Local Methods

CS771: Introduction to Machine Learning

Purushottam Kar

Announcements

Registered students would have received a Piazza invitation on their CC email IDs – please activate your account and join discussions

Please do not forget to form groups of 5 registered students – will be asked to submit group names next week itself

Code repository - https://tinyurl.com/ml19-20ac

Will contain lecture code as well as lecture notes

Tip: do not download individual file from repository – instead, do a git pull operation so that all updates (to old files as well) are received

Recap of Last Lecture

What are features – their types and how are they used in ML

Storing features as vectors – common ML operations on vectors

Learning with Prototypes (LwP) – an extremely simple method that gives lightweight models (just one prototype per class)

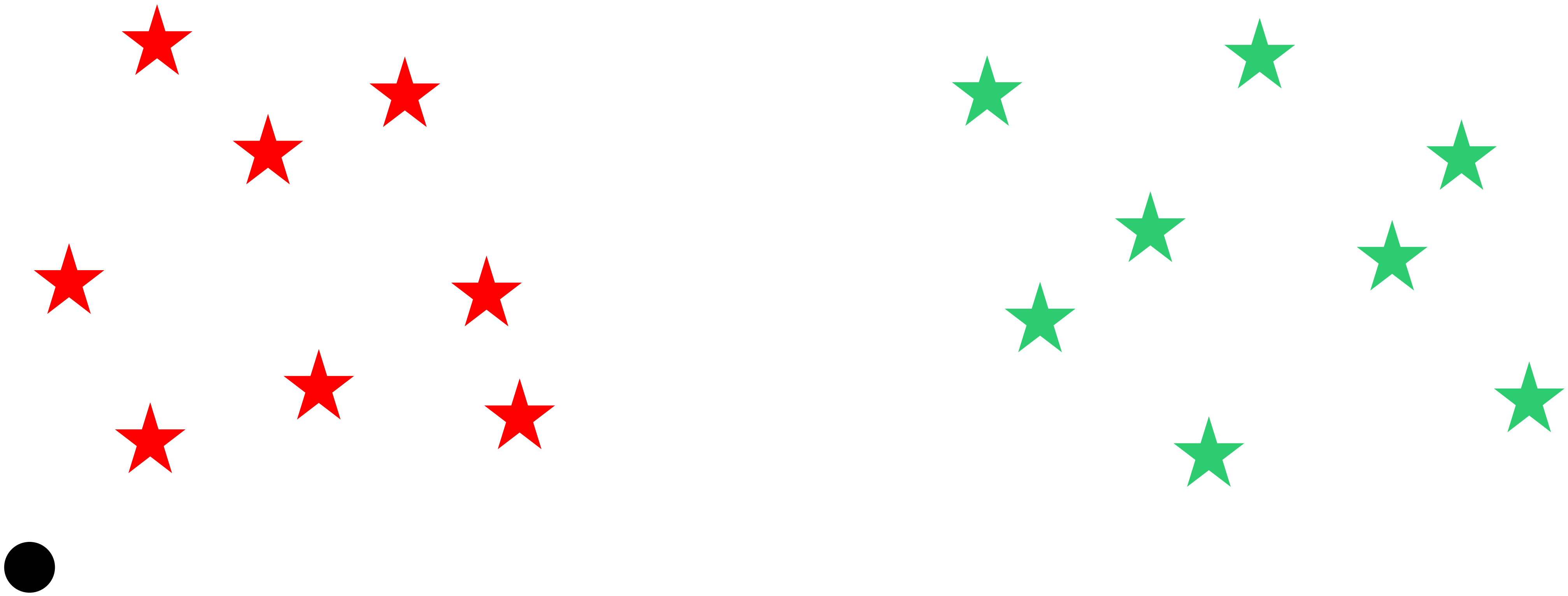
When LwP fails – when data points in class are very diverse, or else oddly distributed

