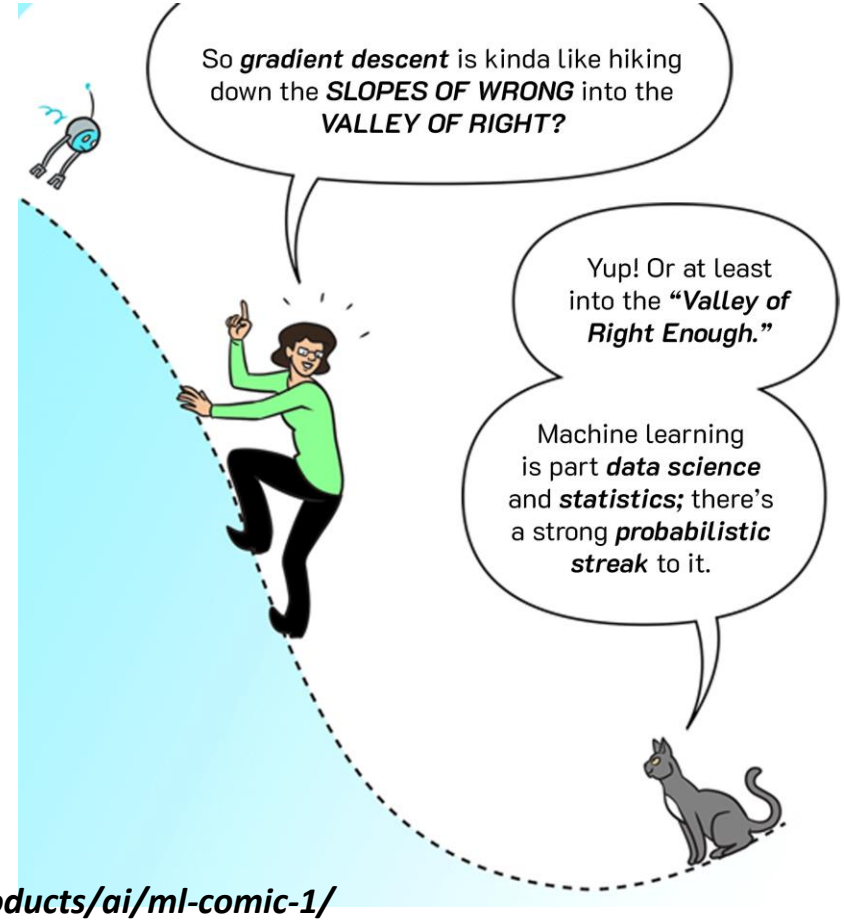
Machine Learning

L3_Geometric view and K-NN



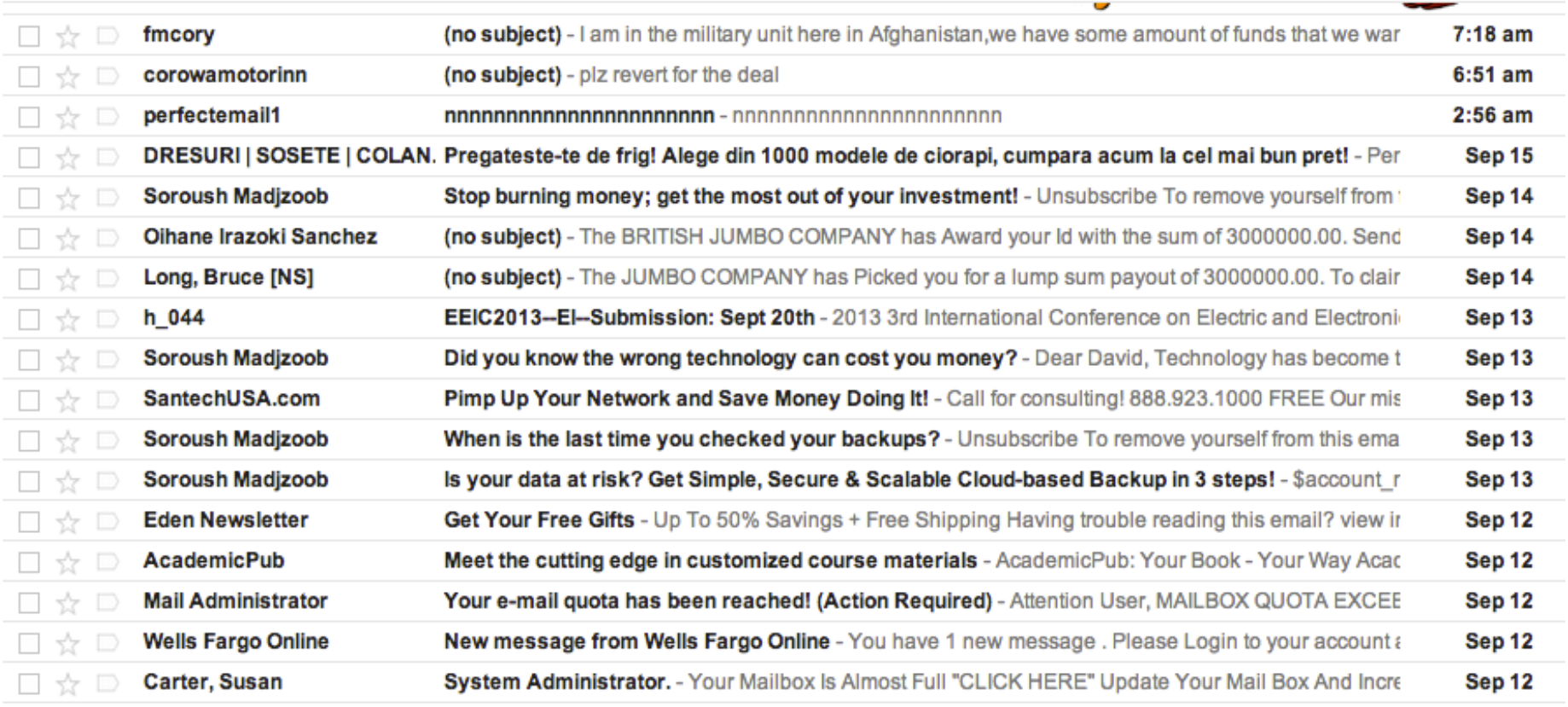
https://cloud.google.com/products/ai/ml-comic-1/

Today's outline

- Notes on Supervised learning evaluation
- Geometric view of data
- K-nearest neighbor
- K-nearest neighbor vs Decision Trees

Real-world classification

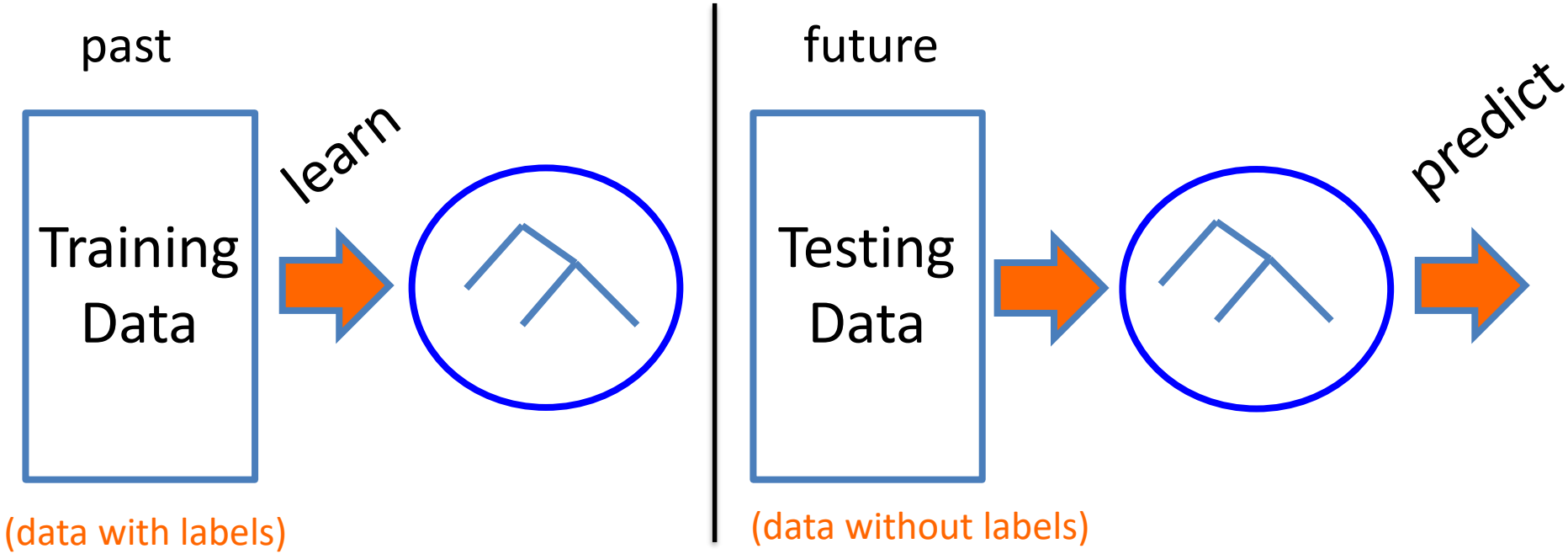
Google has labeled training data, for example from people clicking the "spam" button, but when new messages come in, they're not labeled



fmcory	**(no subject)** - I am in the military unit here in Afghanistan,we have some amount of funds that we war	7:18 am
corowamotorinn	**(no subject)** - plz revert for the deal	6:51 am
perfectemail1	**nnnnnnnnnnnnnnnnnnnn** - nnnnnnnnnnnnnnnnnnnn	2:56 am
DRESURI | SOSETE | COLAN.	**Pregateste-te de frig! Alege din 1000 modele de ciorapi, cumpara acum la cel mai bun pret!** - Per	Sep 15
Soroush Madjzoob	**Stop burning money; get the most out of your investment!** - Unsubscribe To remove yourself from	Sep 14
Oihane Irazoki Sanchez	**(no subject)** - The BRITISH JUMBO COMPANY has Award your Id with the sum of 3000000.00. Send	Sep 14
Long, Bruce [NS]	**(no subject)** - The JUMBO COMPANY has Picked you for a lump sum payout of 3000000.00. To clair	Sep 14
h_044	**EEIC2013--EI--Submission: Sept 20th** - 2013 3rd International Conference on Electric and Electroni	Sep 13
Soroush Madjzoob	**Did you know the wrong technology can cost you money?** - Dear David, Technology has become t	Sep 13
SantechUSA.com	**Pimp Up Your Network and Save Money Doing It!** - Call for consulting! 888.923.1000 FREE Our mis	Sep 13
Soroush Madjzoob	**When is the last time you checked your backups?** - Unsubscribe To remove yourself from this ema	Sep 13
Soroush Madjzoob	**Is your data at risk? Get Simple, Secure & Scalable Cloud-based Backup in 3 steps!** - $account_r	Sep 13
Eden Newsletter	**Get Your Free Gifts** - Up To 50% Savings + Free Shipping Having trouble reading this email? view ir	Sep 12
AcademicPub	**Meet the cutting edge in customized course materials** - AcademicPub: Your Book - Your Way Acac	Sep 12
Mail Administrator	**Your e-mail quota has been reached! (Action Required)** - Attention User, MAILBOX QUOTA EXCEE	Sep 12
Wells Fargo Online	**New message from Wells Fargo Online** - You have 1 new message . Please Login to your account a	Sep 12
Carter, Susan	**System Administrator.** - Your Mailbox Is Almost Full "CLICK HERE" Update Your Mail Box And Incre	Sep 12

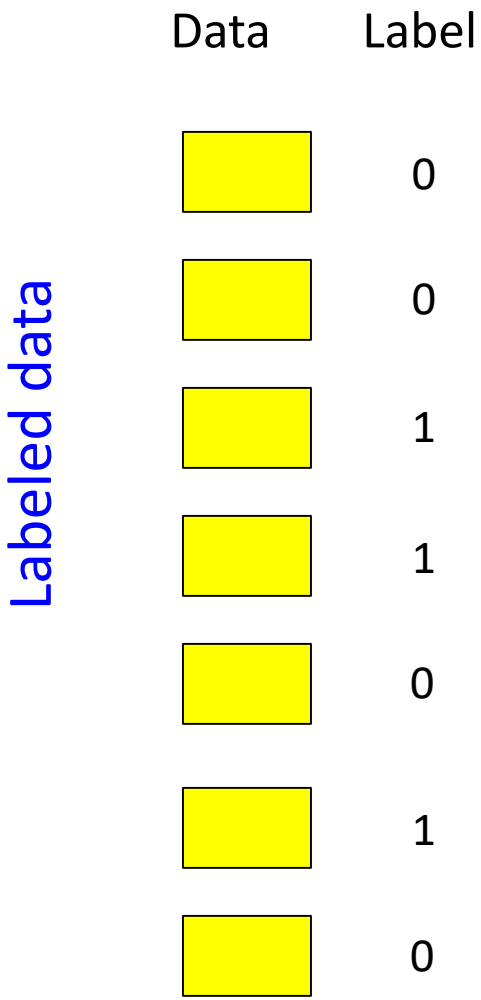
Experimental setup

REAL WORLD USE OF ML ALGORITHMS



How do we tell how well we're doing?

Classification evaluation

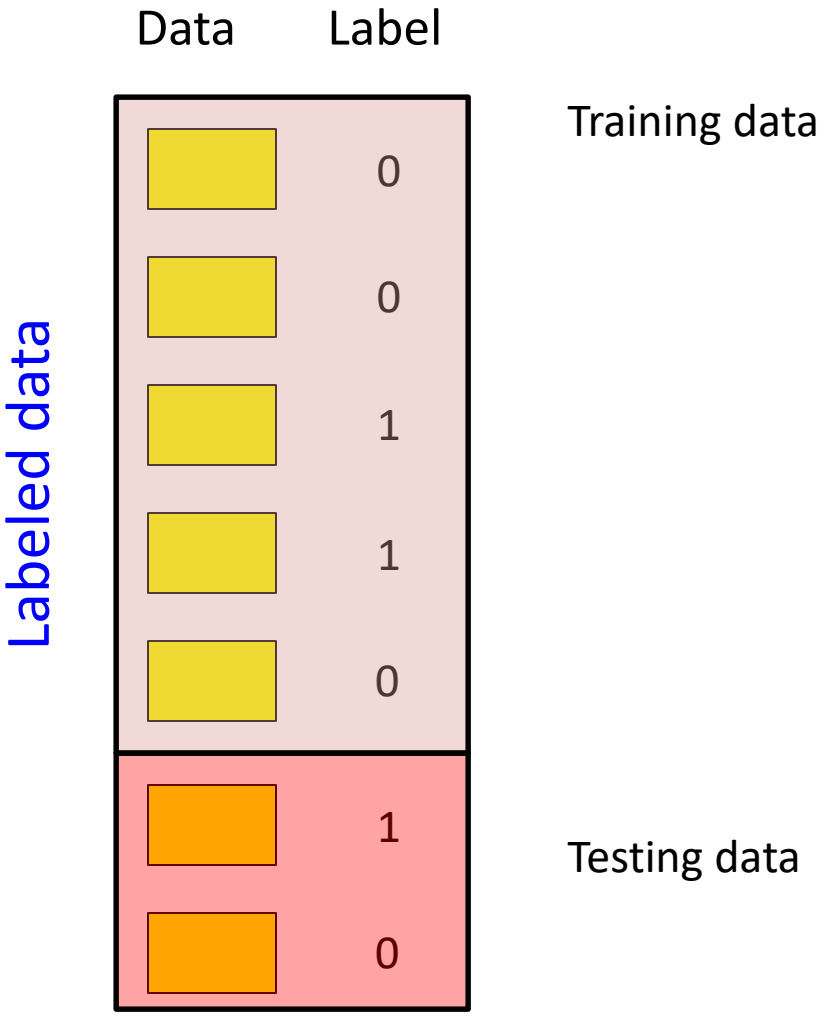


Use the labeled data we have already to create a test set with known labels!

