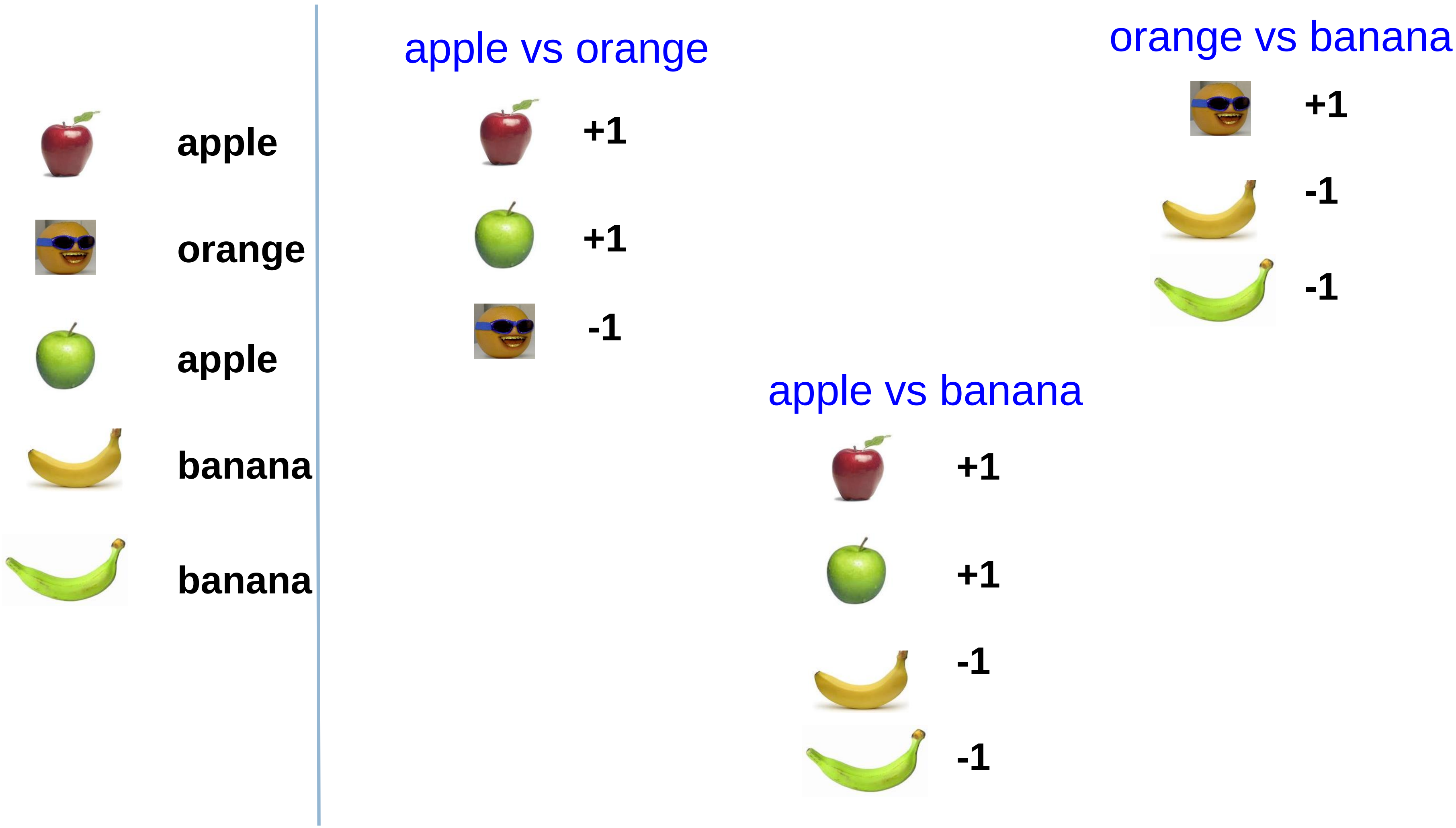
for *k* = i+1 to number of labels:

train a classifier to distinguish between $label_j$ and $label_k$:

- create a dataset with all examples *with* $label_j$ labeled positive and all examples with $label_k$ labeled negative

- train classifier on this subset of the data

AVA training visualized



AVA classify

apple vs orange

 **+1**

 **+1**

 **-1**

apple vs banana

 **+1**

 **+1**

 **-1**

 **-1**

orange vs banana

 **+1**

 **-1**

 **-1**

What class?