One solution – make every training data point a prototype – 1NN

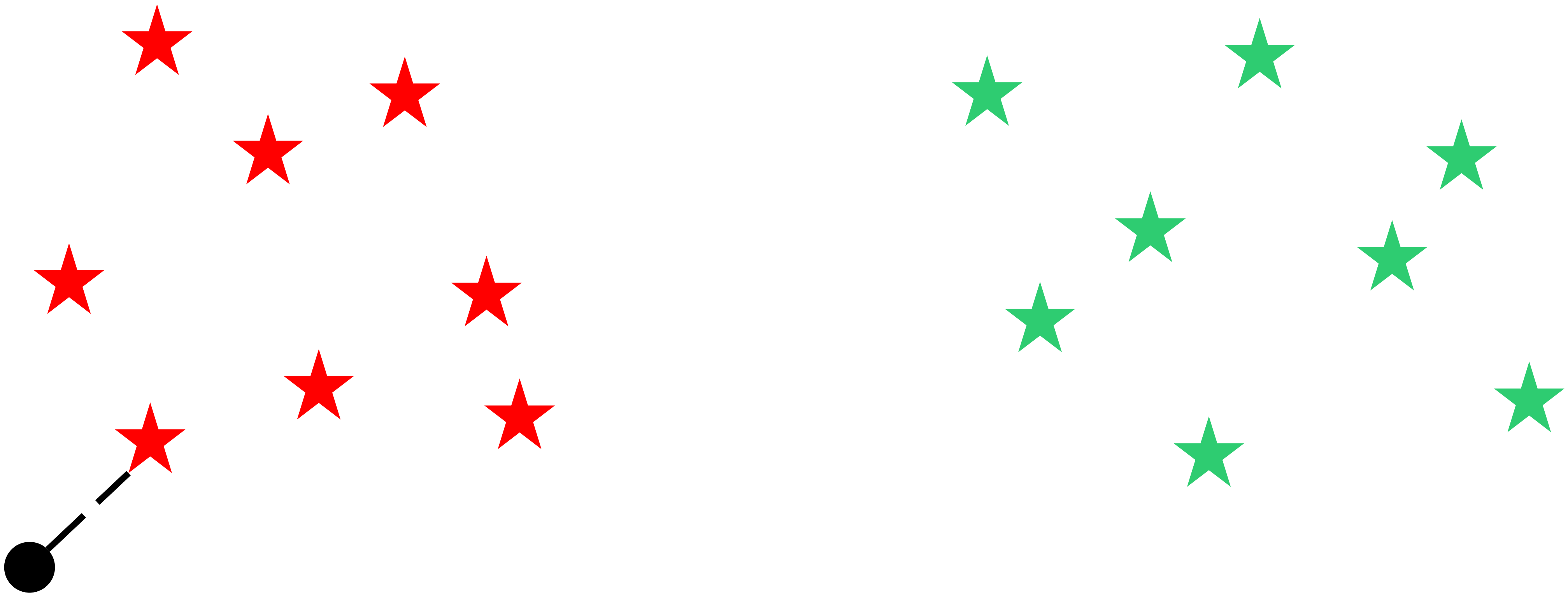
Learning with the 1-NN Classifier

Learning with the 1-NN Classifier

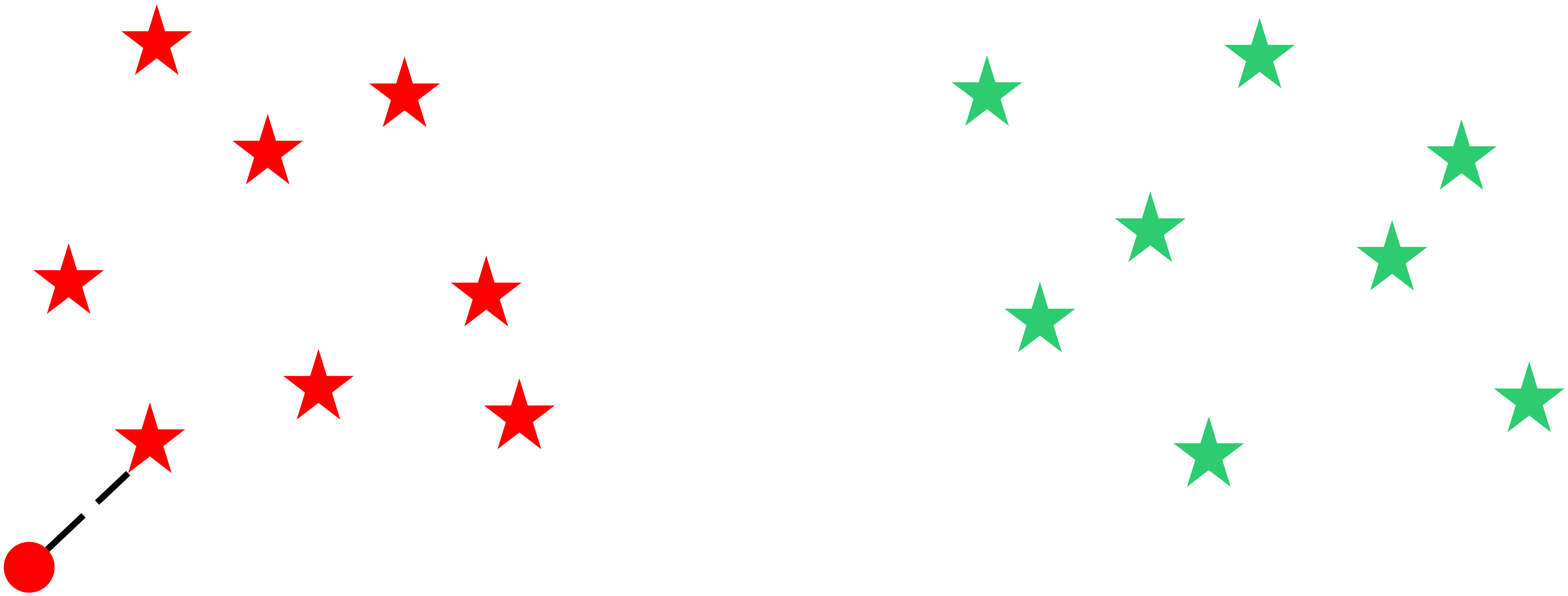
Learning with the 1-NN Classifier



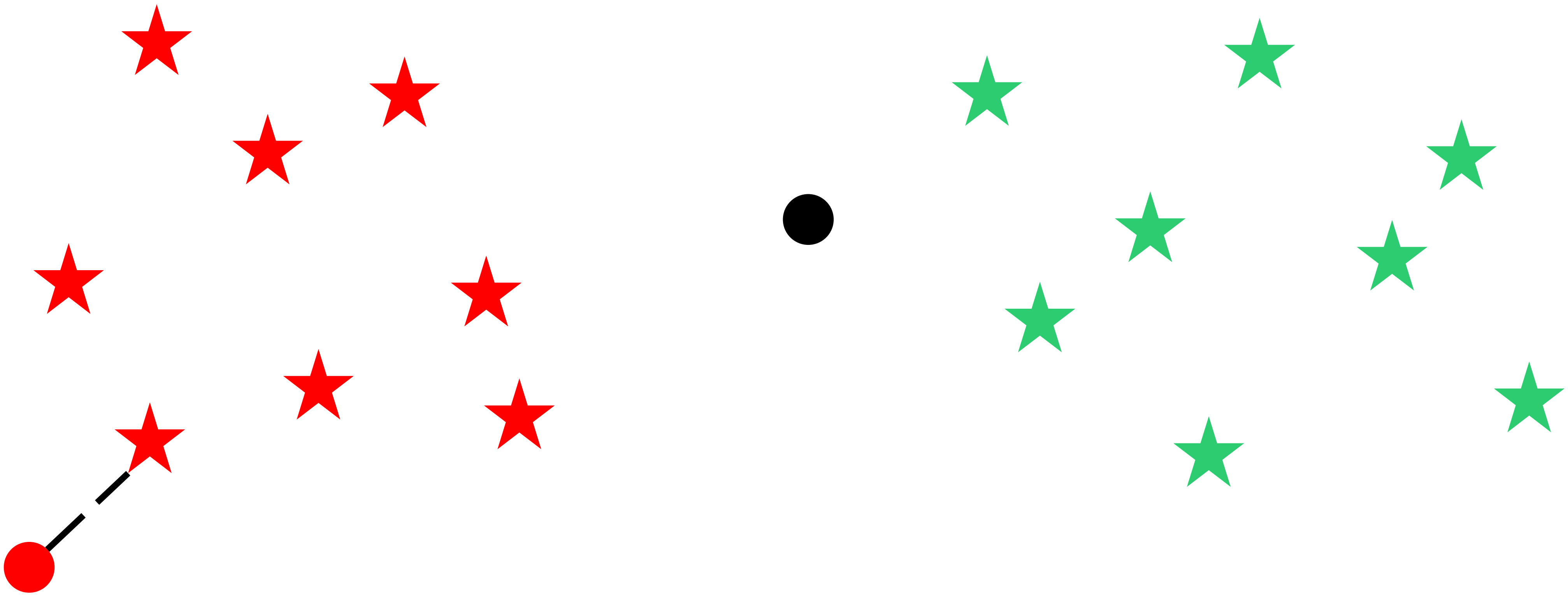