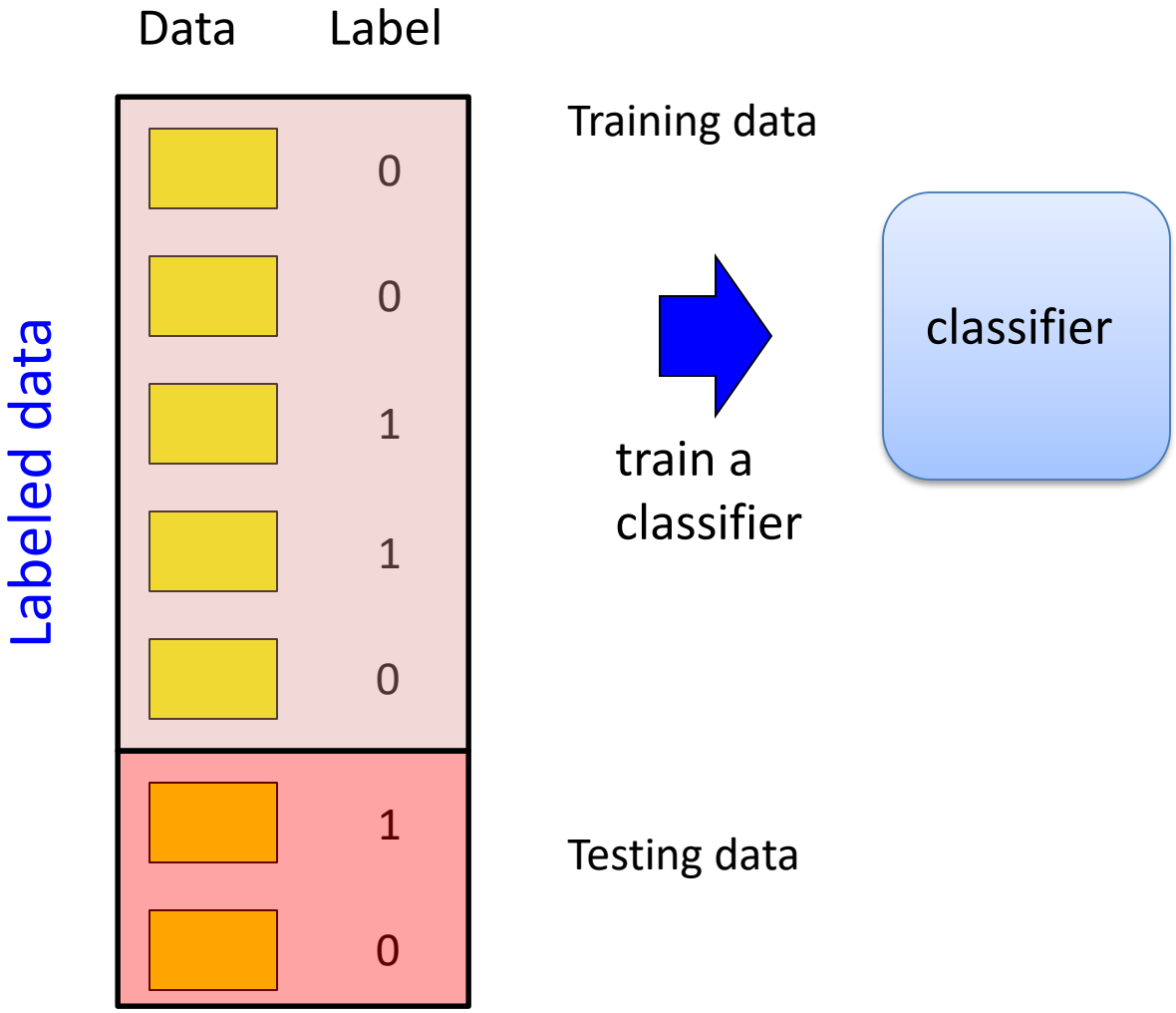
Why can we do this?

Remember, we assume there's an underlying distribution that generates both the training and test examples

Classification evaluation

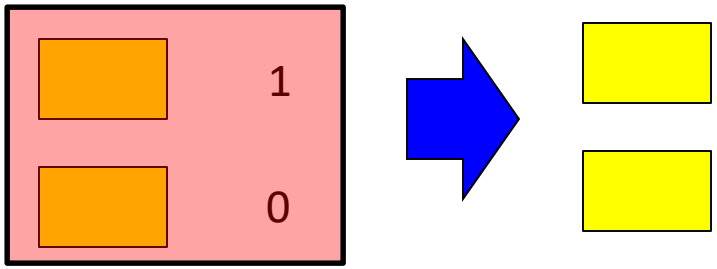


Classification evaluation



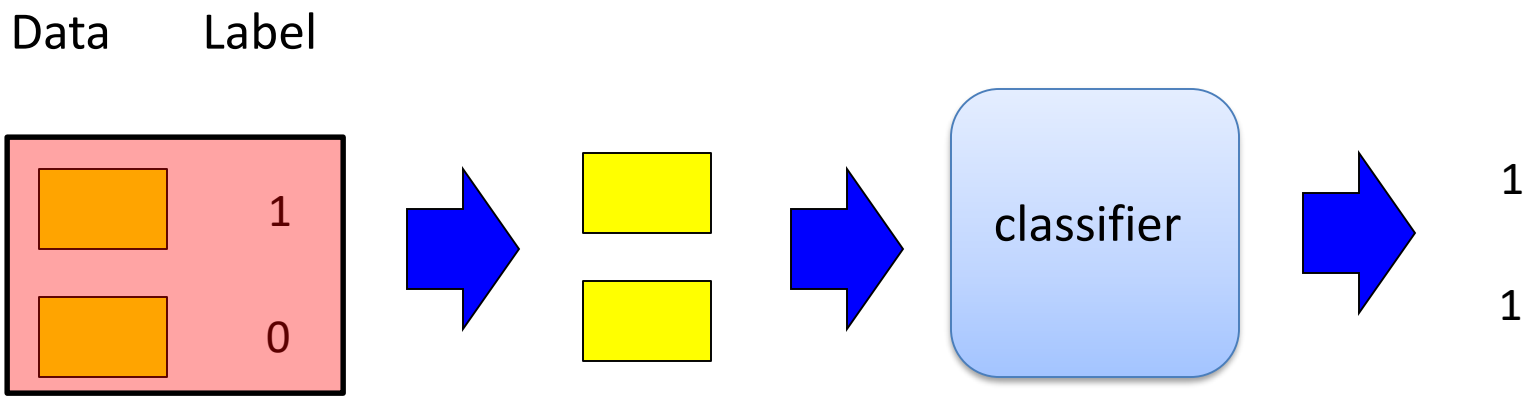
Classification evaluation

Data Label

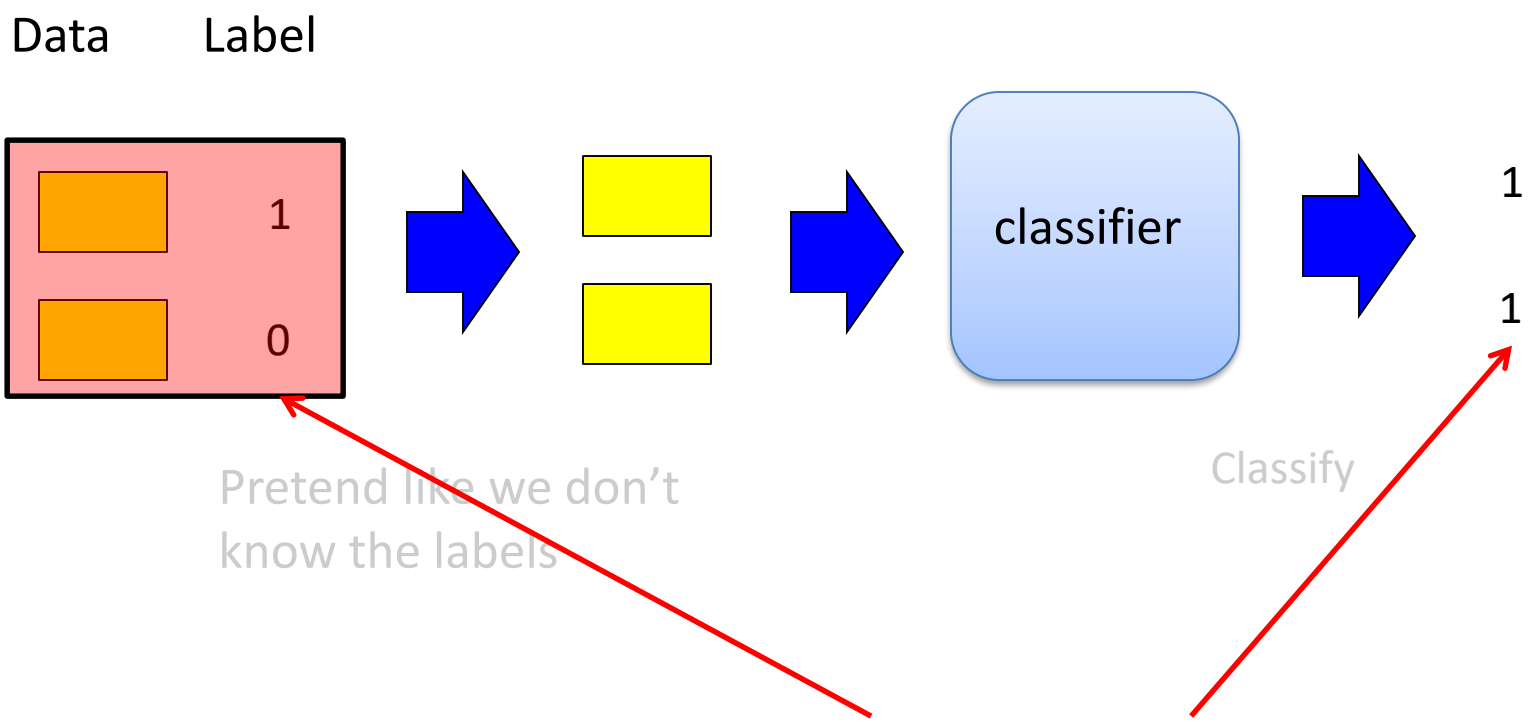


Pretend like we don't know the labels

Classification evaluation



Classification evaluation

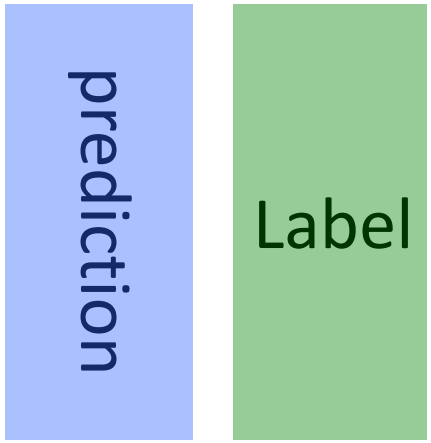


How could we score these for classification?

Compare predicted labels to actual labels

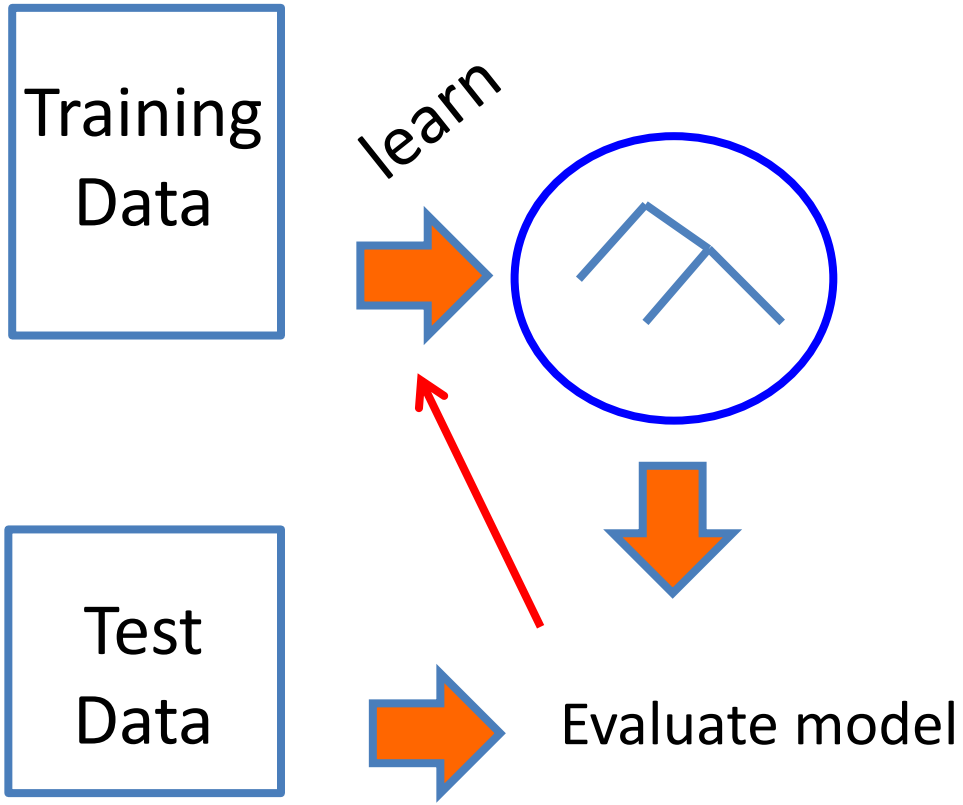
Test accuracy

To evaluate the model, compare the predicted labels to the actual labels



Accuracy: the proportion of examples where we correctly predicted the label

Proper testing



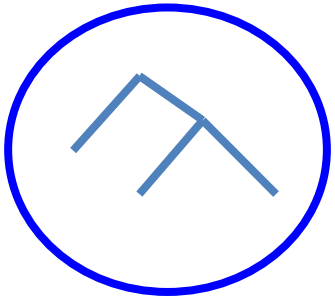
One way to do algorithm development:

- try out an algorithm
- evaluated on test data
- repeat until happy with results

Is this ok?

No. Although we're not explicitly looking at the examples, we're still "cheating" by biasing our algorithm to the test data

Proper testing



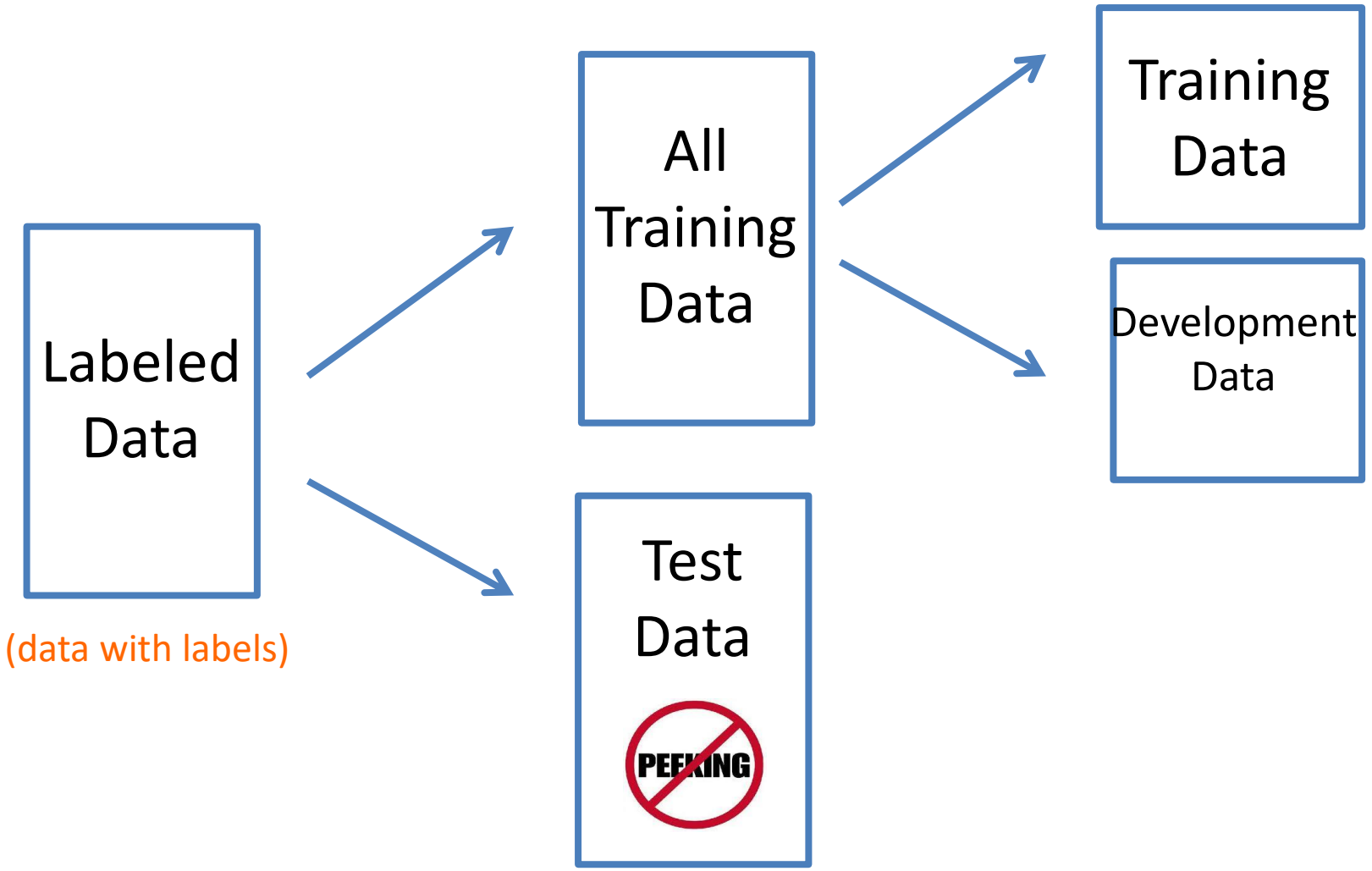
Once you look at/use test data **it is no longer test data!**