AVA classify

apple vs orange


+1


+1


-1

orange

apple vs banana


+1


+1


-1


-1

apple

orange vs banana


+1


-1


-1

orange

orange

In general?

AVA classify

To classify example e, classify with each classifier f_{jk}

We have a few options to choose the final class:

- Take a majority vote
- Take a weighted vote based on confidence
  - $y = f_{jk}(e)$
  - $score_j$ += y   How does this work?
  - $score_k$ -= y

*Here we're assuming that y encompasses both the prediction (+1,-1) and the confidence, i.e. y = prediction * confidence.*

AVA classify

Take a weighted vote based on confidence

- $y = f_{jk}(e)$
- $score_j$ += y
- $score_k$ -= y

If y is positive, classifier thought it was of type j:
- raise the score for j
- lower the score for k

if y is negative, classifier thought it was of type k:
- lower the score for j
- raise the score for k

OVA vs. AVA

Train/classify runtime?

Error? Assume each binary classifier makes an error with probability ε

OVA vs. AVA

Train time:

AVA learns more classifiers, however, they're trained on much smaller data this tends to make it faster if the labels are equally balanced

Test time:

AVA has more classifiers

Error (see the book for more justification):

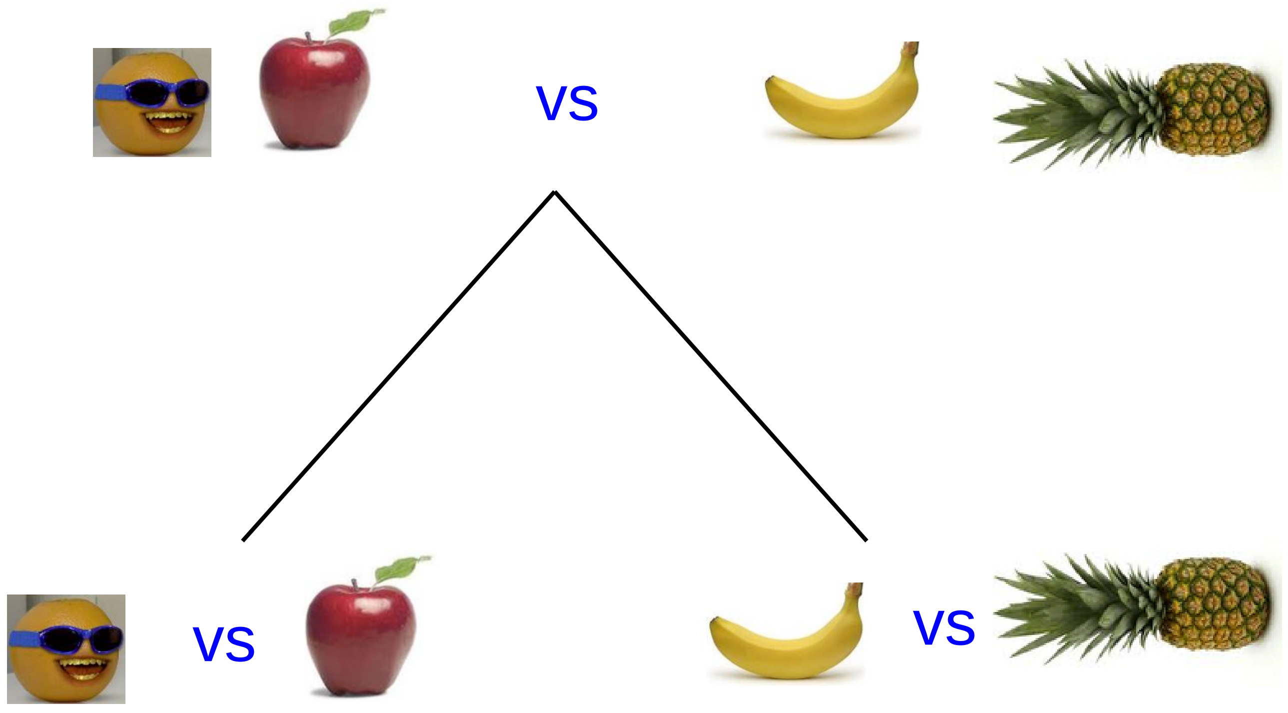
- AVA trains on more balanced data sets
- AVA tests with more classifiers and therefore has more chances for errors