Learning with the 1-NN Classifier



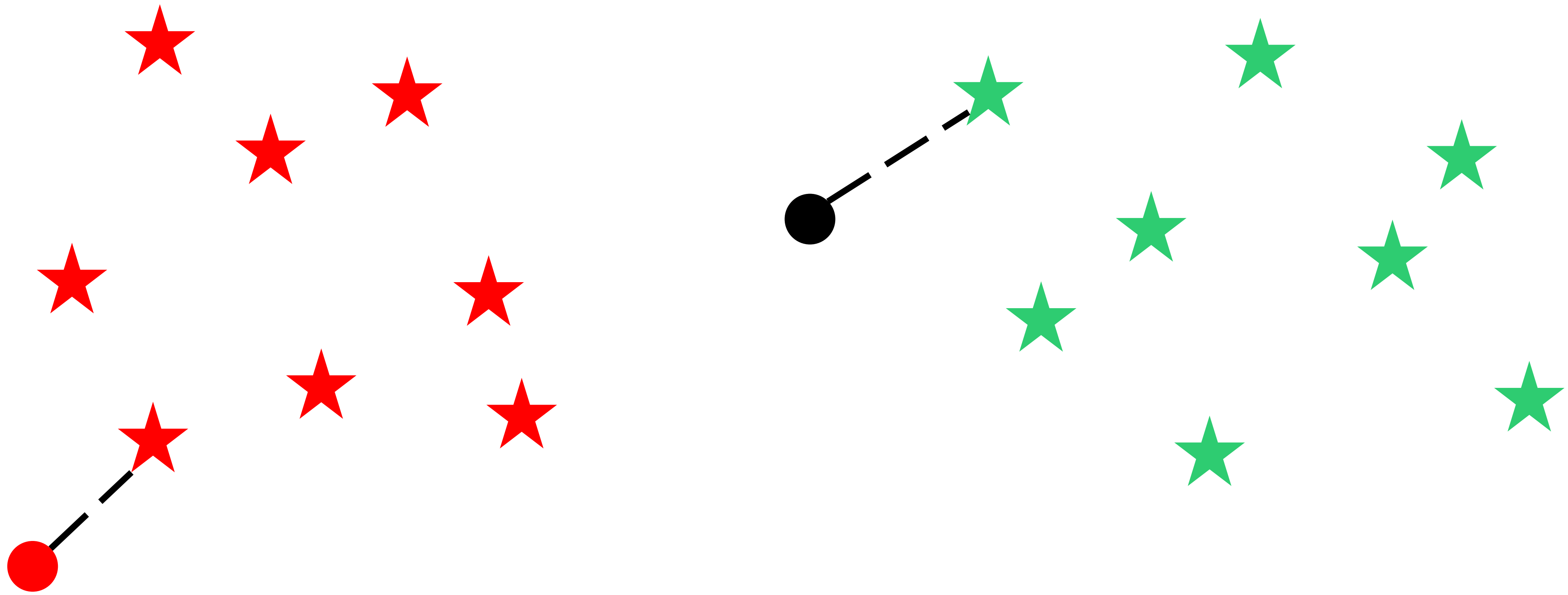
Learning with the 1-NN Classifier



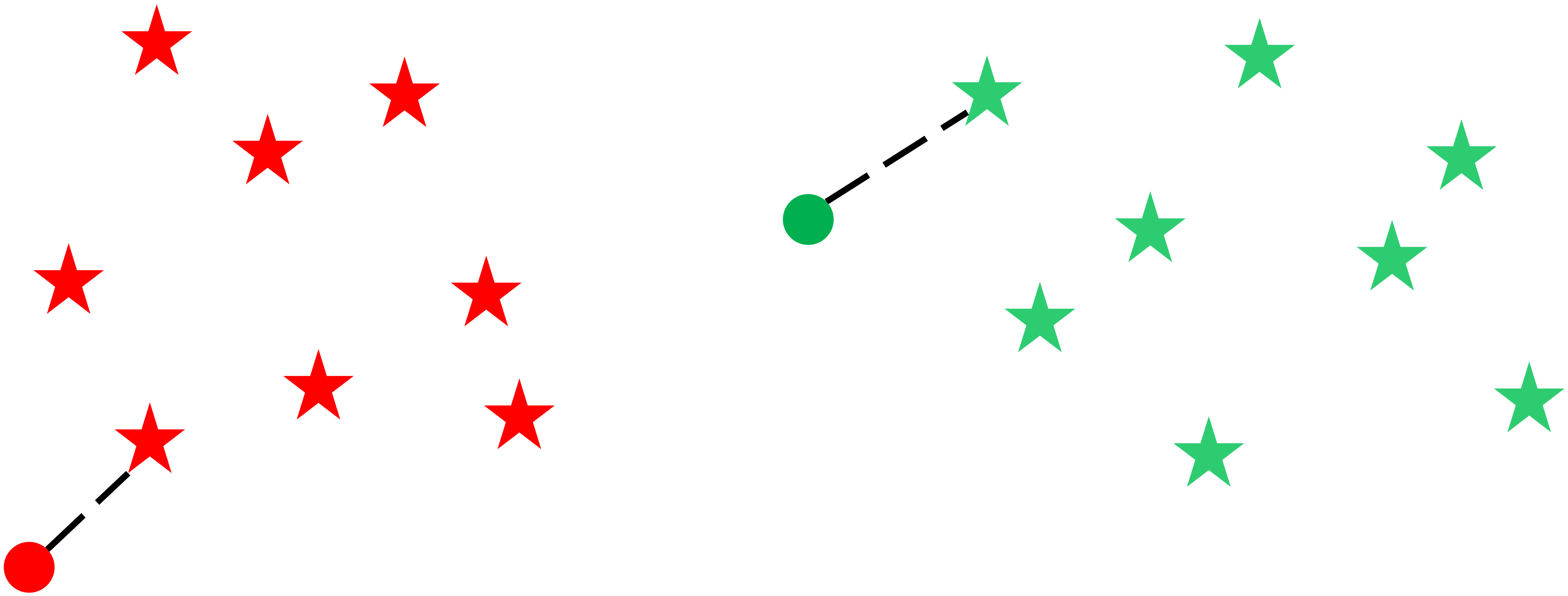
Learning with the 1-NN Classifier



Learning with the 1-NN Classifier

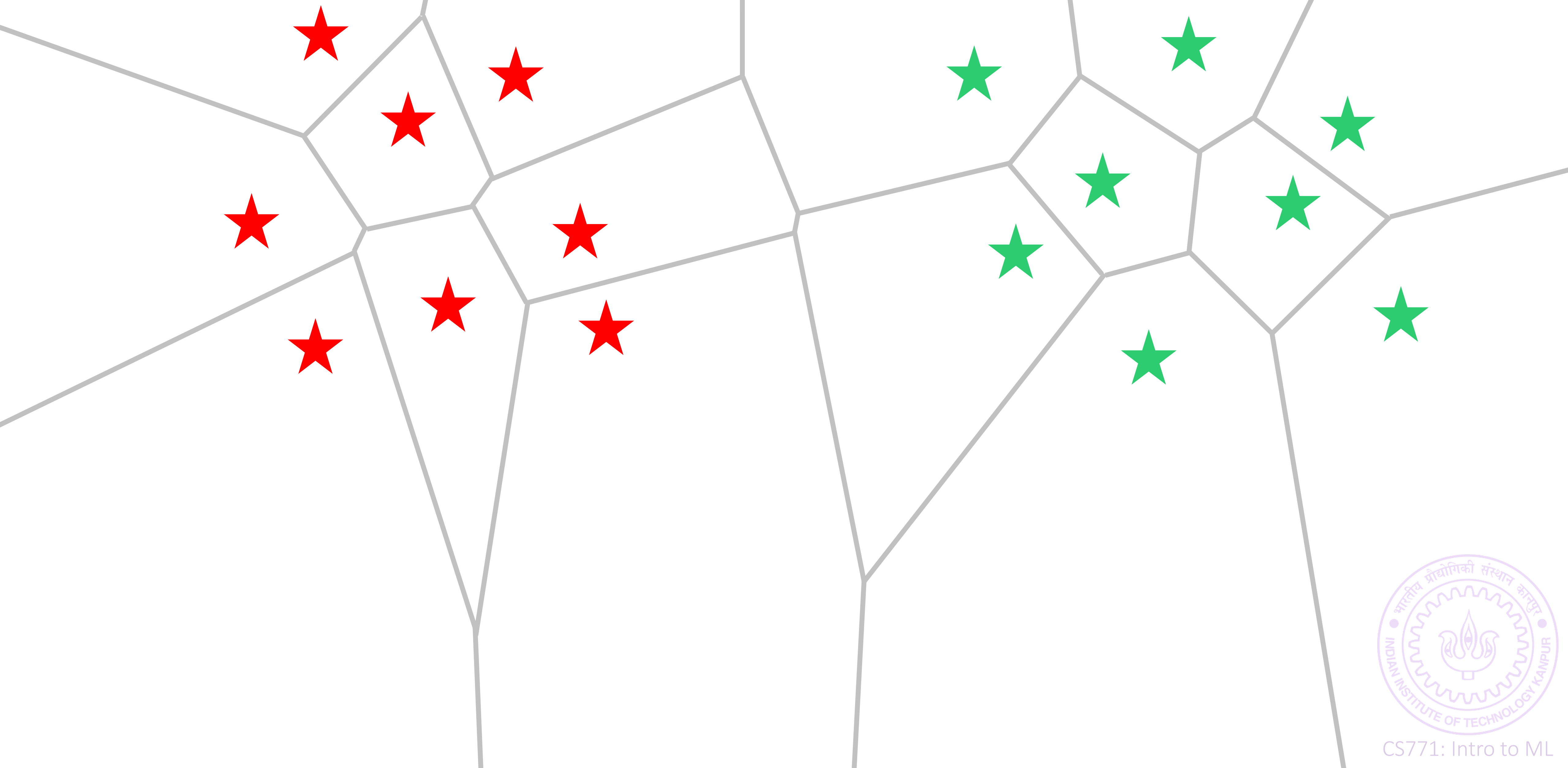


Learning with the 1-NN Classifier