Test Data

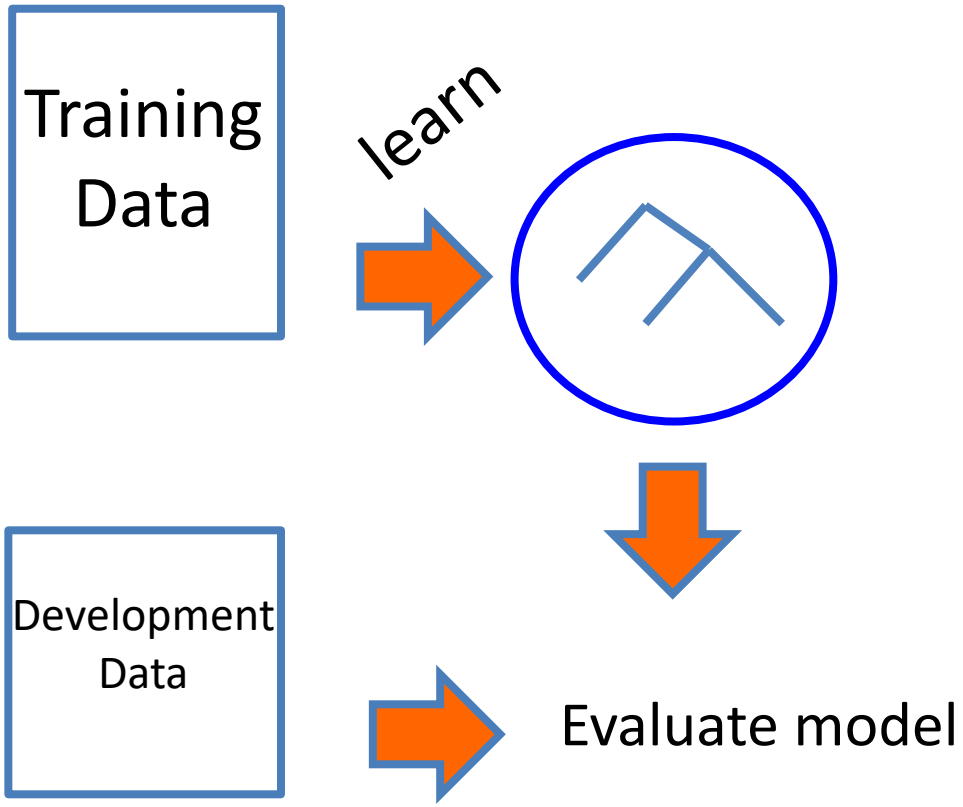
Evaluate model

So, how can we evaluate our algorithm during development?

Development set



Proper testing

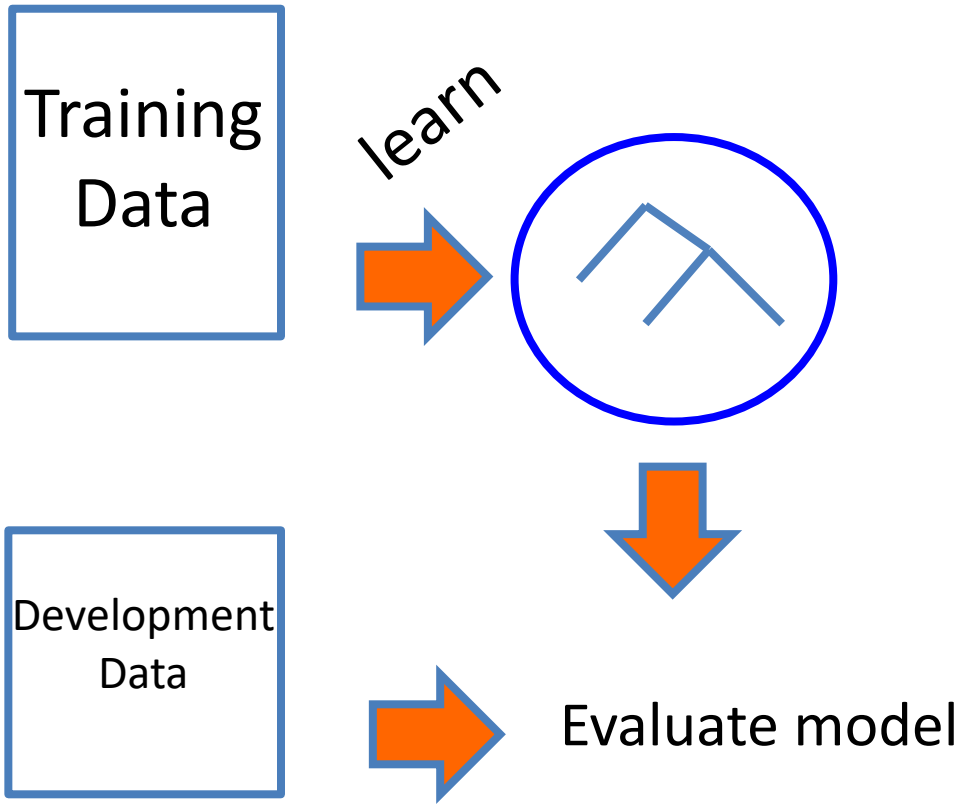


Using the **development data**:

- try out an algorithm
- evaluated on development data
- repeat until happy with results

When satisfied, evaluate on test data

Proper testing



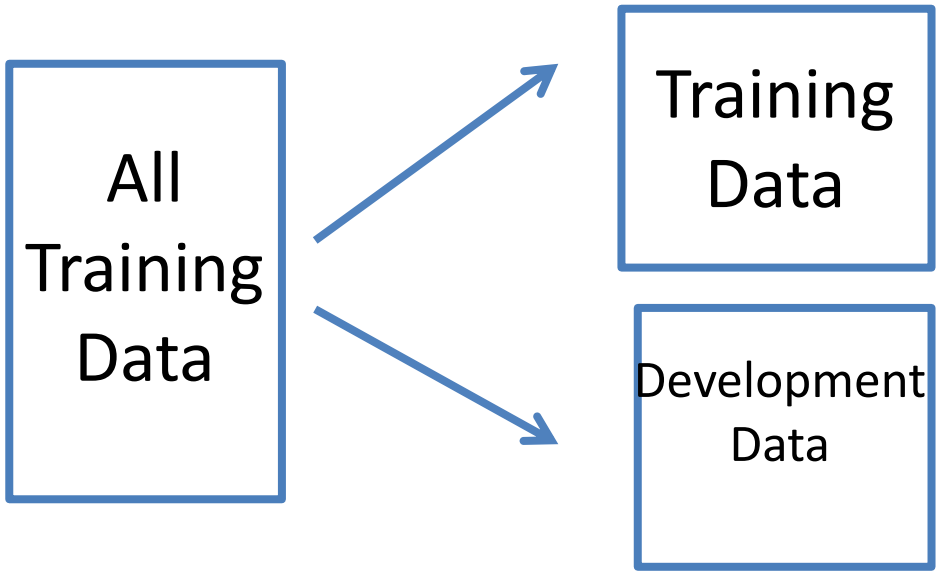
Using the **development data**:

- try out an algorithm
- evaluated on development data
- repeat until happy with results

Any problems with this?

Overfitting to development data

Be careful not to overfit to the development data!



Often we'll split off development data multiple times (e.g. K-fold cross validation)

you can still overfit, but this helps avoid it

Machine Learning: A Geometric View

Apples vs. Bananas

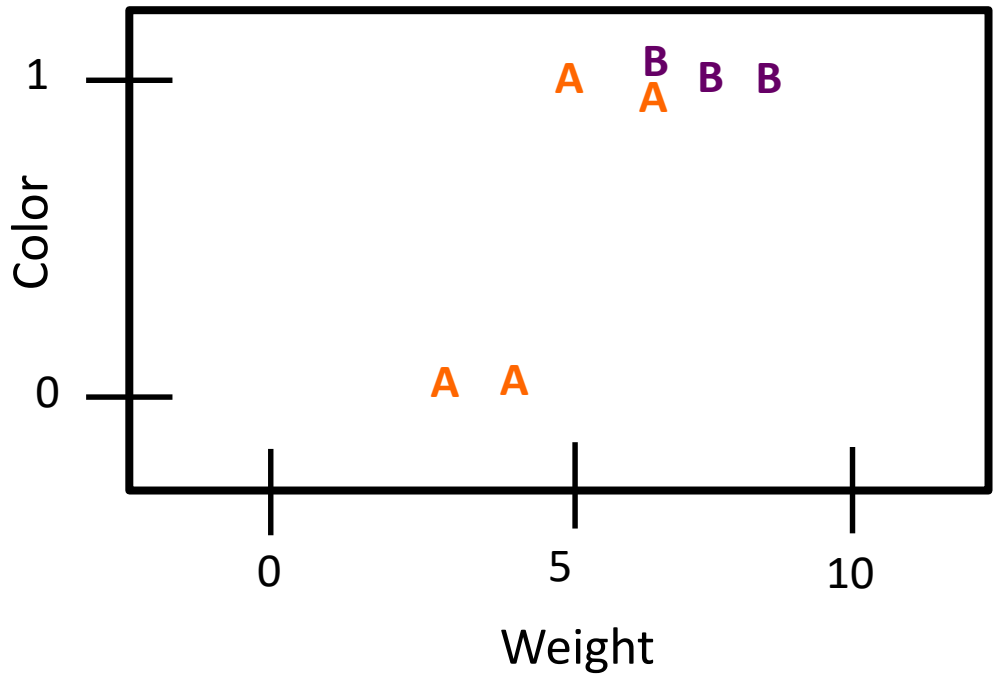
Weight	Color	Label
4	Red	Apple
5	Yellow	Apple
6	Yellow	Banana
3	Red	Apple
7	Yellow	Banana
8	Yellow	Banana
6	Yellow	Apple

Can we visualize this data?

Apples vs. Bananas

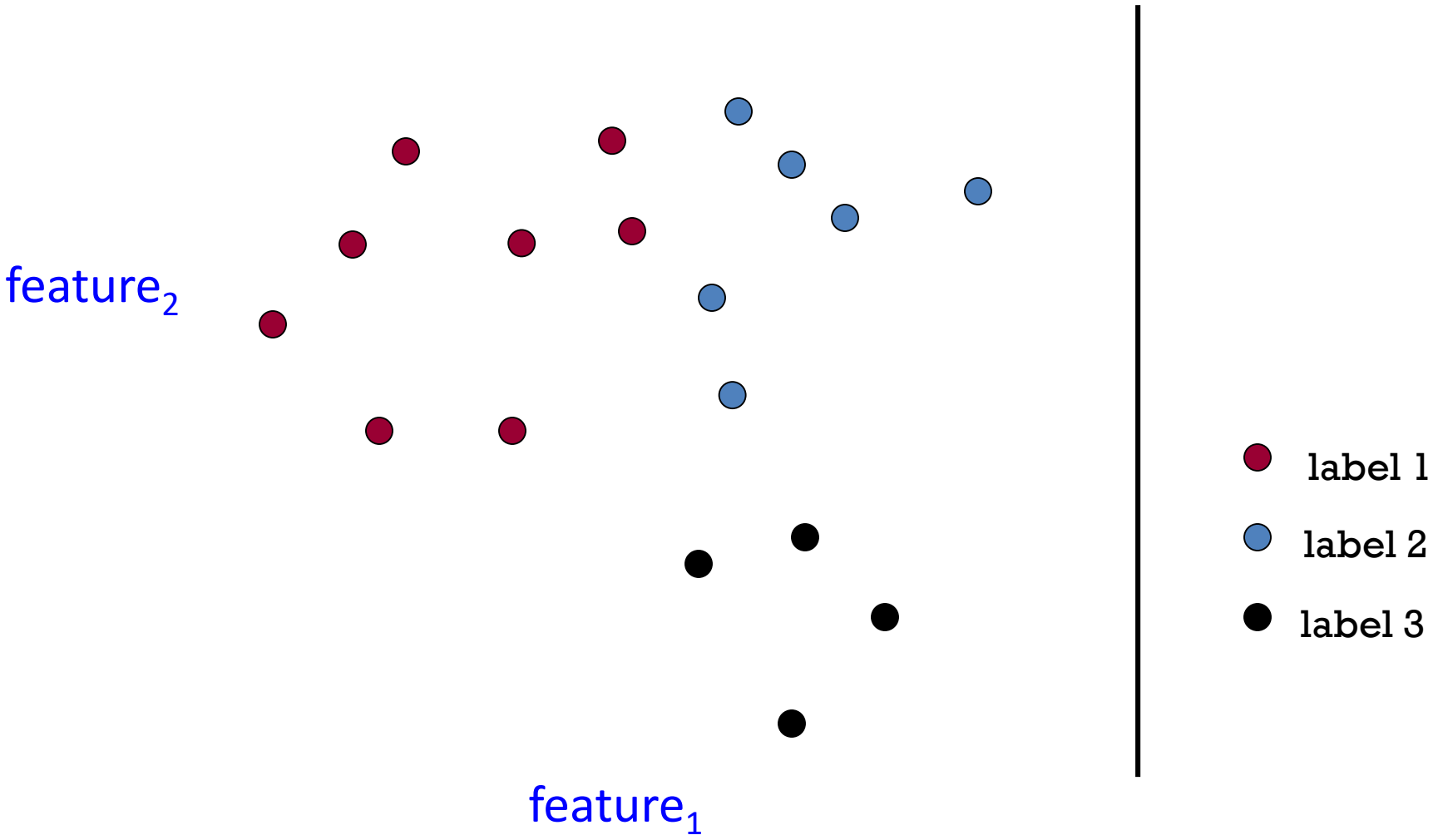
Turn features into numerical values

Weight	Color	Label
4	0	Apple
5	1	Apple
6	1	Banana
3	0	Apple
7	1	Banana
8	1	Banana
6	1	Apple

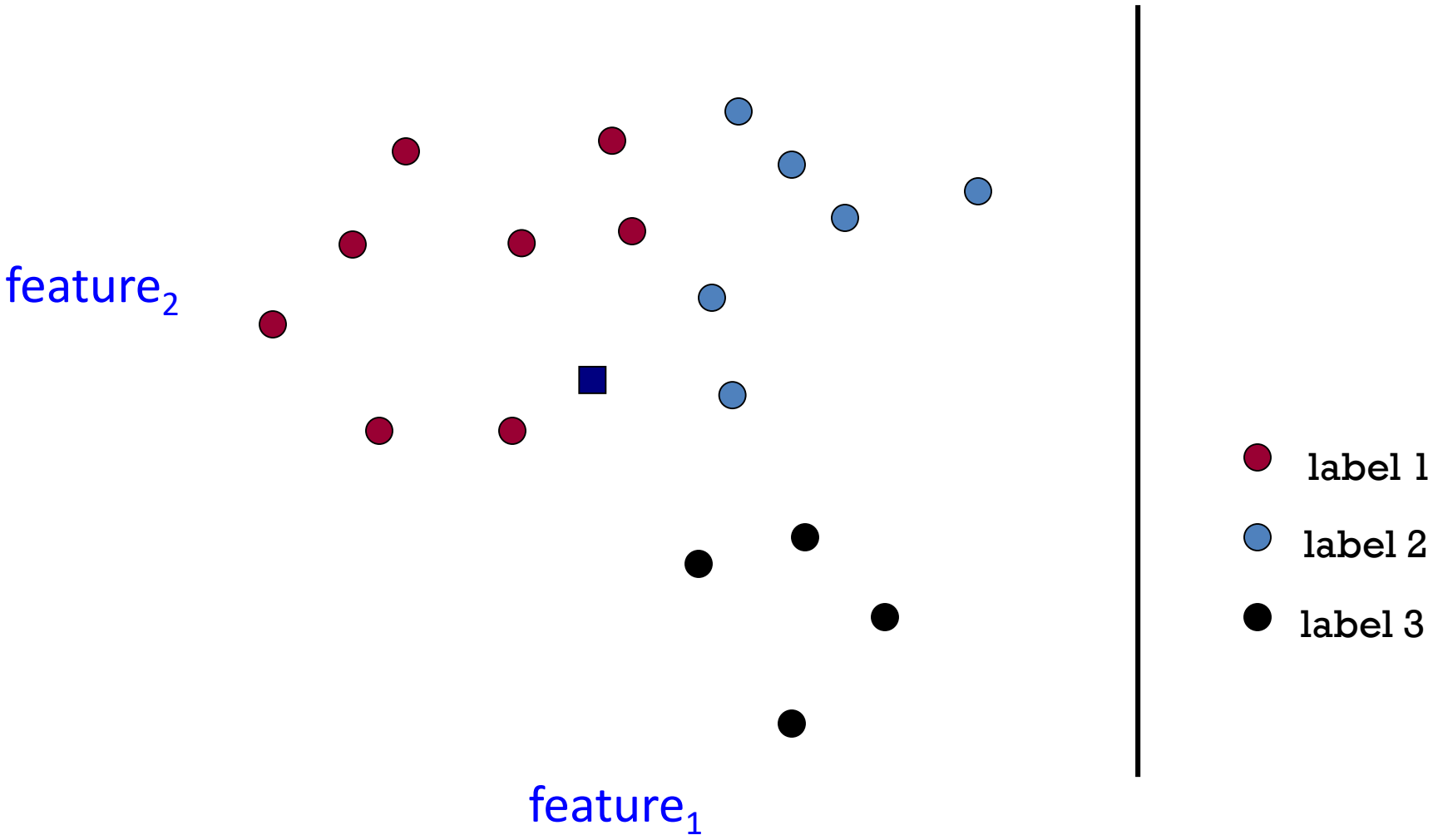


We can view examples as points in an *n*-dimensional space where *n* is the number of features