- Theoretically:

-- OVA: ε (number of labels -1)

-- AVA: 2 ε (number of labels -1)

Approach 3: Divide and conquer



Pros/cons vs. AVA?

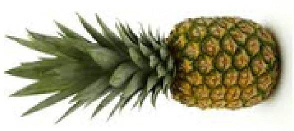
Multiclass summary

If using a binary classifier, the most common thing to do is OVA

Otherwise, use a classifier that allows for multiple labels:

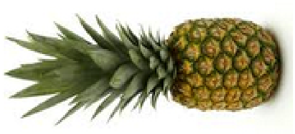
- DT and k-NN work reasonably well
- We'll see a few more in the coming weeks that will often work better

Multiclass evaluation

	label	prediction
	apple	orange
	orange	orange
	apple	apple
	banana	pineapple
	banana	banana
	pineapple	pineapple

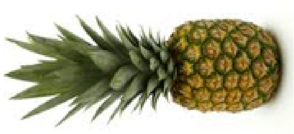
How should we evaluate?

Multiclass evaluation

	label	prediction
	apple	orange
	orange	orange
	apple	apple
	banana	pineapple
	banana	banana
	pineapple	pineapple

Accuracy: 4/6

Multiclass evaluation imbalanced data

	label	prediction
	apple	orange
	…	
	apple	apple
	banana	pineapple
	banana	banana
	pineapple	pineapple

Any problems?

Data imbalance!

Macroaveraging vs. microaveraging

microaveraging: average over examples (this is the “normal” way of calculating)

macroaveraging: calculate evaluation score (e.g. accuracy) for each label, then average over labels

What effect does this have?
Why include it?

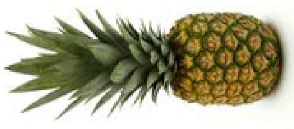
Macroaveraging vs. microaveraging

microaveraging: average over examples (this is the "normal" way of calculating)

macroaveraging: calculate evaluation score (e.g. accuracy) for each label, then average over labels

- Puts more weight/emphasis on rarer labels
- Allows another dimension of analysis

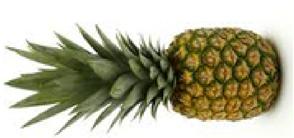
Macroaveraging vs. microaveraging

	label	prediction
	apple	orange
	orange	orange
	apple	apple
	banana	pineapple
	banana	banana
	pineapple	pineapple

microaveraging:
average over examples

macroaveraging:
calculate evaluation score (e.g. accuracy) for each label, then average over labels

Macroaveraging vs. microaveraging

	label	prediction
	apple	orange
	orange	orange
	apple	apple
	banana	pineapple
	banana	banana
	pineapple	pineapple

microaveraging: 4/6

macroaveraging:

apple = 1/2

orange = 1/1

banana = 1/2

pineapple = 1/1

total = (1/2 + 1 + 1/2 + 1)/4

= 3/4

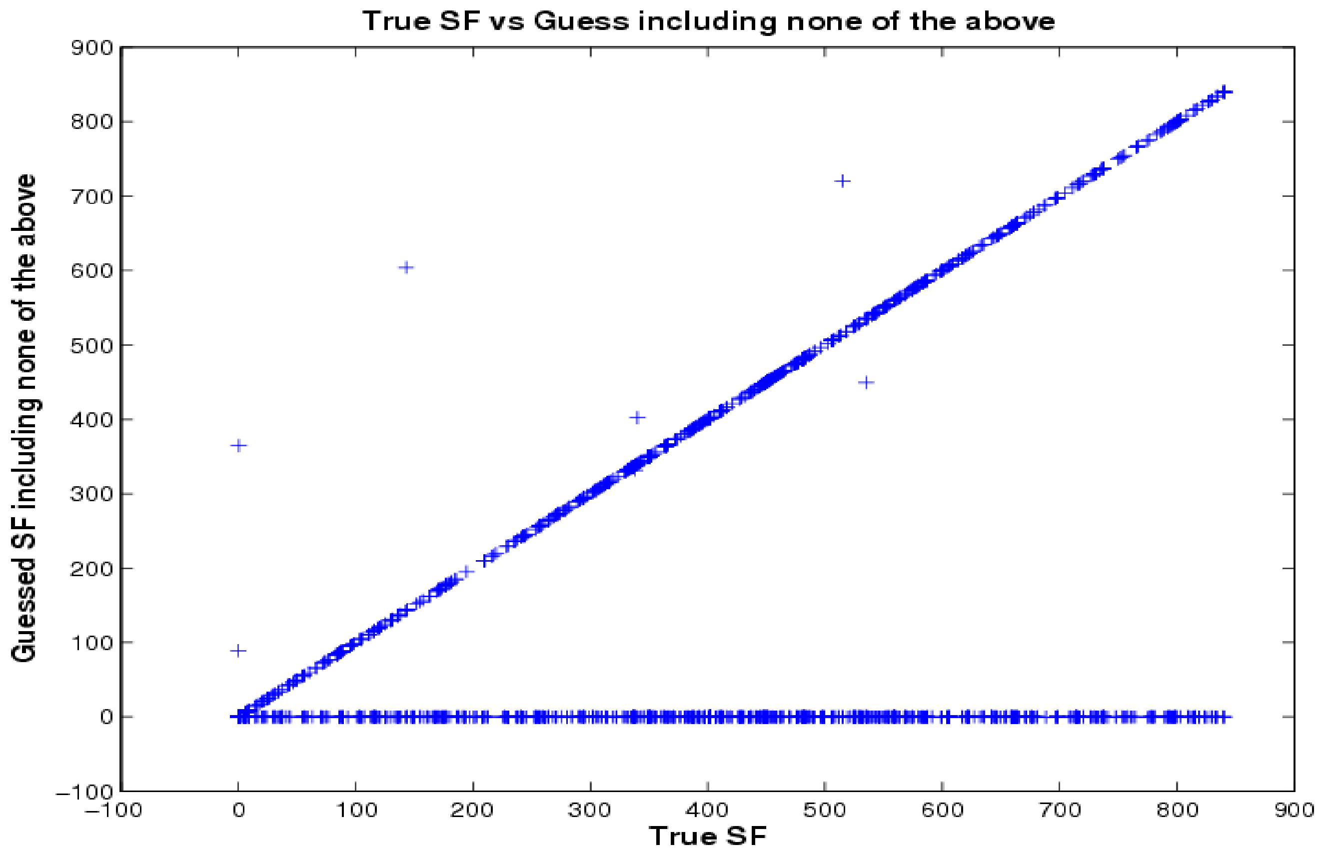
Confusion matrix

entry *(i, j)* represents the number of examples with label *i* that were predicted to have label *j*

another way to understand both the data and the classifier

	Classic	Country	Disco	Hiphop	Jazz	Rock
Classic	86	2	0	4	18	1
Country	1	57	5	1	12	13
Disco	0	6	55	4	0	5
Hiphop	0	15	28	90	4	18
Jazz	7	1	0	0	37	12
Rock	6	19	11	0	27	48

Confusion matrix



BLAST classification of proteins in 850 superfamilies