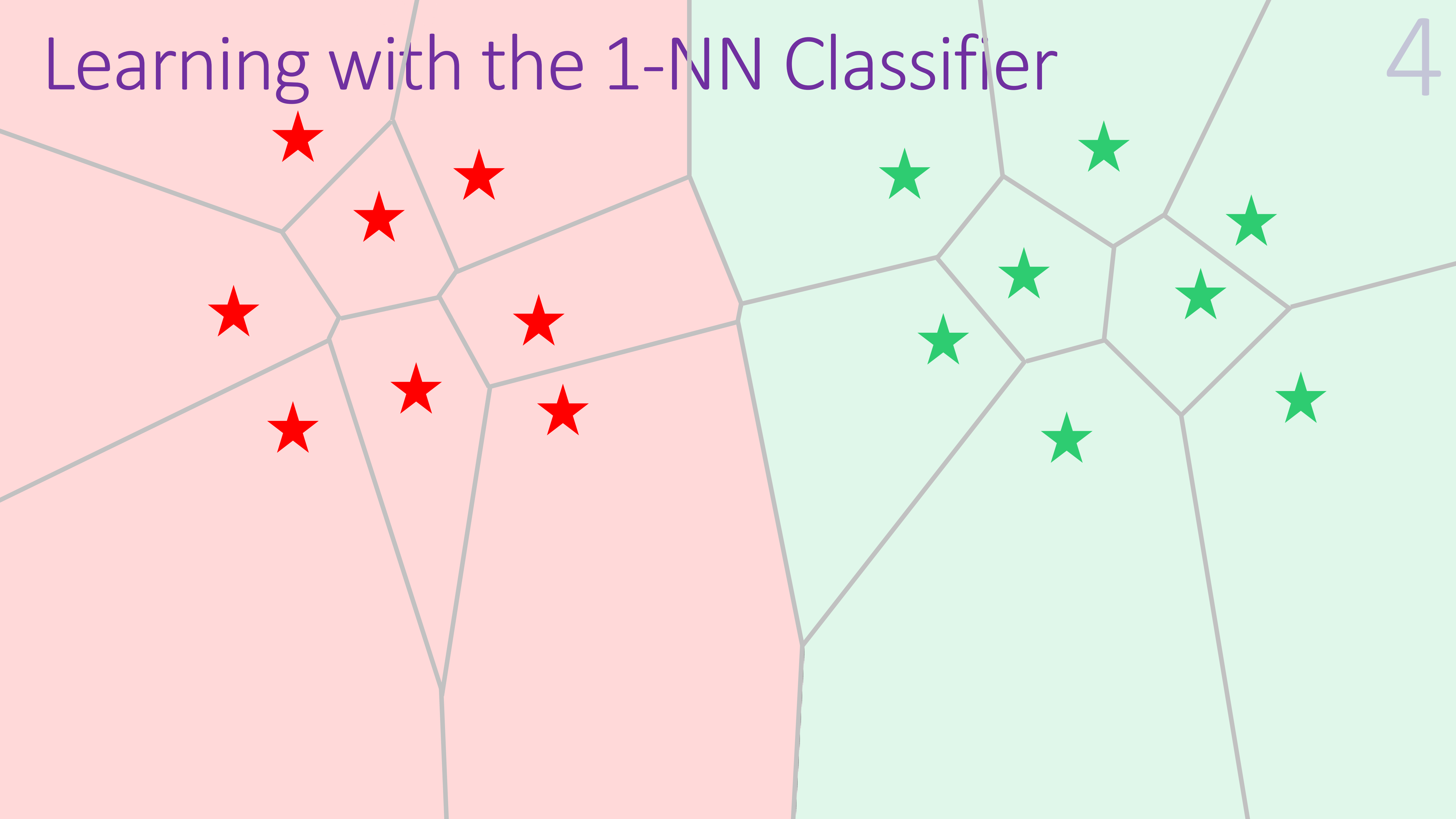


Learning with the 1-NN Classifier

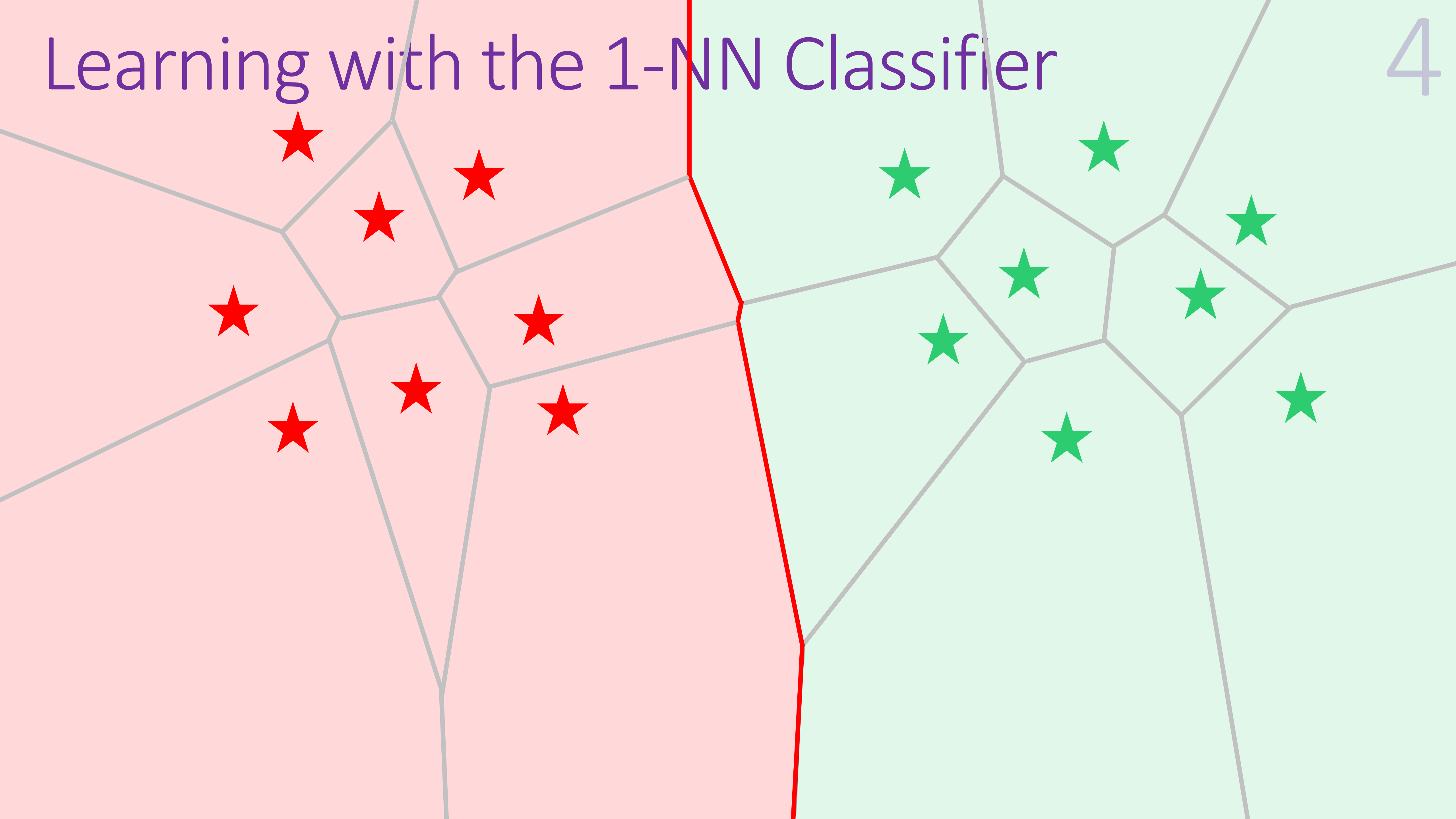
Learning with the 1-NN Classifier



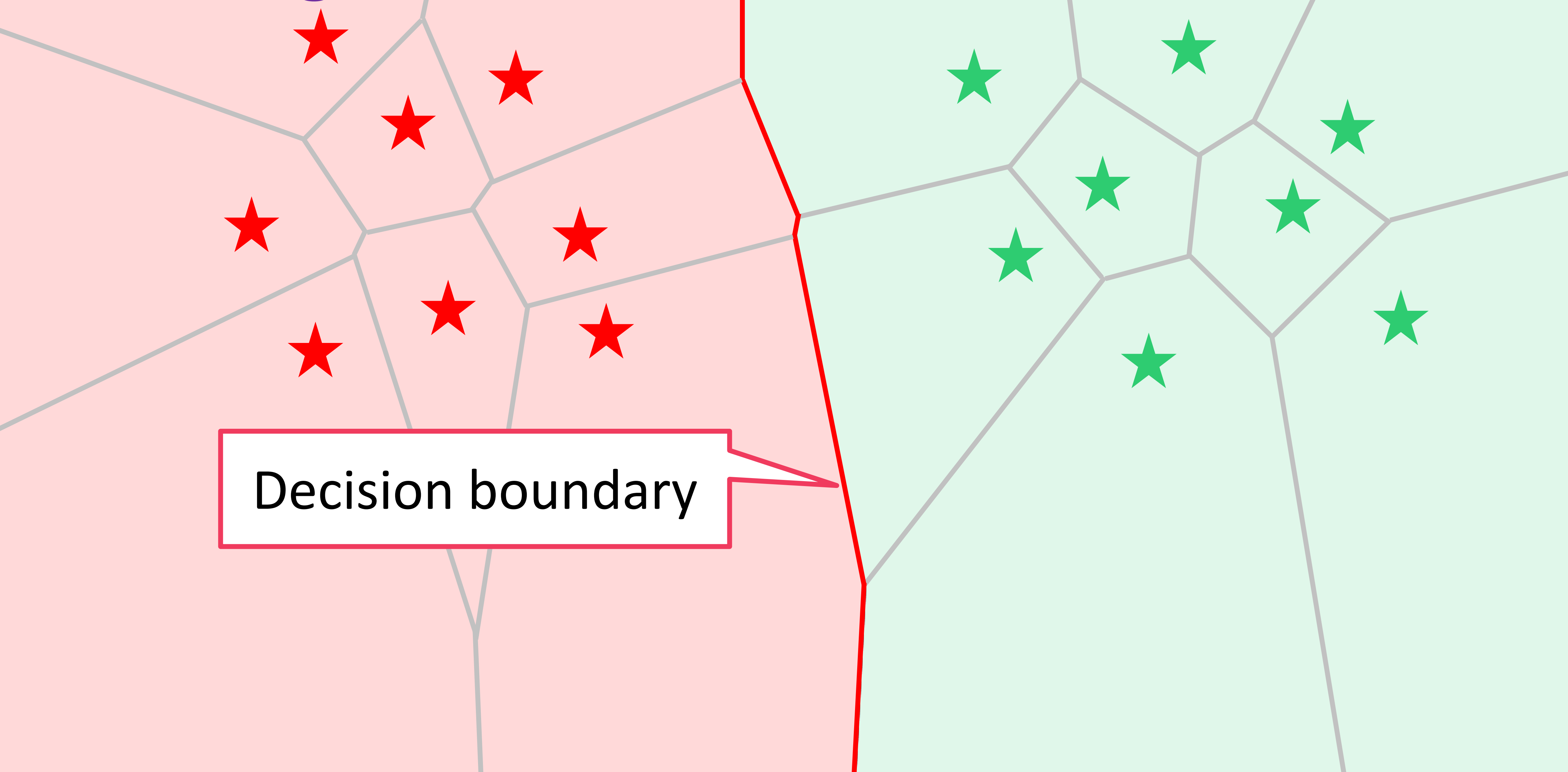
Learning with the 1-NN Classifier



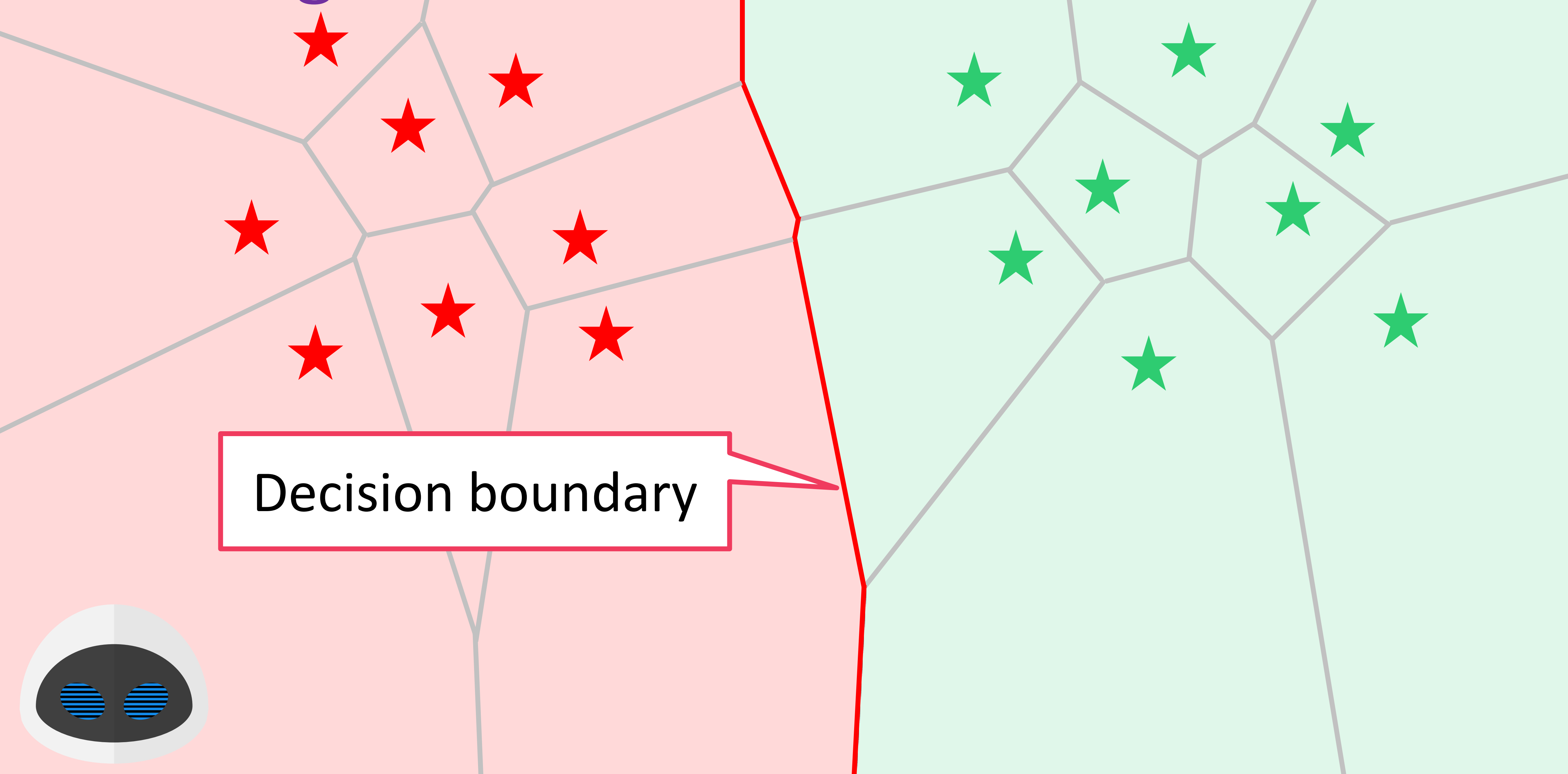
Learning with the 1-NN Classifier