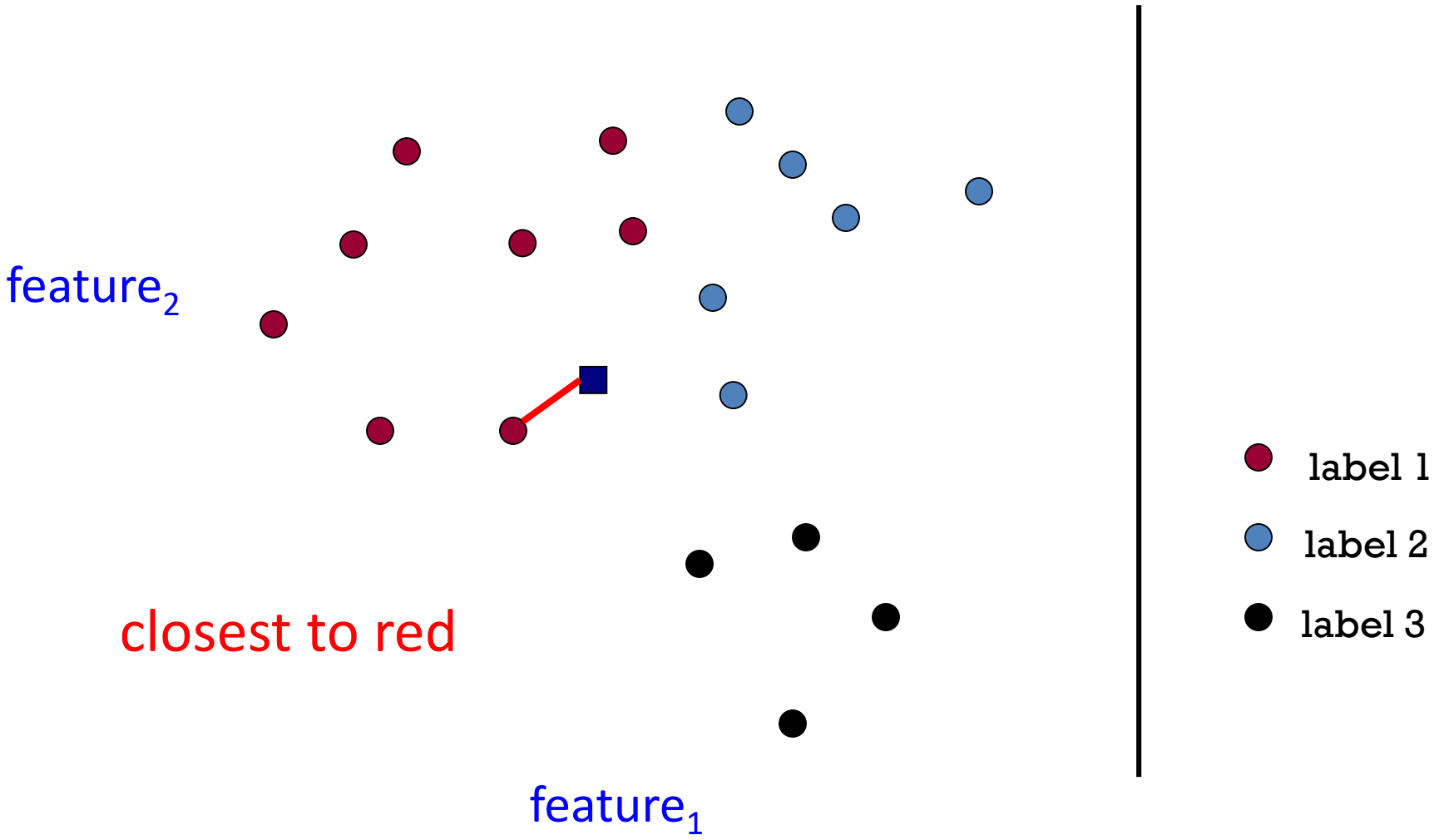
Examples in a feature space



Test example: what class?



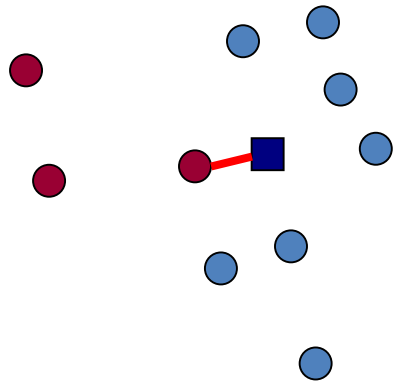
Test example: what class?



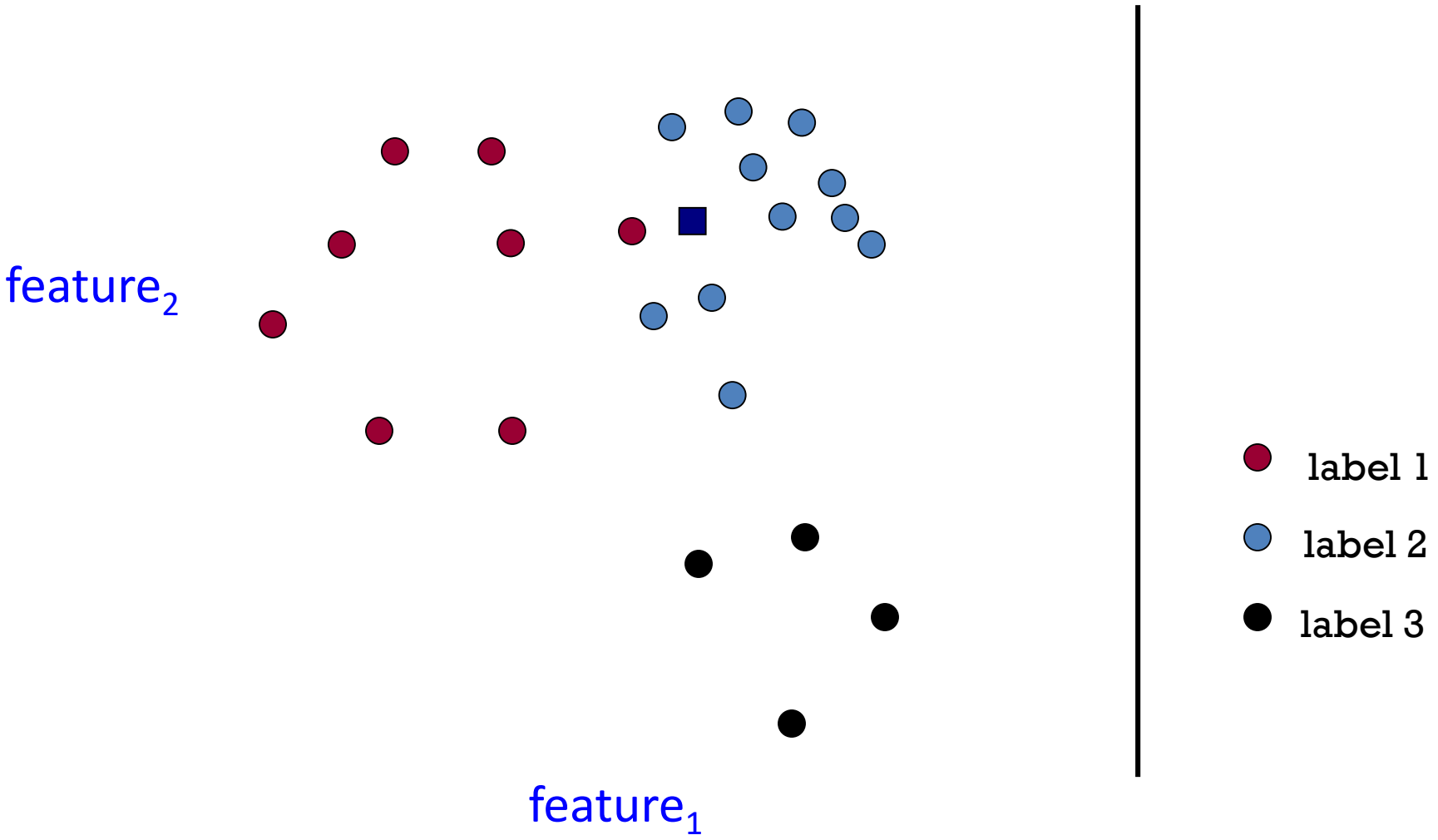
Another classification algorithm?

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

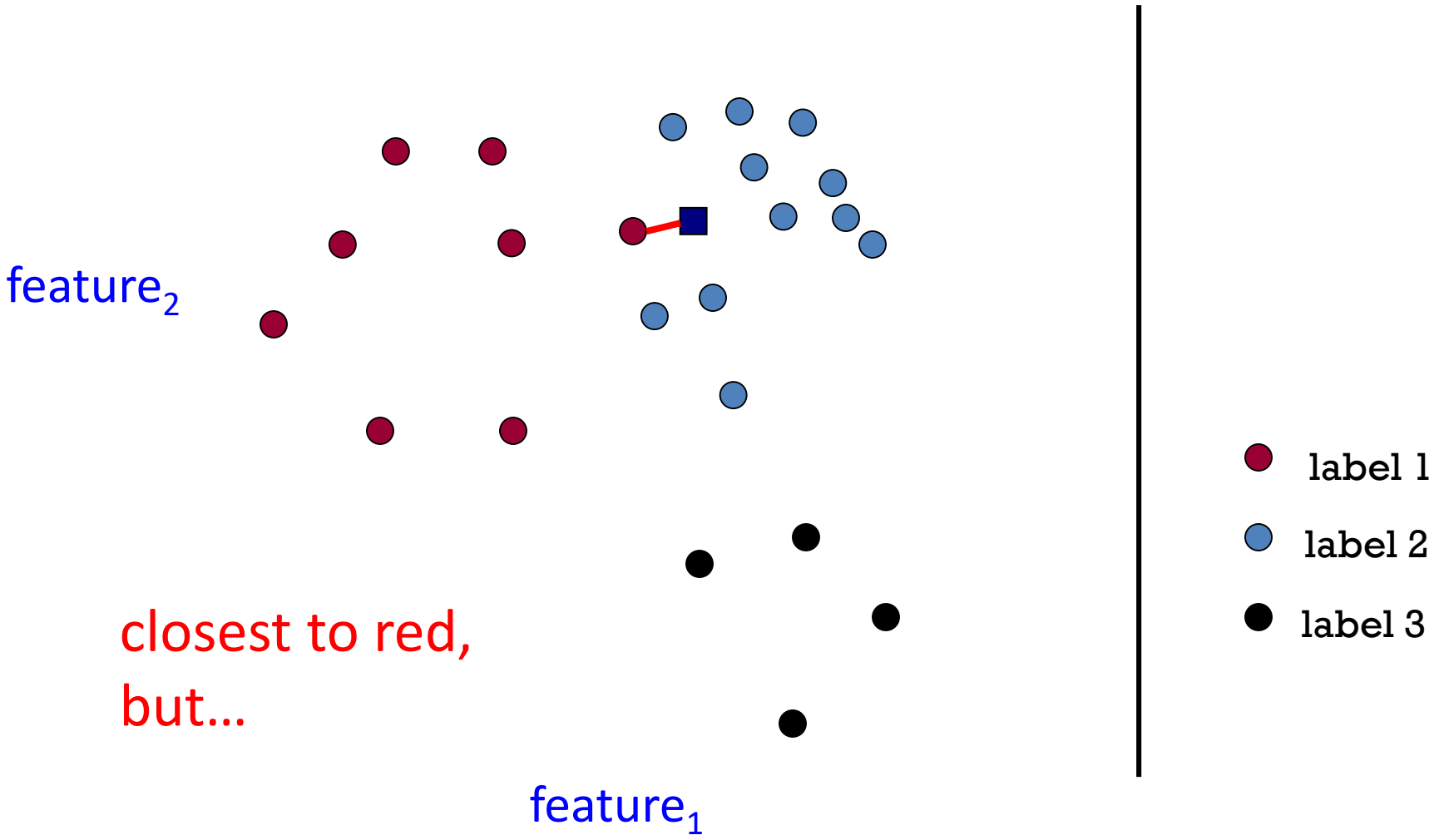
Label ***d*** with the label of the closest example to ***d*** in the training set



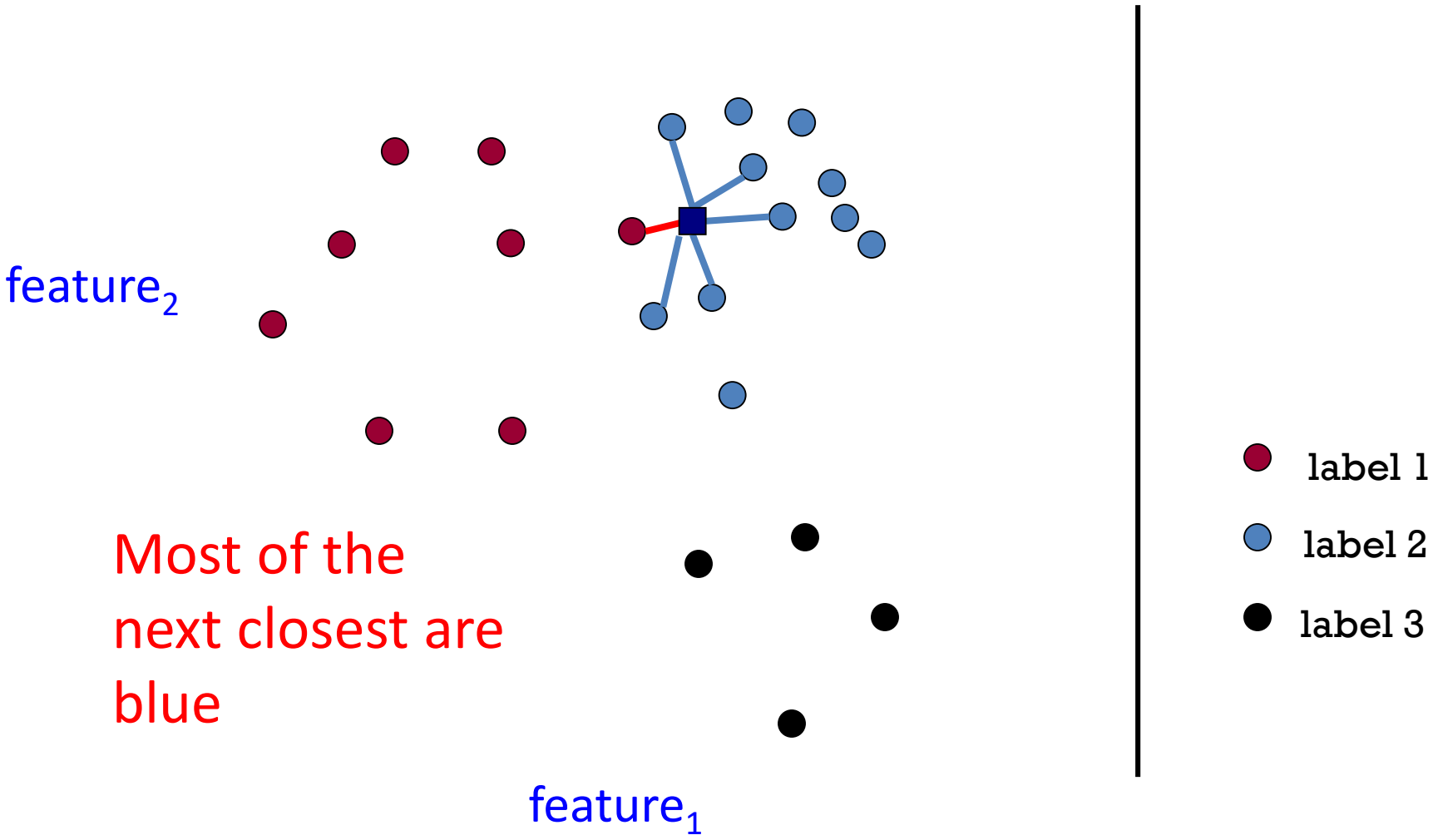
What about this example?



What about his example?



What about his example?



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

k-Nearest Neighbor (k-NN)

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

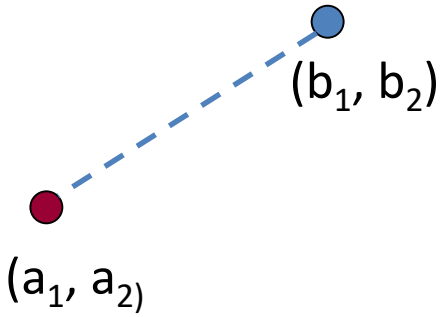
Find ***k*** *nearest* neighbors of ***d***

Choose as the label the majority label within the ***k*** nearest neighbors

How do we measure “nearest”?

Euclidean distance

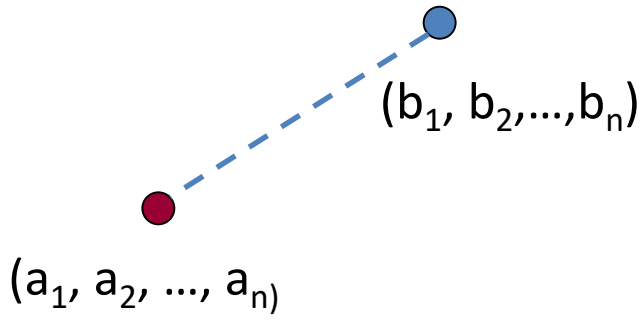
In two dimensions, how do we compute the distance?



$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

Euclidean distance

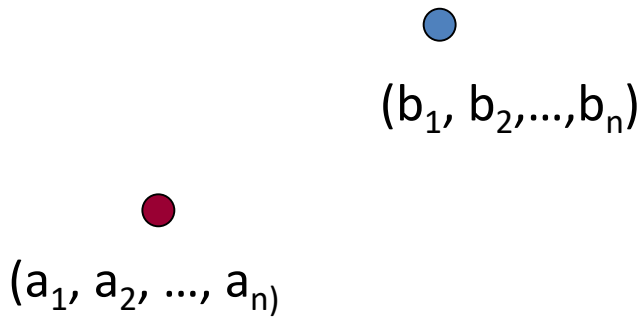
In n-dimensions, how do we compute the distance?



$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

Euclidean distance

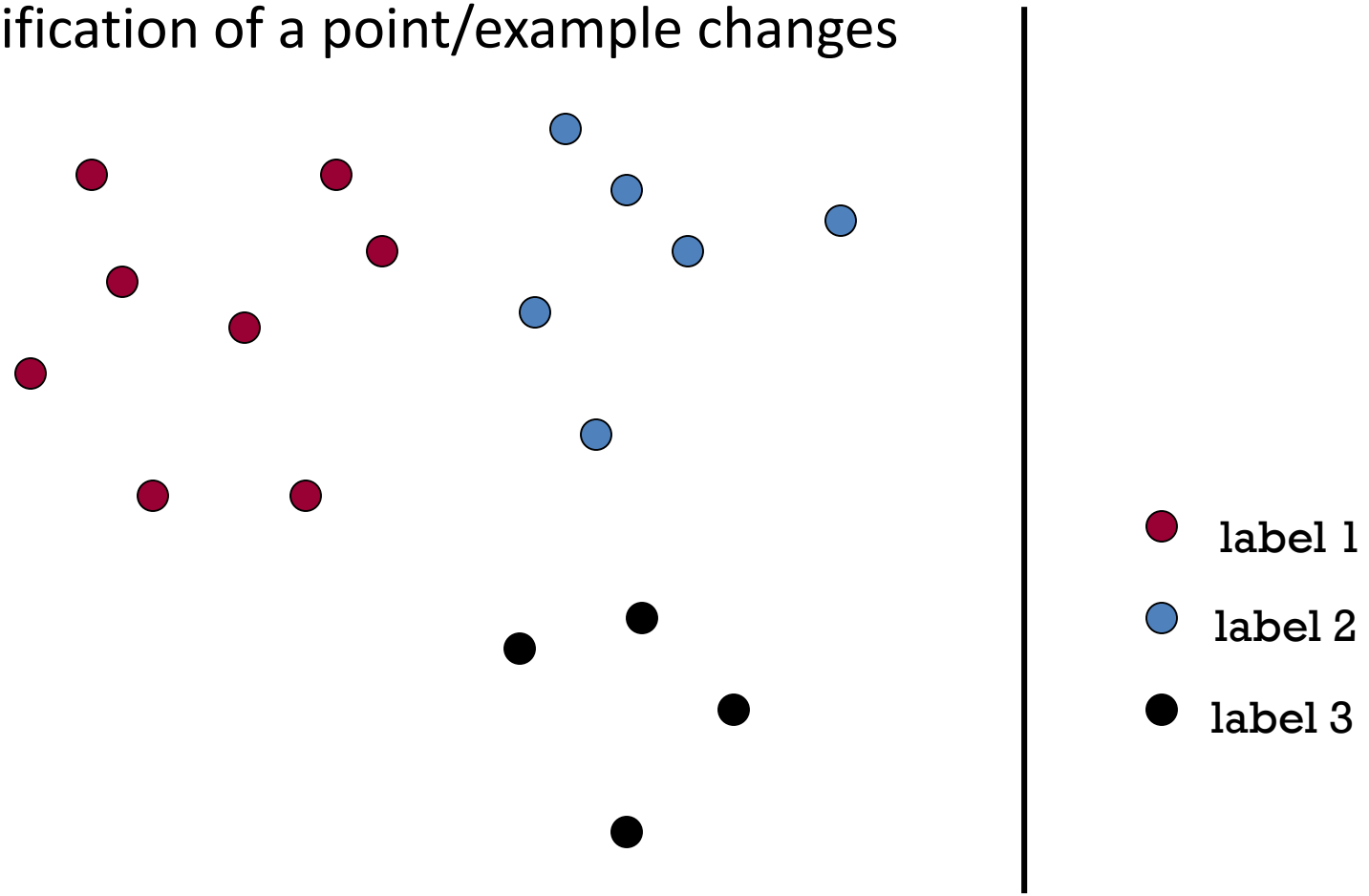
In n-dimensions, how do we compute the distance?



Measuring distance/similarity is a domain-specific problem and there are many, many different variations!

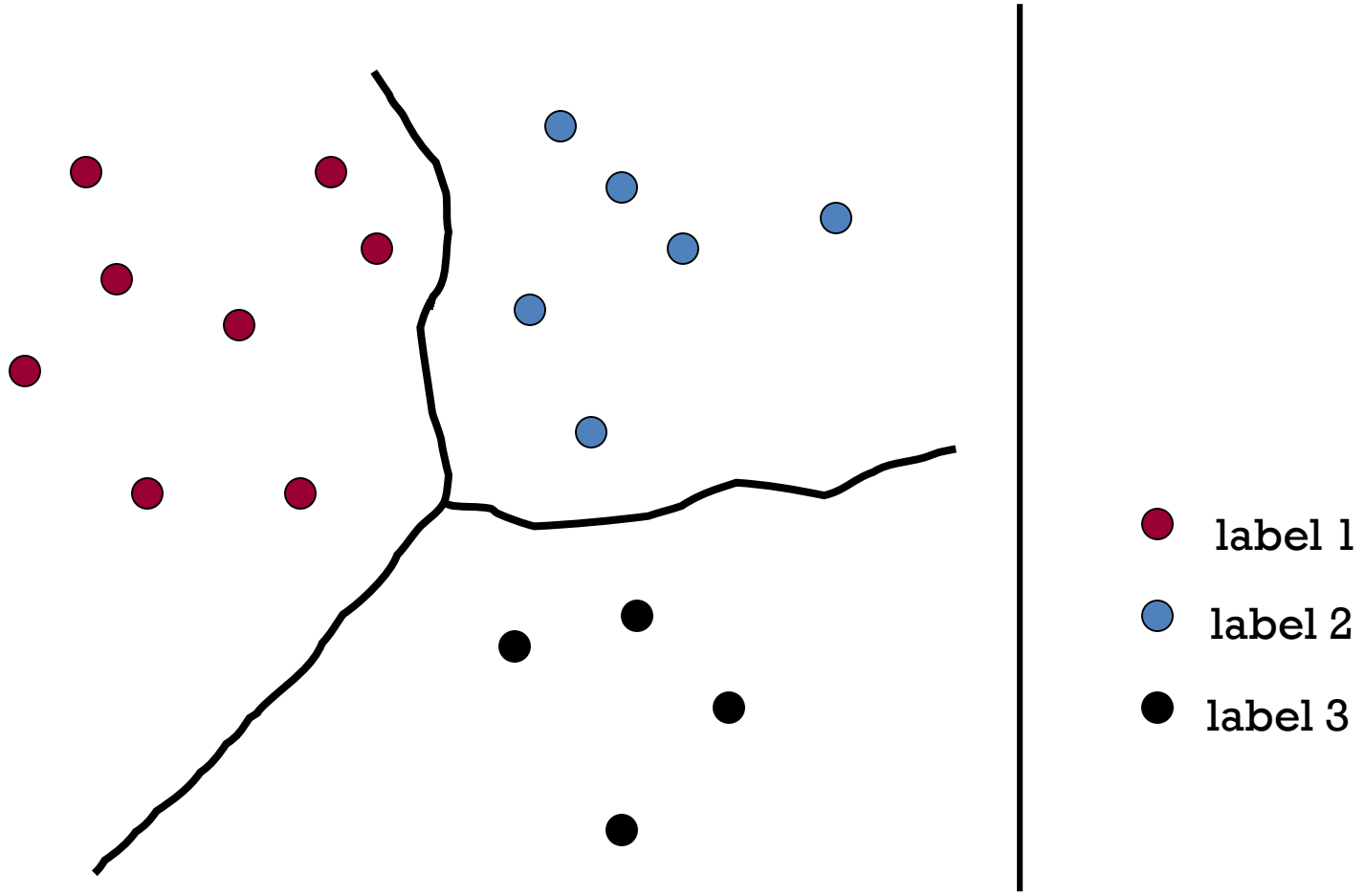
Decision boundaries

The **decision boundaries** are places in the features space where the classification of a point/example changes



Where are the decision boundaries for k-NN?

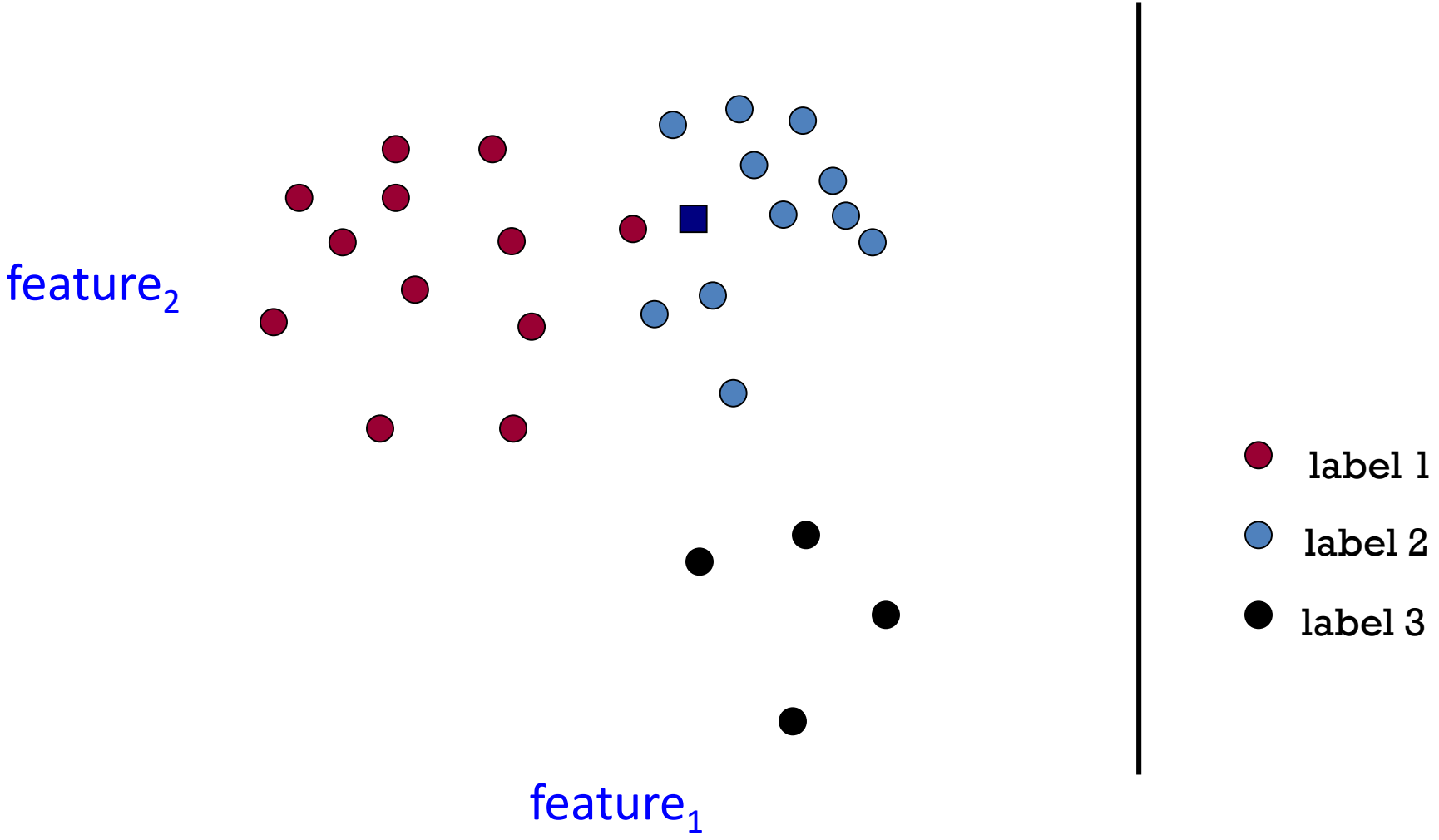
k-NN decision boundaries



k-NN gives locally defined decision boundaries between classes

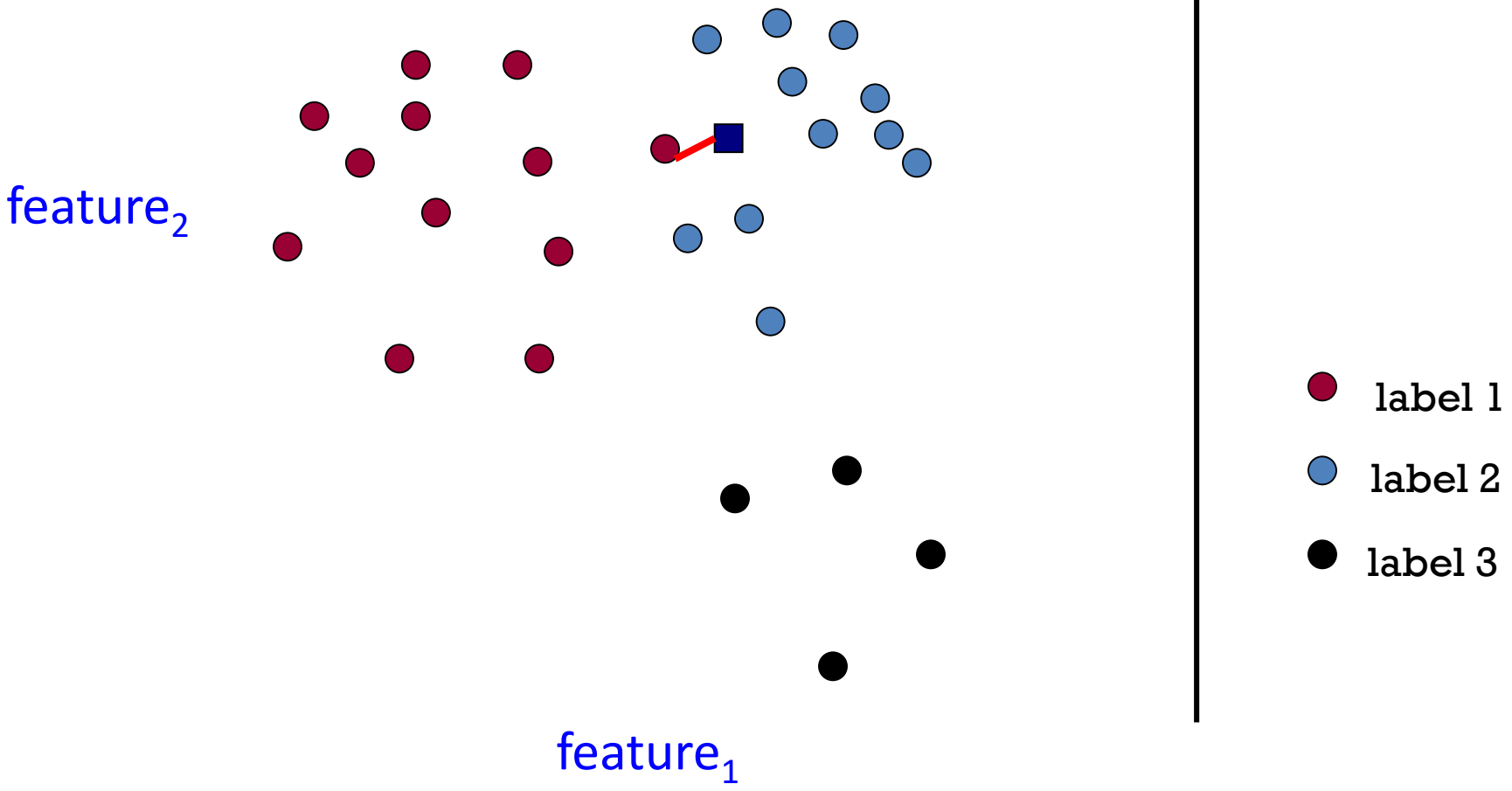
Choosing k

What is the label with k = 1?