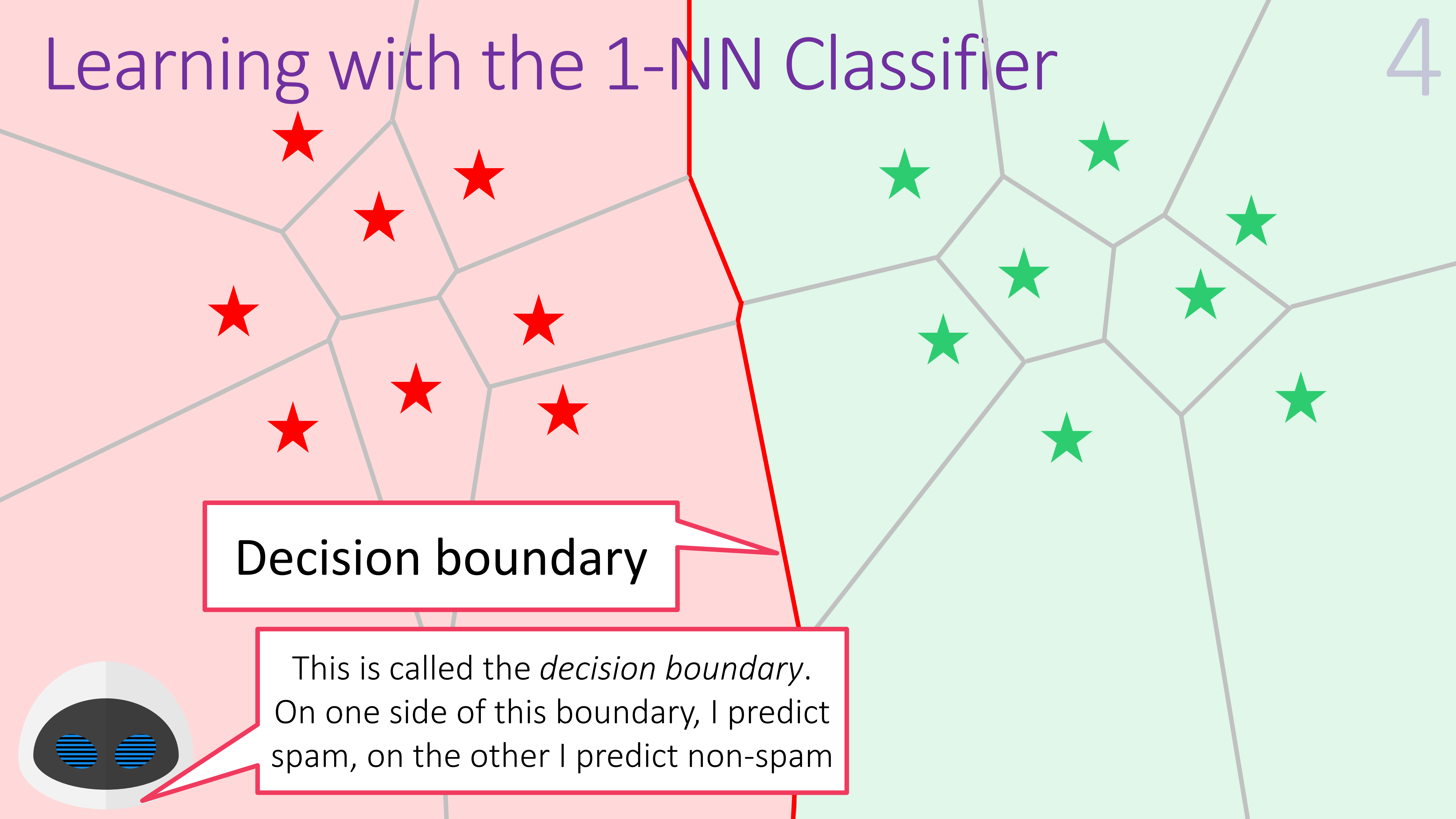
Learning with the 1-NN Classifier



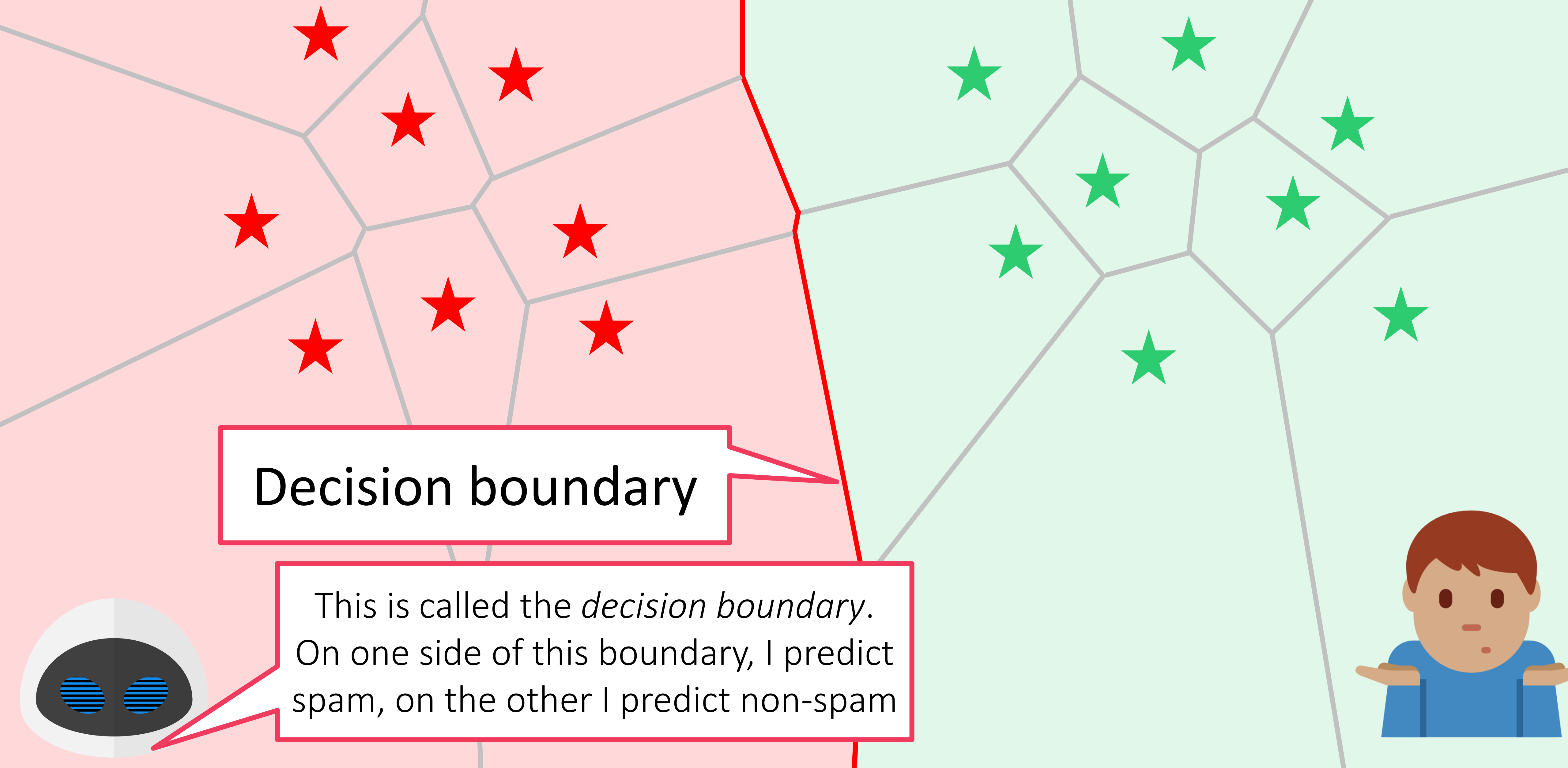
Learning with the 1-NN Classifier



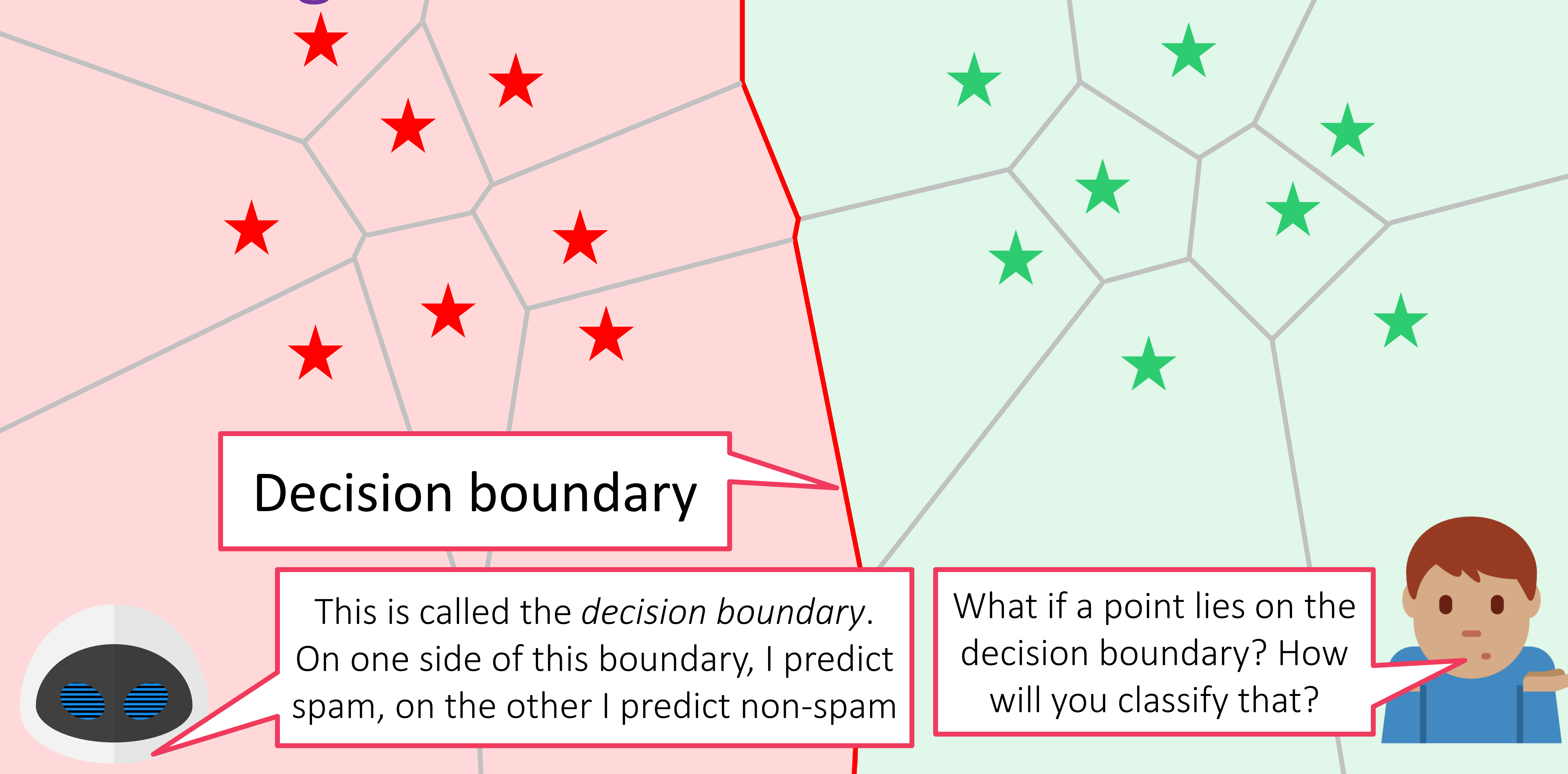
Learning with the 1-NN Classifier



Learning with the 1-NN Classifier



Learning with the 1-NN Classifier

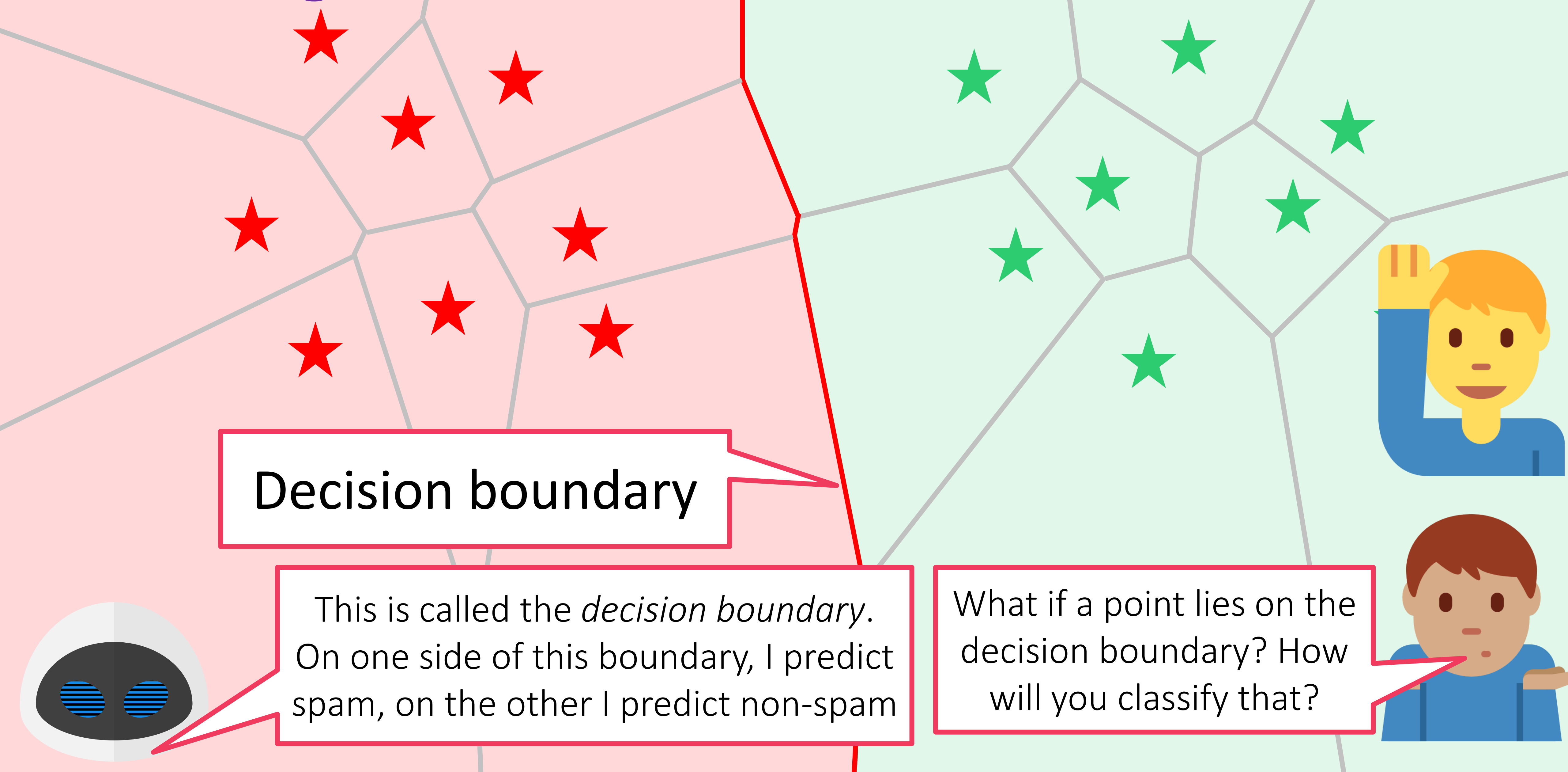


Decision boundary

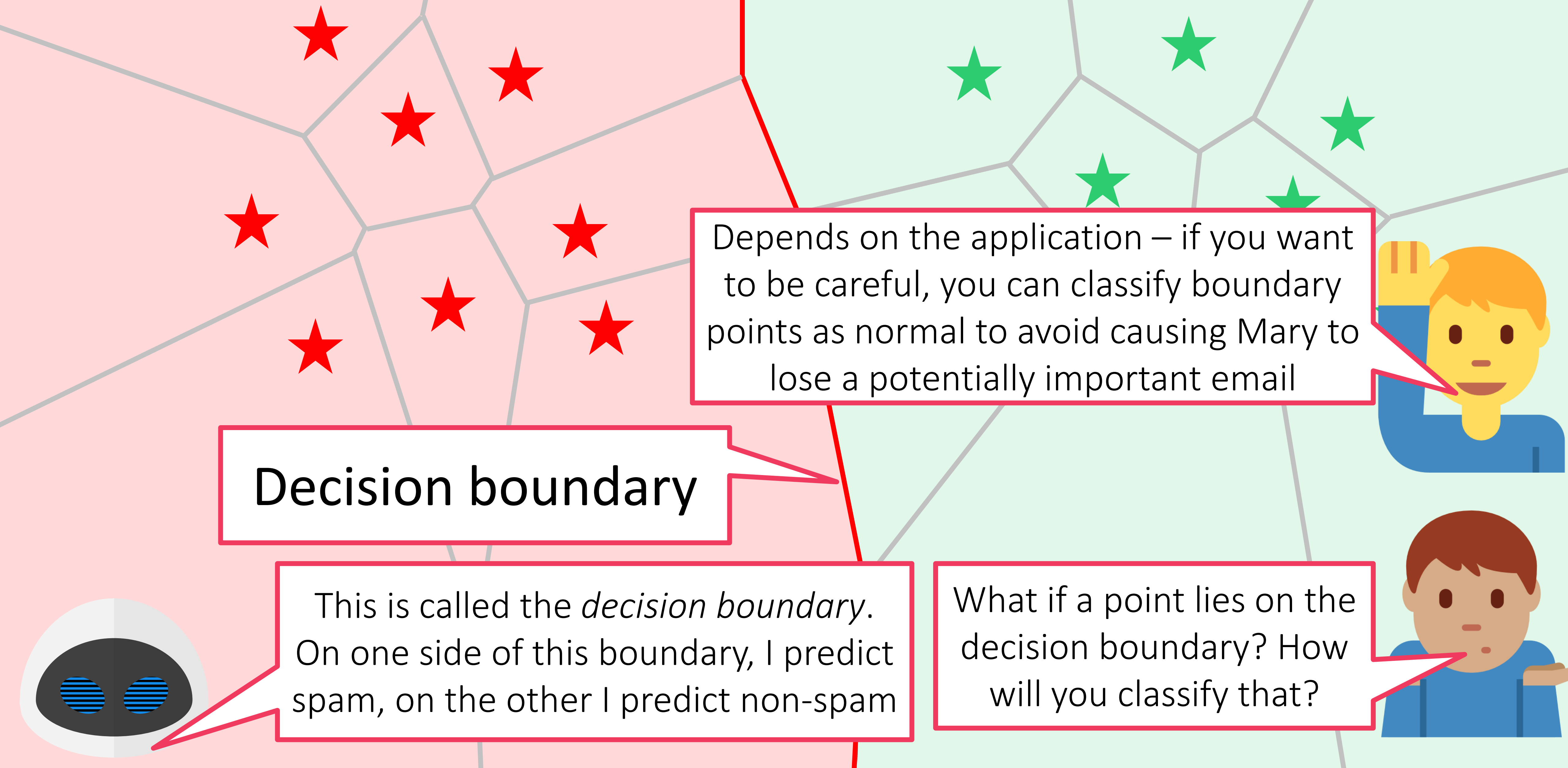
This is called the *decision boundary*. On one side of this boundary, I predict spam, on the other I predict non-spam

What if a point lies on the decision boundary? How will you classify that?