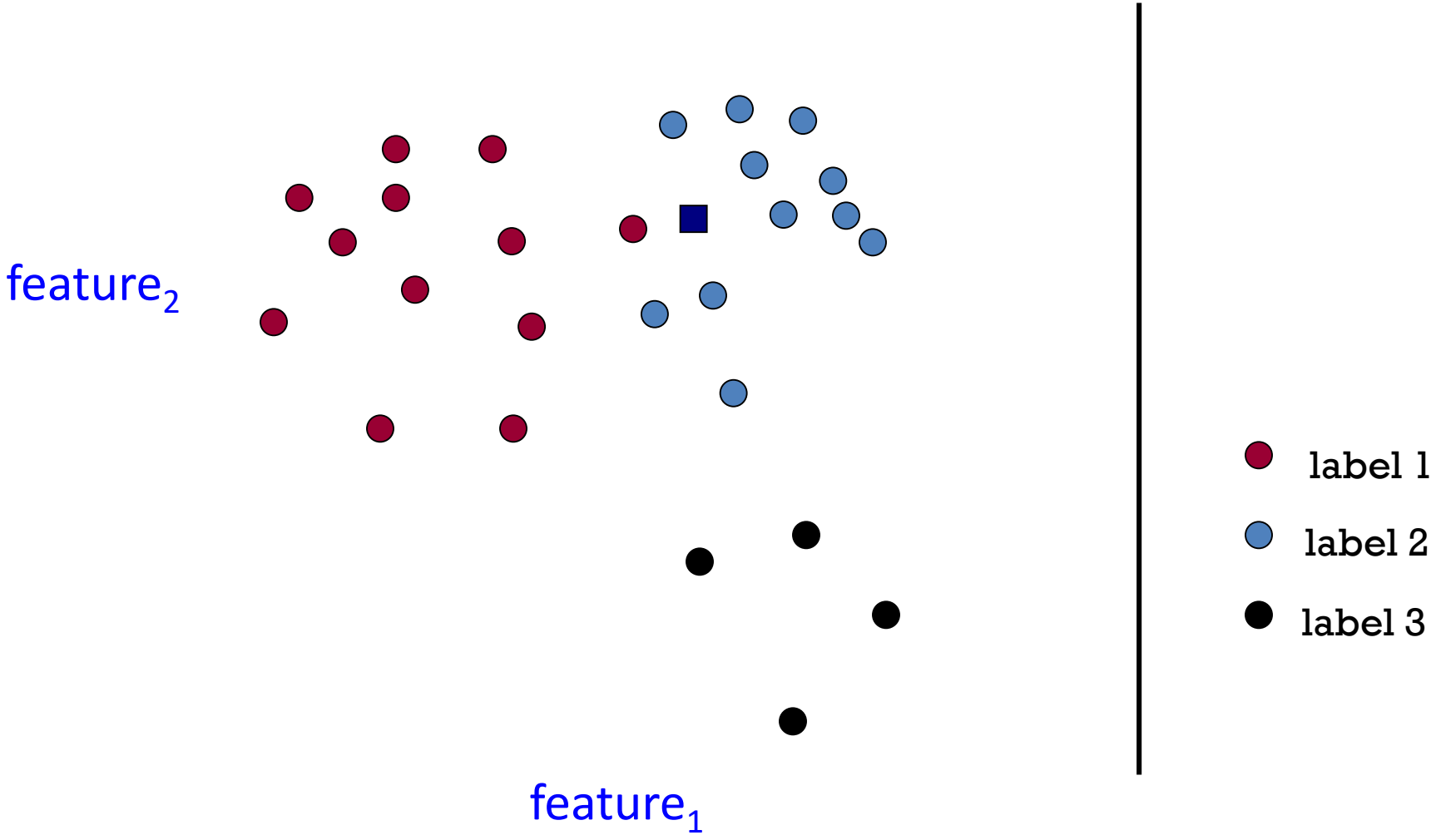
Choosing k

We'd choose red. Do you agree?



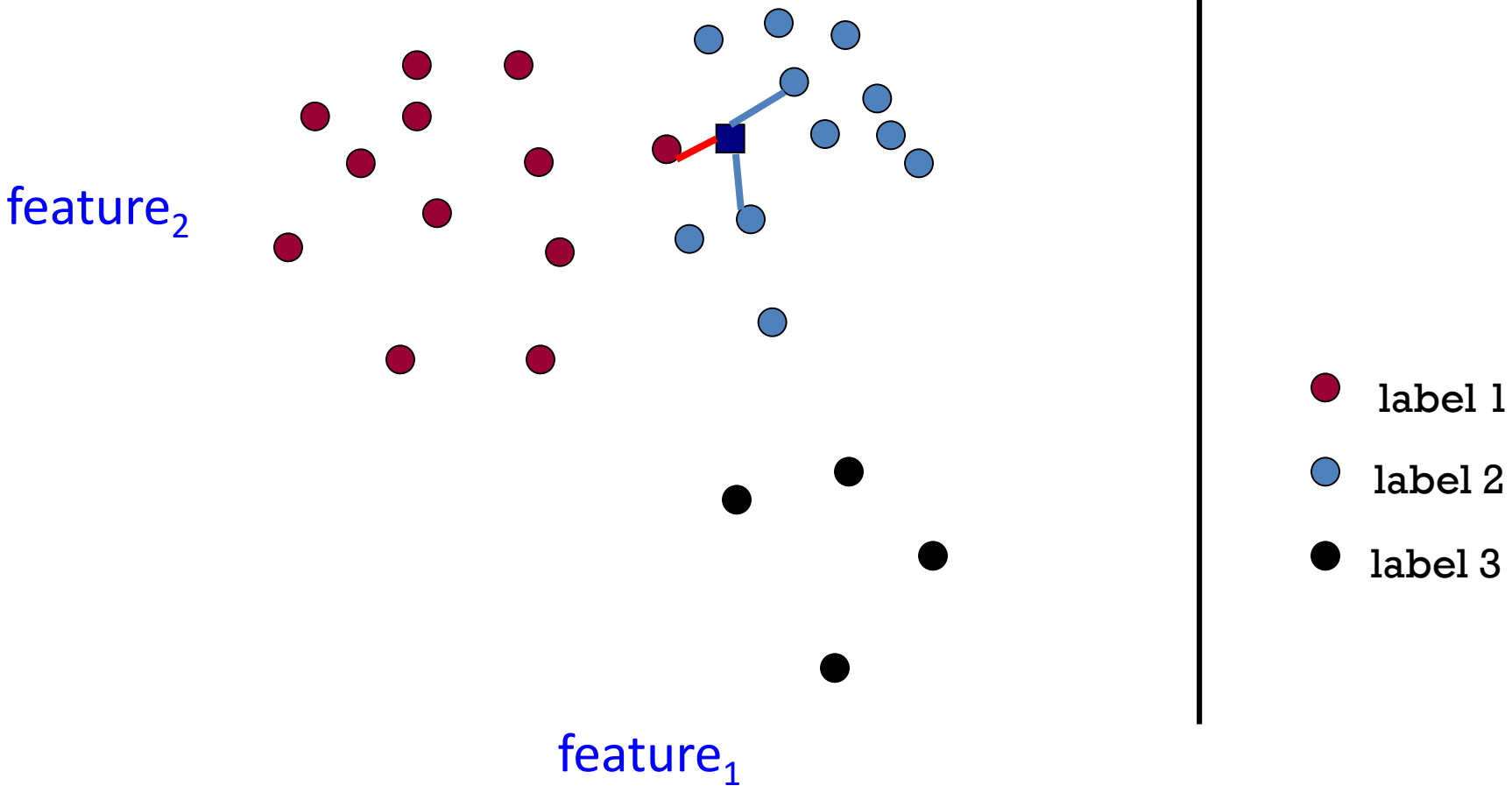
Choosing k

What is the label with k = 3?



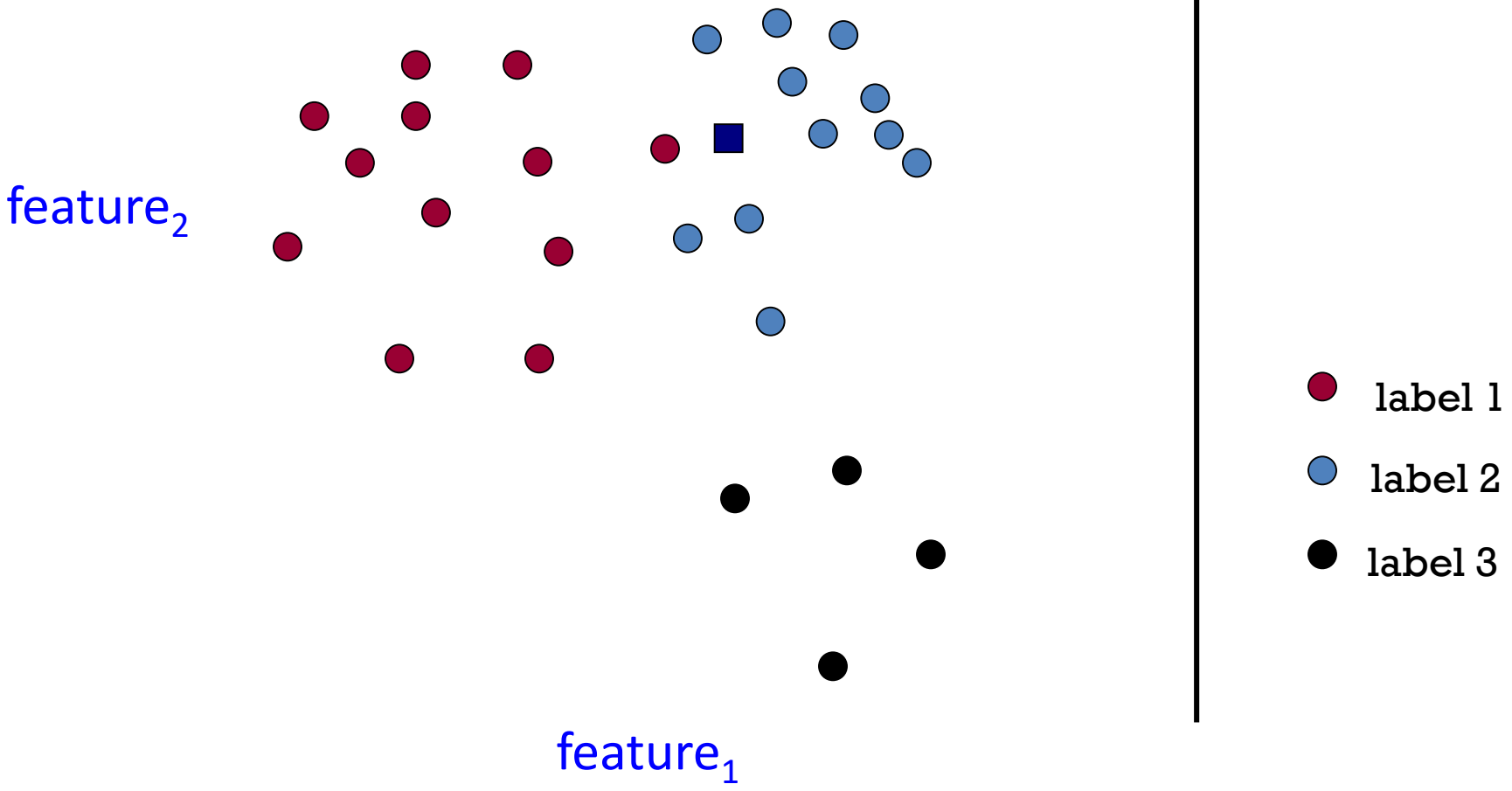
Choosing k

We'd choose blue. Do you agree?



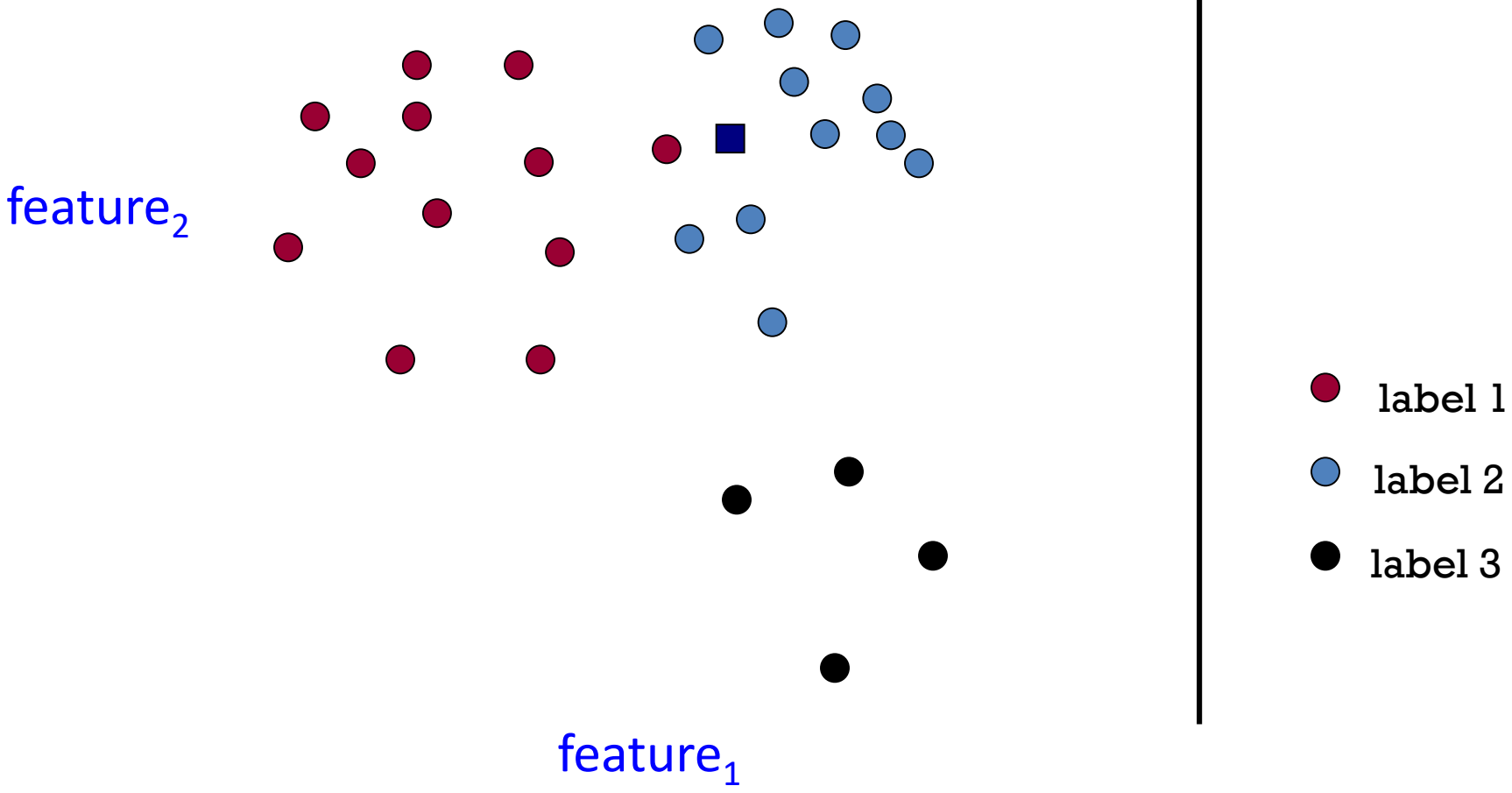
Choosing k

What is the label with k = 100?

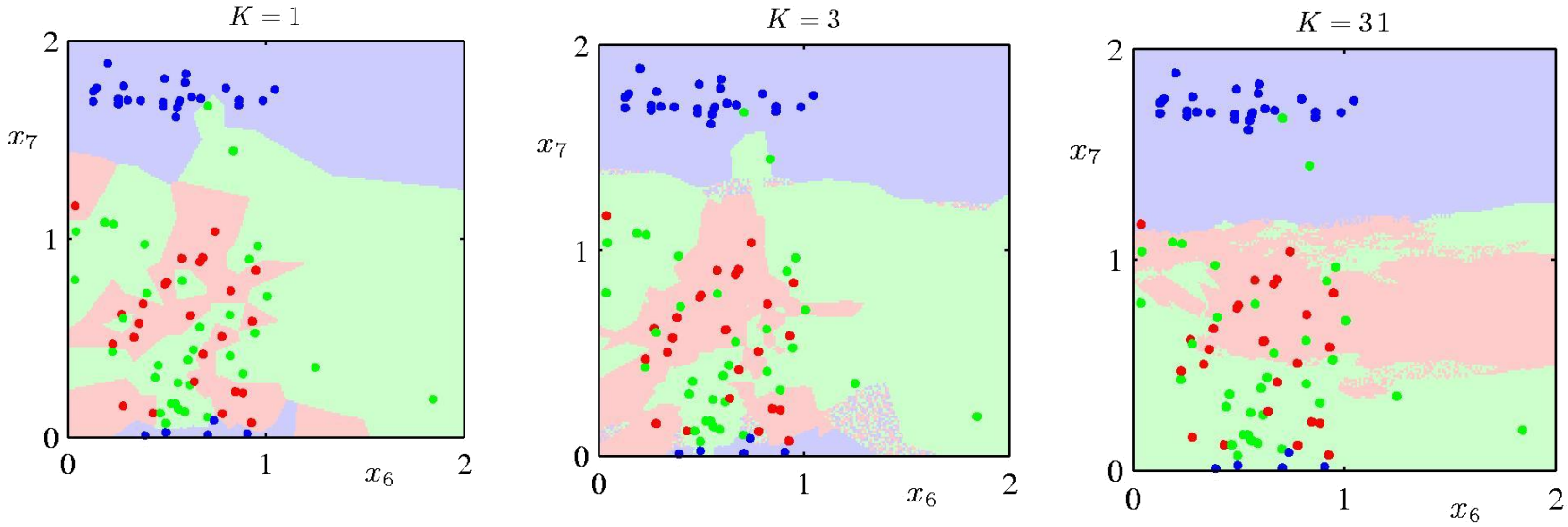


Choosing k

We'd choose red. Do you agree?



The impact of k



What is the role of k? Defines the decision boundary, controls the degree of smoothing

How does it relate to overfitting and underfitting? Low k may lead to overfitting (decision boundary tries to separate every single instance), large k may lead to underfitting (decision boundary generalizes in the separation)

How did we control this for decision trees? E.g. with tree depth parameter

k-Nearest Neighbor (k-NN)

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

Find ***k*** nearest neighbors of ***d***

Choose as the class the majority class within the ***k*** nearest neighbors

How do we choose *k*?

How to pick k

Common heuristics:

often 3, 5, 7

choose an odd number to avoid ties

Use development data

- Experiment with different k values (using cross validation for example) and choose the best k (that has best performance measure)

k-NN variants

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

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

Any variation ideas?

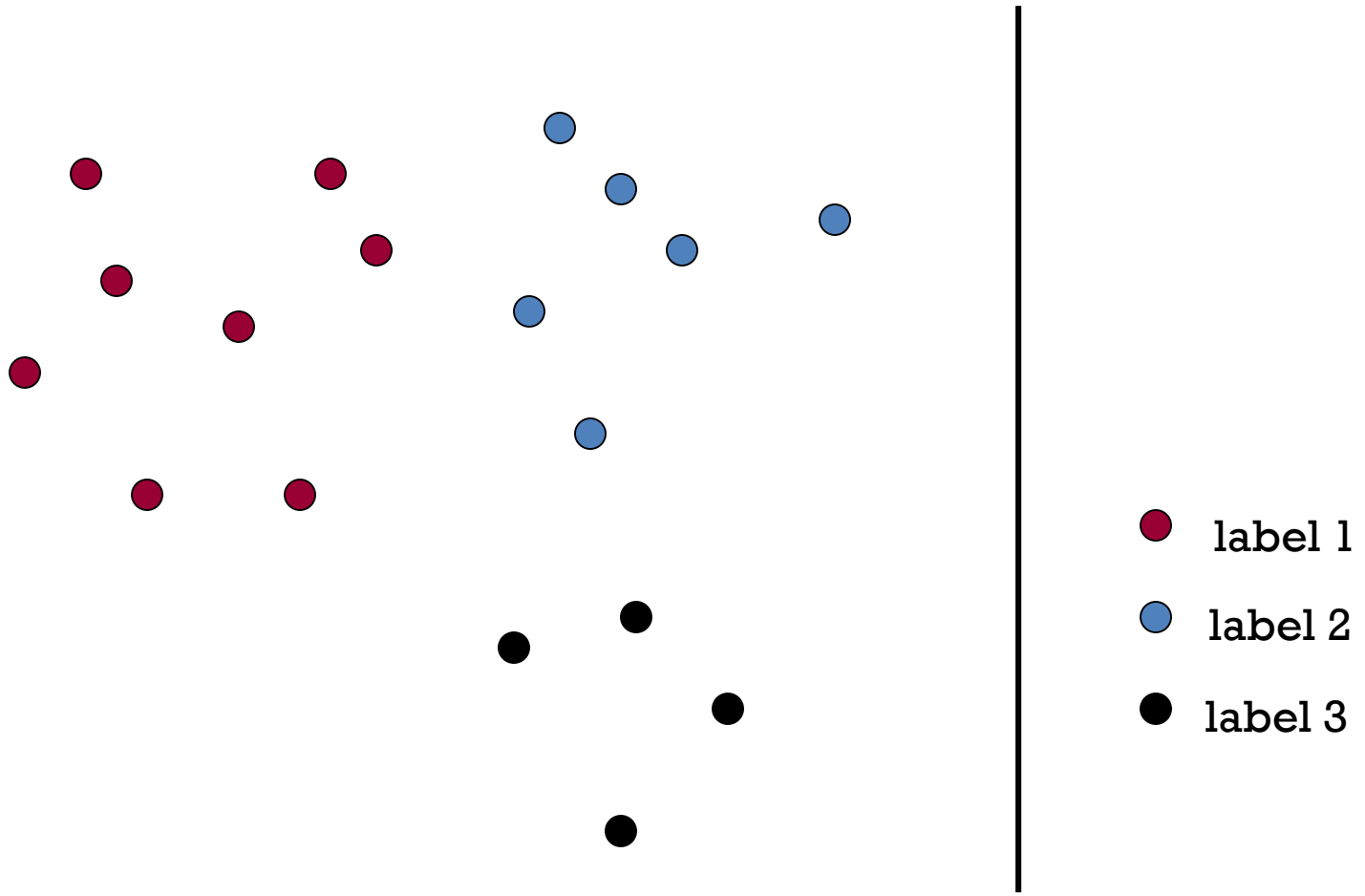
k-NN variations

Instead of *k* nearest neighbors, count majority from all examples within a fixed distance

Weighted *k*-NN:

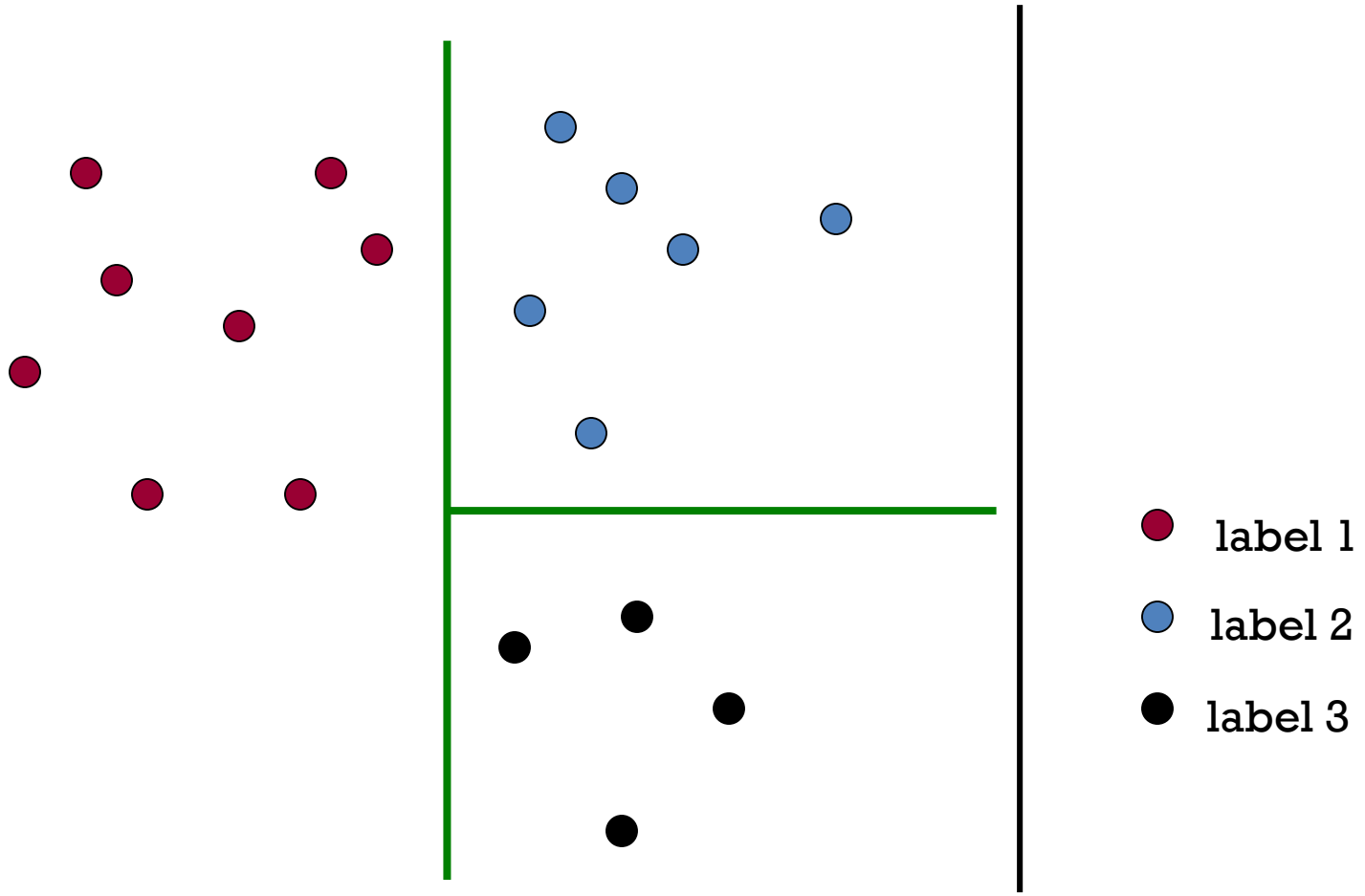
- Right now, all examples are treated equally
- weight the “vote” of the examples, so that closer examples have more vote/weight

Decision boundaries for decision trees



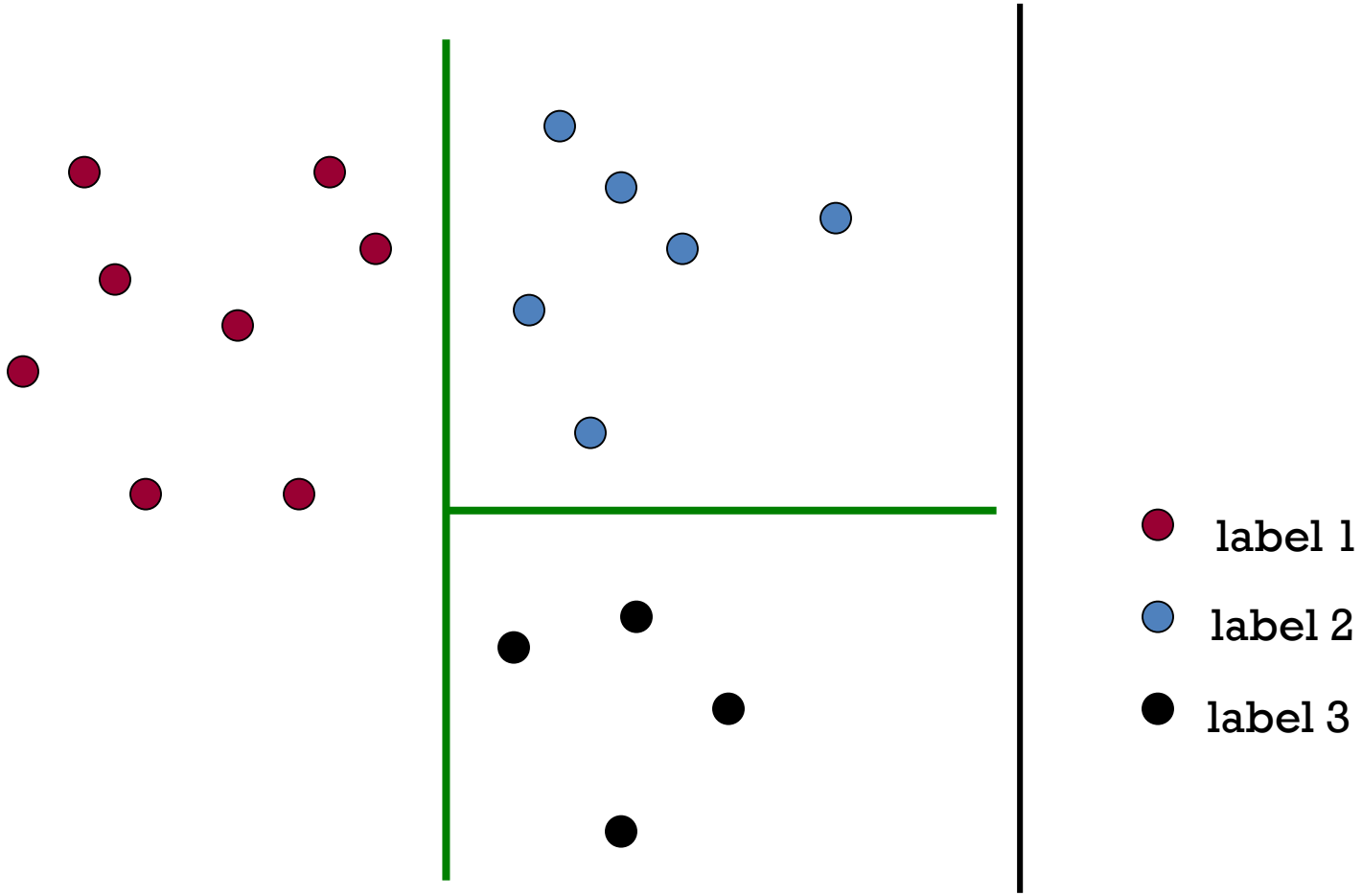
What are the decision boundaries for decision trees like?

Decision boundaries for decision trees



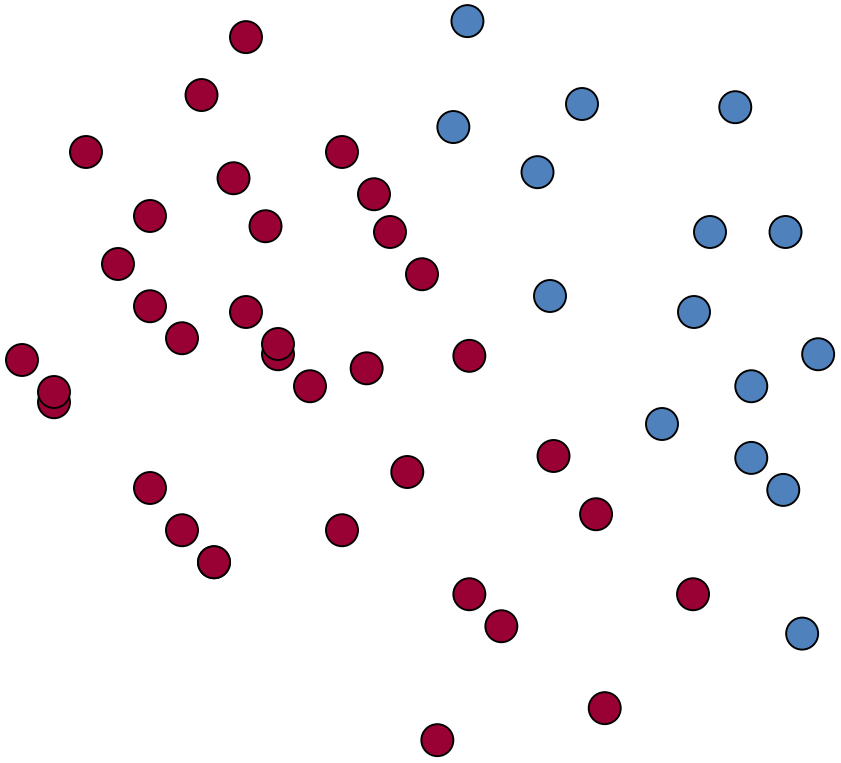
Axis-aligned splits/cuts of the data

Decision boundaries for decision trees



What types of data sets will DT work poorly on?