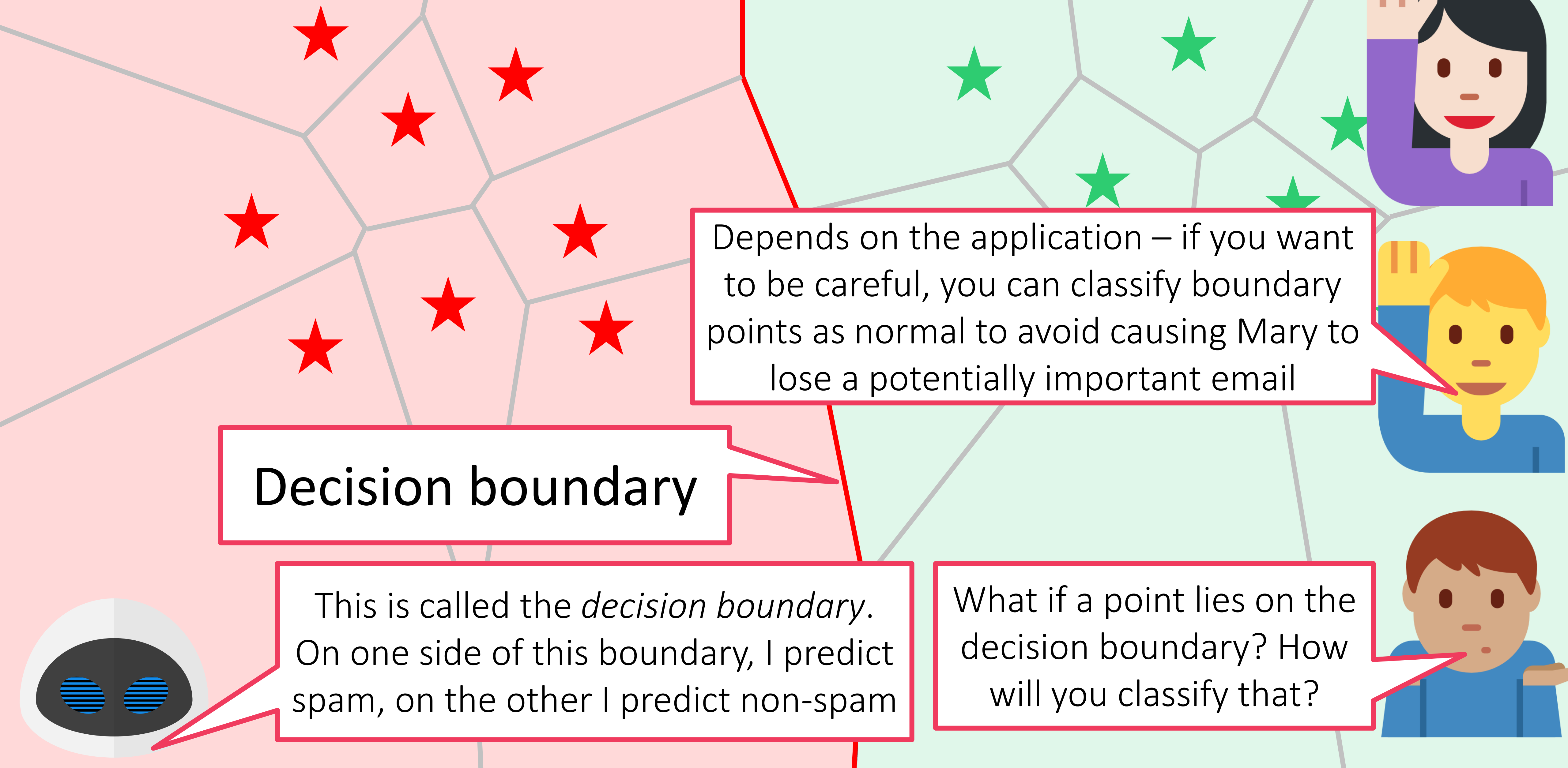
Learning with the 1-NN Classifier

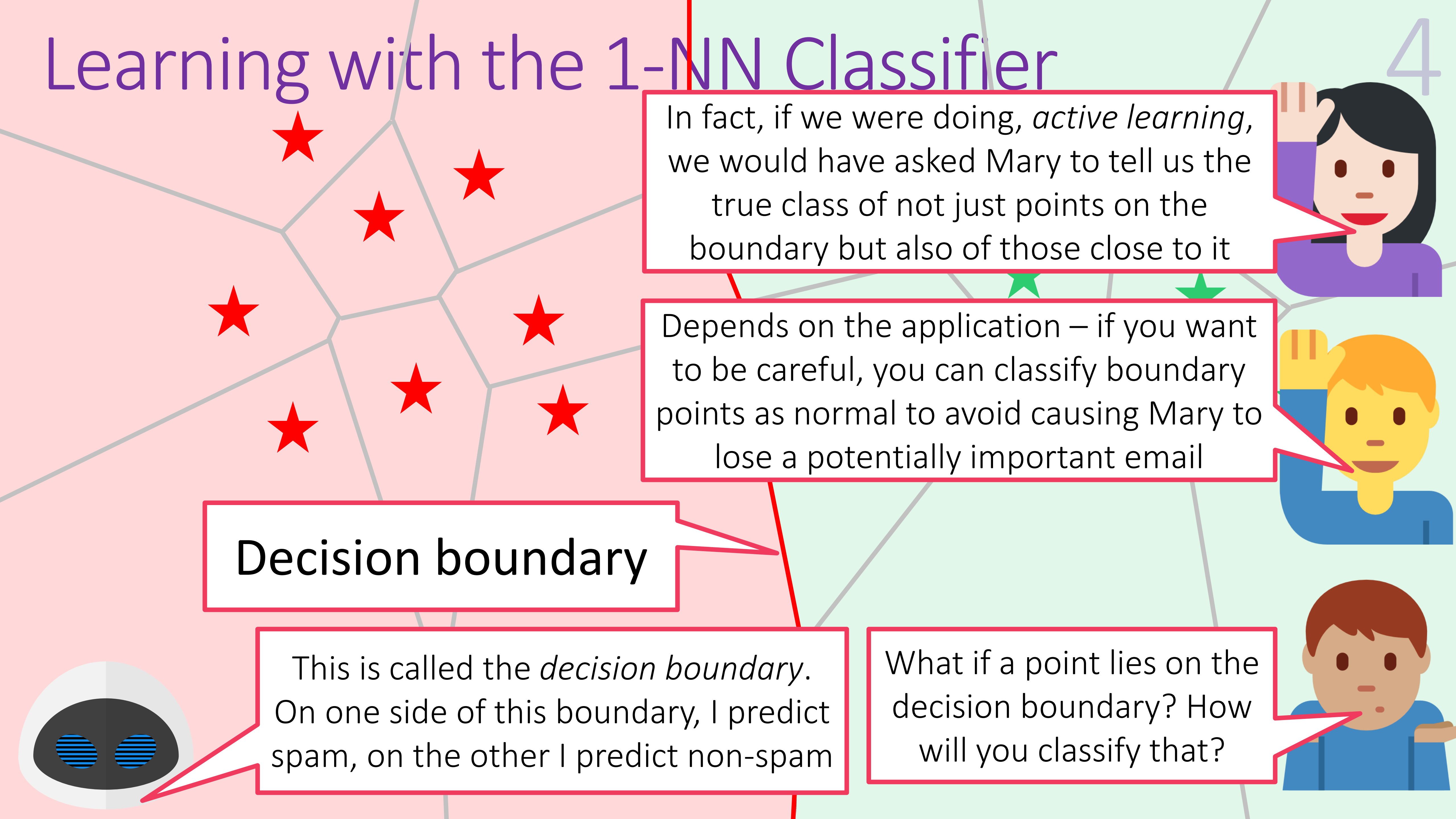


Learning with the 1-NN Classifier



Learning with the 1-NN Classifier




Learning with the 1-NN Classifier
4
In fact, if we were doing, *active learning*, we would have asked Mary to tell us the true class of not just points on the boundary but also of those close to it
Depends on the application – if you want to be careful, you can classify boundary points as normal to avoid causing Mary to lose a potentially important email
Decision boundary
This is called the *decision boundary*. On one side of this boundary, I predict spam, on the other I predict non-spam
What if a point lies on the decision boundary? How will you classify that?

Learning with the 1-NN Classifier

In fact, if we were doing, *active learning*, we would have asked Mary to tell us the true class of not just points on the boundary but also of those close to it

Depends on the application – if you want to be careful, you can classify boundary points as normal to avoid causing Mary to lose a potentially important email

Decision boundary

What is the decision boundary of LwP classifier?

This is called the *decision boundary*. On one side of this boundary, I predict spam, on the other I predict non-spam

What if a point lies on the decision boundary? How will you classify that?

LwP – behind the scenes

Let $\boldsymbol{\mu}^+, \boldsymbol{\mu}^-$ be the prototypes of the spam, non-spam classes resp.

Recall that we classify an email with feature vector $\mathbf{x}$ as spam if

$$\|\mathbf{x} - \boldsymbol{\mu}^+\|_2 < \|\mathbf{x} - \boldsymbol{\mu}^-\|_2$$

$$\Leftrightarrow \|\mathbf{x} - \boldsymbol{\mu}^+\|_2^2 < \|\mathbf{x} - \boldsymbol{\mu}^-\|_2^2$$

$$\Leftrightarrow \|\mathbf{x}\|_2^2 + \|\boldsymbol{\mu}^+\|_2^2 - 2\langle \mathbf{x}, \boldsymbol{\mu}^+ \rangle < \|\mathbf{x}\|_2^2 + \|\boldsymbol{\mu}^-\|_2^2 - 2\langle \mathbf{x}, \boldsymbol{\mu}^- \rangle$$

$$\Leftrightarrow \|\boldsymbol{\mu}^+\|_2^2 - 2\langle \mathbf{x}, \boldsymbol{\mu}^+ \rangle < \|\boldsymbol{\mu}^-\|_2^2 - 2\langle \mathbf{x}, \boldsymbol{\mu}^- \rangle$$

$$\Leftrightarrow \langle \mathbf{x}, 2(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-) \rangle + \|\boldsymbol{\mu}^-\|_2^2 - \|\boldsymbol{\mu}^+\|_2^2 > 0$$

$$\equiv \ \langle \mathbf{x}, \mathbf{w} \rangle + b > 0 \text{ with } \mathbf{w} = 2(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-), \mathrm{b} = \|\boldsymbol{\mu}^-\|_2^2 - \|\boldsymbol{\mu}^+\|_2^2$$

LwP – behind the scenes

Let $\boldsymbol{\mu}^+, \boldsymbol{\mu}^-$ be the prototypes of the spa

Recall that we classify an email with feat

$$\|\mathbf{x} - \boldsymbol{\mu}^+\|_2 < \|\mathbf{x} - \boldsymbol{\mu}^-\|_2$$

$$\Leftrightarrow \|\mathbf{x} - \boldsymbol{\mu}^+\|_2^2 < \|\mathbf{x} - \boldsymbol{\mu}^-\|_2^2$$