Problems for DT



Decision trees vs. *k*-NN

Which is faster to train?

Which is faster to classify?

Do they use the features in the same way to label the examples?

Decision trees vs. *k*-NN

Which is faster to train?

k-NN doesn't require any training

Which is faster to classify?

For most data sets, decision trees

Do they use the features in the same way to label the examples?

k-NN treats all features equally. Decision trees "select" important features

Machine learning models

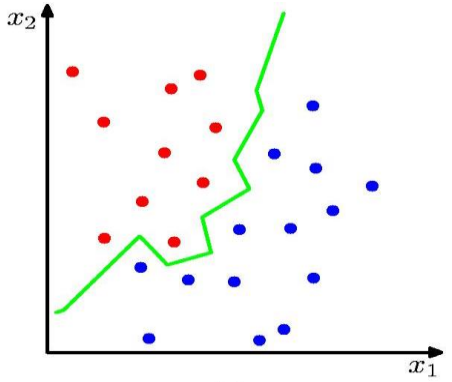
- Some machine learning approaches make strong assumptions about the data
  - If the assumptions are true it can often lead to better performance
  - If the assumptions aren't true, the approach can fail miserably
- Other approaches don't make many assumptions about the data
  - This can allow us to learn from more varied data
  - But, they are more prone to overfitting and generally require more training data

Machine learning models

What are the *model* assumptions (if any) that *k*-NN and decision trees make about the data?

K-NN

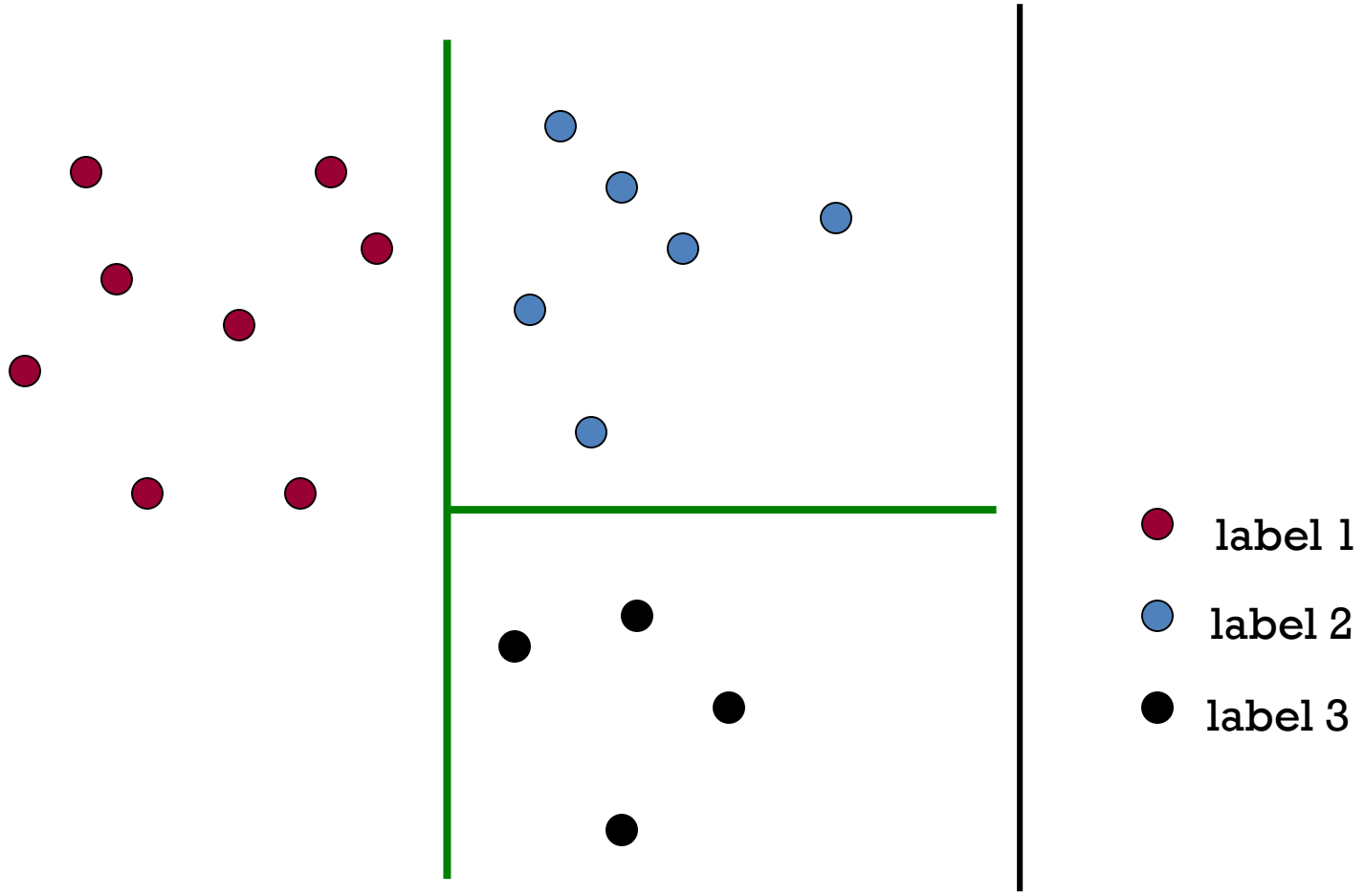
- No model assumptions: this why K-NN is called non-parametric
- It only assumes that proximity relates to class



- K-NN is an Instance-based learning algorithm: it doesn't explicitly learn a model. Instead, it chooses to memorize the training instances which are subsequently used as "knowledge" for the prediction phase.

Decision tree model

Axis-aligned splits/cuts of the data



Bias

The “bias” of a model is how strong the model assumptions are.

low-bias classifiers make minimal assumptions about the data (*k*-NN and DT are generally considered low bias)

high-bias classifiers make strong assumptions about the data

- E.g. linear classifiers

Summary

- KNN can be used for regression
- Instead of outputing the majority class (in case of classification), output the mean labels of neighbors

Summary: Pros and Cons of KNN

Pros:

- Simple to understand and easy to implement.
- KNN works just as easily with multiclass data sets whereas other algorithms are hardcoded for the binary setting.
- Non-parametric nature of KNN gives it an edge in certain settings where the data may be highly “unusual”.

Cons:

- Computationally expensive testing phase
- KNN can suffer from skewed class distributions. For example, if a certain class is very frequent in the training set, it will tend to dominate the majority voting of the new example (large number = more common).
- Accuracy can be severely degraded with high-dimension data

Summary: Pros and Cons of KNN

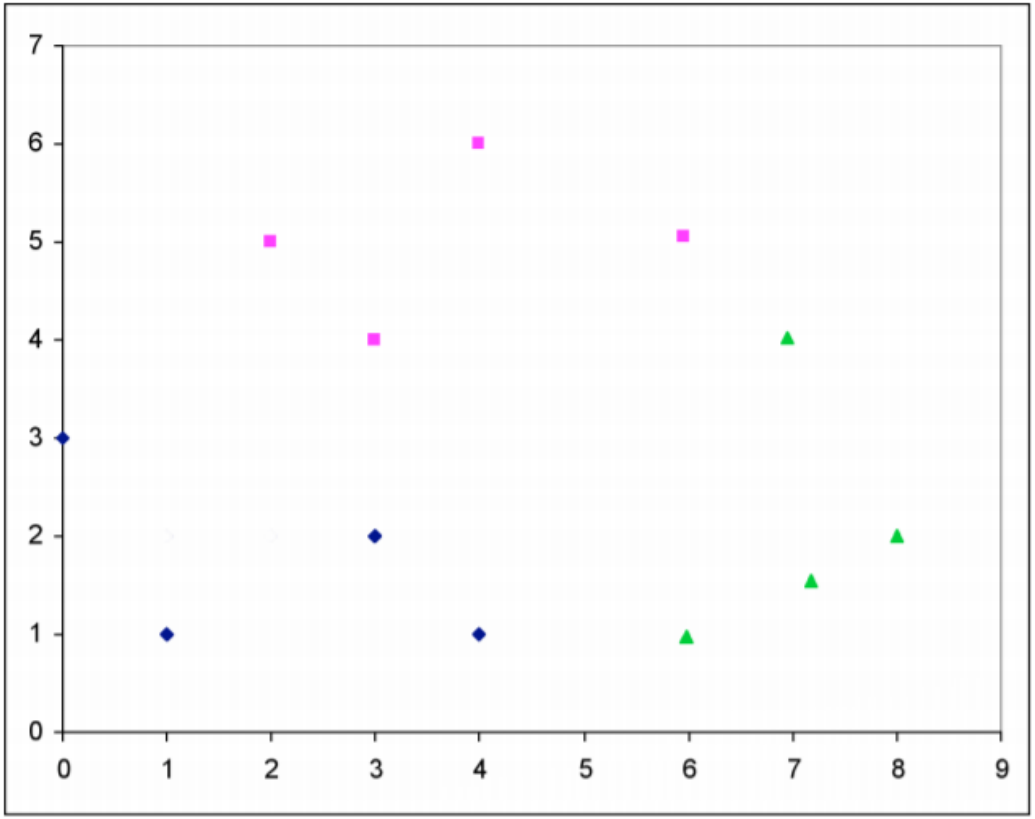
Cons:

- Sensitive to outliers
- Need to handle missing data before using KNN

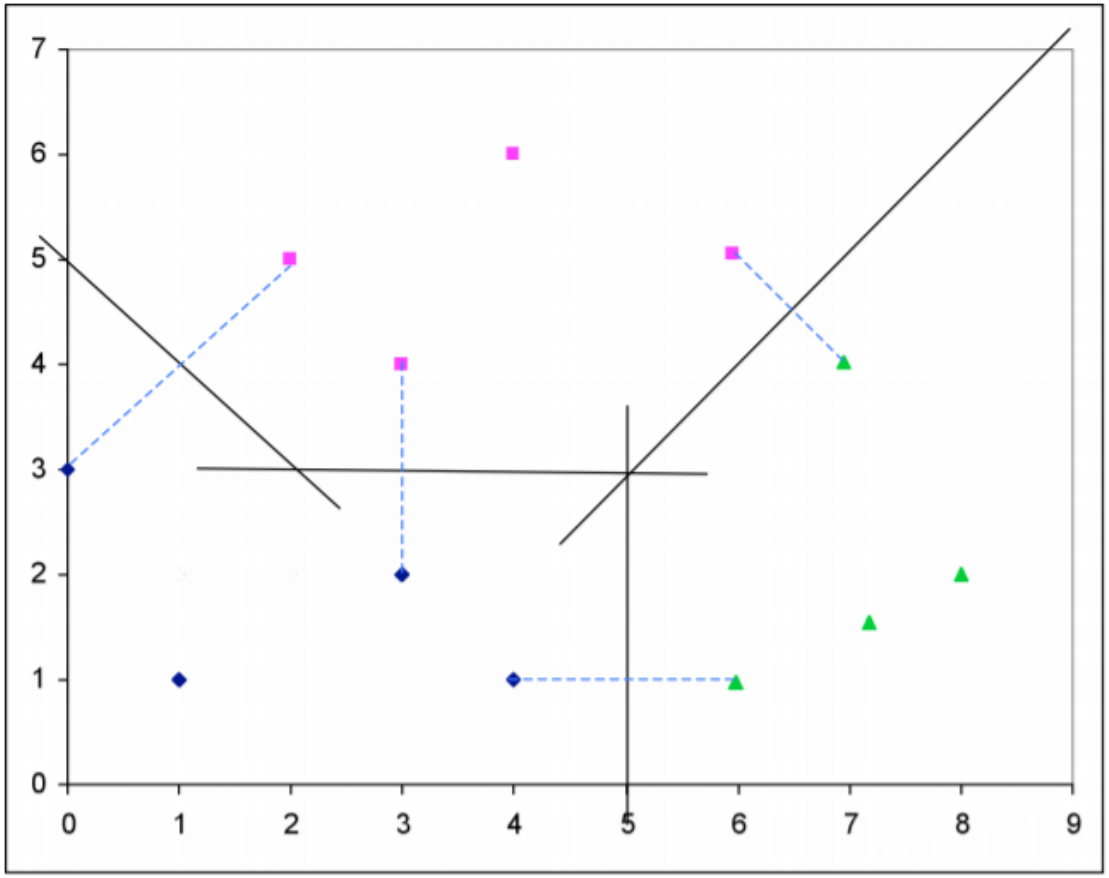
Summary: Improvements to KNN

- For skewed class distributions: use weighted K-NN. This ensures that nearer neighbors contribute more to the final vote than the more distant ones.
- Changing the distance metric for different applications may help improve the accuracy of the algorithm. (e.g. Hamming distance for text classification)
- Rescaling your data makes the distance metric more meaningful. For instance, given 2 features height and weight, an observation such as x=[180,70] will clearly skew the distance metric in favor of height. One way of fixing this is by normalizing data
- Dimensionality reduction techniques like PCA should be executed prior to appplying KNN and help make the distance metric more meaningful.

Extra: KNN Decision boundaries



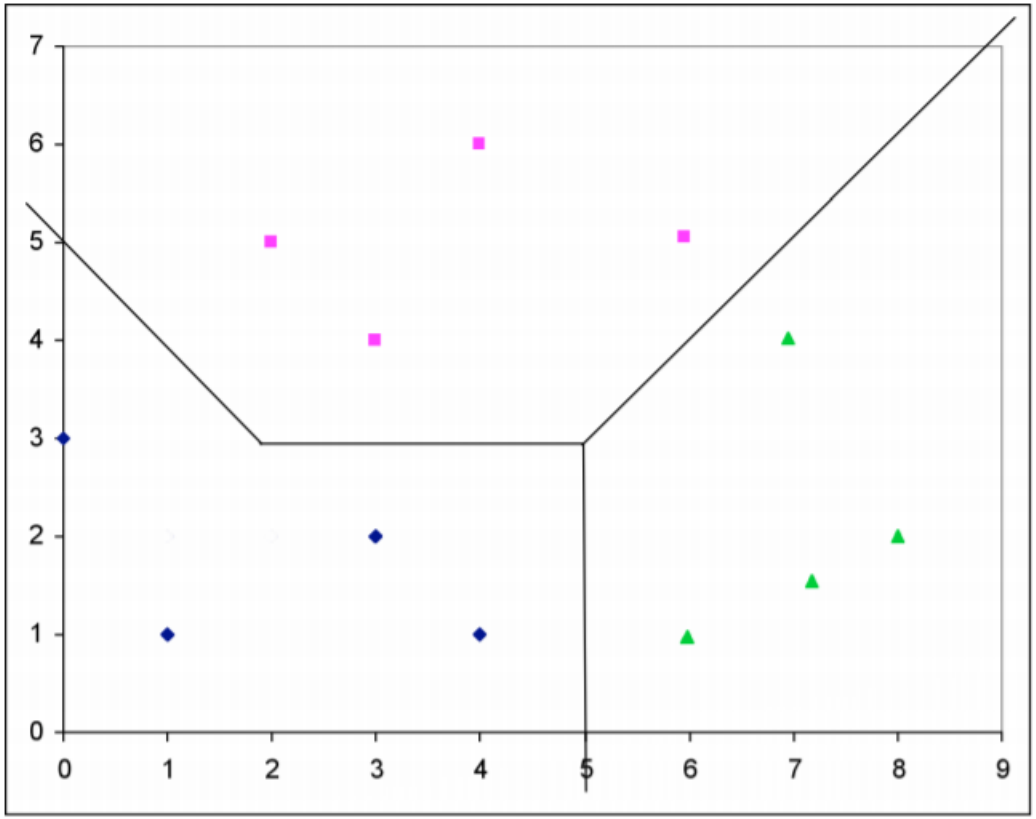
Extra: KNN Decision boundaries



Construct lines between closest pairs of points in different classes.

Draw perpendicular bisectors.

Extra: KNN Decision boundaries



End bisectors at intersections; extend beyond axes (to infinity).

Reading

A Course in Machine Learning. Hal Daumé III (HL)

- Chapter 3

Shalev-Shwartz and Ben-David (2014) Understanding Machine Learning: From Theory to Algorithms (MLA)

- 2.1 and 2.2