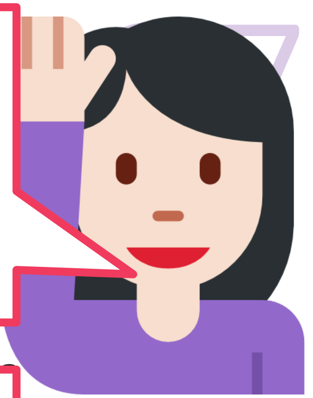
$$\Leftrightarrow \|\mathbf{x}\|_2^2 + \|\boldsymbol{\mu}^+\|_2^2 - 2\langle \mathbf{x}, \boldsymbol{\mu}^+ \rangle < \|\mathbf{x}\|_2^2 +$$

$$\Leftrightarrow \|\boldsymbol{\mu}^+\|_2^2 - 2\langle \mathbf{x}, \boldsymbol{\mu}^+ \rangle < \|\boldsymbol{\mu}^-\|_2^2 - 2\langle \mathbf{x}, \boldsymbol{\mu}^-\rangle$$

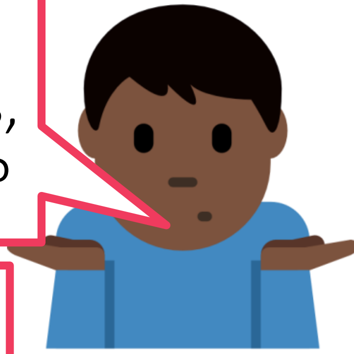
$$\Leftrightarrow \langle \mathbf{x}, 2(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-)\rangle + \|\boldsymbol{\mu}^-\|_2^2 - \|\boldsymbol{\mu}^+\|_2^2 >$$

$$\equiv \langle \mathbf{x}, \mathbf{w} \rangle + b > 0 \text{ with } \mathbf{w} = 2($$

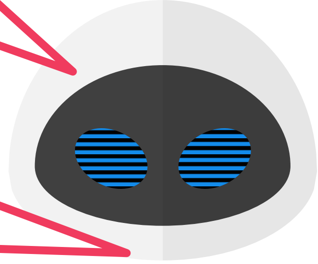
Classifiers with linear decision boundaries are called *linear classifiers*. Thus, LwP is a linear classifier



What happens if there are 2 or more prototypes per class, or else if there are 3 classes?

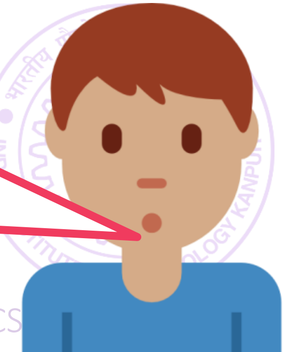


Think about this on your own – will discuss later



Yes, this is known as a *linear decision boundary*.

So the decision boundary of the LwP classifier is always a line or a hyperplane if the distance function is Euclidean!!



Linear/hyperplane Classifiers

The model is a single vector $\mathbf{w}$ of dimension d (features are also d-dim), and an optional scalar term (called *bias*) b

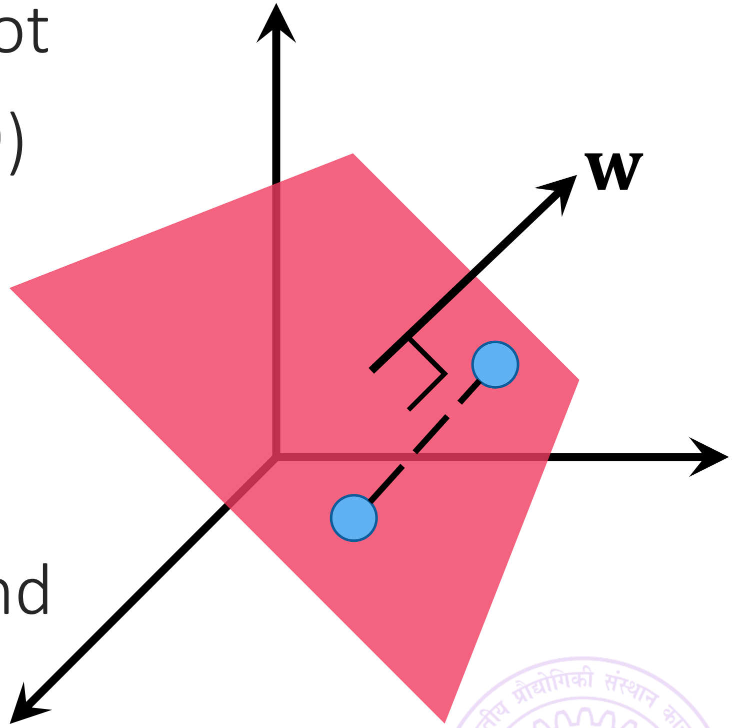
Predict on a test point $\mathbf{x}$ by checking if $\mathbf{w}^\top \mathbf{x} + b > 0$ or not

Decision boundary: line/hyperplane (where $\mathbf{w}^\top \mathbf{x} + b = 0$)

The vector w is called the *normal* or *perpendicular* vector of the hyperplane – why?

Consider any two vectors $\mathbf{x}, \mathbf{y}$ on the hyperplane i.e. $\mathbf{w}^\top \mathbf{x} + b = 0 = \mathbf{w}^\top \mathbf{y} + b$. This means $\mathbf{w}^\top (\mathbf{x} - \mathbf{y}) = 0$. Note that the vector $\mathbf{x} - \mathbf{y}$ is parallel to the hyperplane and $\mathbf{w}$ perpendicular to all such vectors

The bias term b if changed, shifts the plane – it can be thought of as a threshold as well – how large does $\mathbf{w}^\top \mathbf{x}$ have to be in order for us to classify $\mathbf{x}$ as spam etc!



To b or not to b – that is the question!

Trivia: the closest point (Euclidean distance) on the hyperplane to the origin is at a distance $|b|/\|\mathbf{w}\|_2$ from the origin – can you show why?

Sometimes, it is convenient to not have a separate bias term

Create another dim in feature vector and fill it with 1 i.e. $\tilde{\mathbf{x}} = [\mathbf{x}, 1]$

So now features (and model) are $d + 1$-dimensional

However, note that if we have a model $\tilde{\mathbf{w}} = [w_0, w_1, \ldots, w_d] \in \mathbb{R}^{d+1}$ over the new features and if we denote $w = [w_0, \ldots w_{d-1}] \in \mathbb{R}^d$, then

$$\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}} = \mathbf{w}^\top \mathbf{x} + w_d$$

Thus, w_d effectively acts as a bias term for us ☺

Making LwP more Powerful

The Euclidean distance is nice but gives all features equal weight

Also does not allow features to talk to each other

E.g. $\|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^{d} \mathbf{x}_j^2}$ has no $\mathbf{x}_j \mathbf{x}_k$ term for $j \neq k$

Using a different distance function really helps

Metric learning: learn this distance function as well

A very popular family of metrics – Mahalanobis metrics

Given a symmetric $d \times d$ matrix $A \in \mathbb{R}^{d \times d}$, we define a distance

$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top A (\mathbf{x} - \mathbf{y})}$$

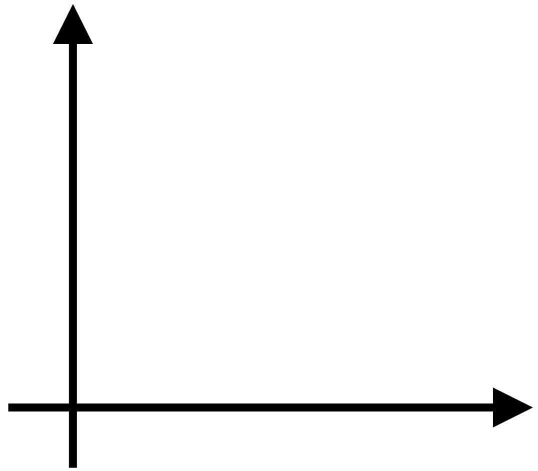
Taking $A = I_d$ i.e. identity matrix, gives us the usual Euclidean distance

Making LwP more Powerful

The Euclidean distance is nice but gives all features equal weight

Also does not allow features to talk to each other

E.g. $\|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^{d} \mathbf{x}_j^2}$ has no $\mathbf{x}_j \mathbf{x}_k$ term for $j \neq k$



Using a different distance function really helps

Metric learning: learn this distance function as well

A very popular family of metrics – Mahalanobis metrics

Given a symmetric $d \times d$ matrix $A \in \mathbb{R}^{d \times d}$, we define a distance

$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top A (\mathbf{x} - \mathbf{y})}$$

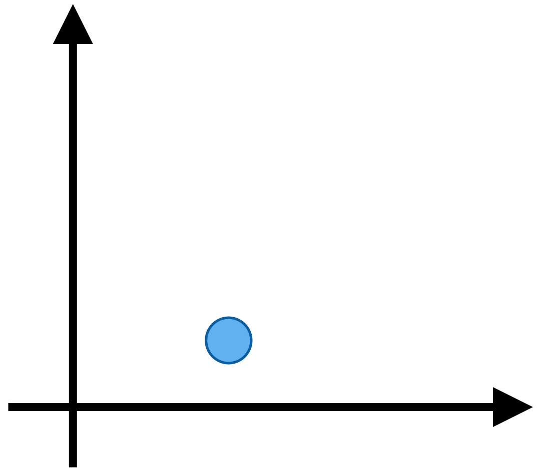
Taking $A = I_d$ i.e. identity matrix, gives us the usual Euclidean distance

Making LwP more Powerful

The Euclidean distance is nice but gives all features equal weight

Also does not allow features to talk to each other

E.g. $\|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^{d} \mathbf{x}_j^2}$ has no $\mathbf{x}_j \mathbf{x}_k$ term for $j \neq k$



Using a different distance function really helps

Metric learning: learn this distance function as well

A very popular family of metrics – Mahalanobis metrics

Given a symmetric $d \times d$ matrix $A \in \mathbb{R}^{d \times d}$, we define a distance

$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top A (\mathbf{x} - \mathbf{y})}$$

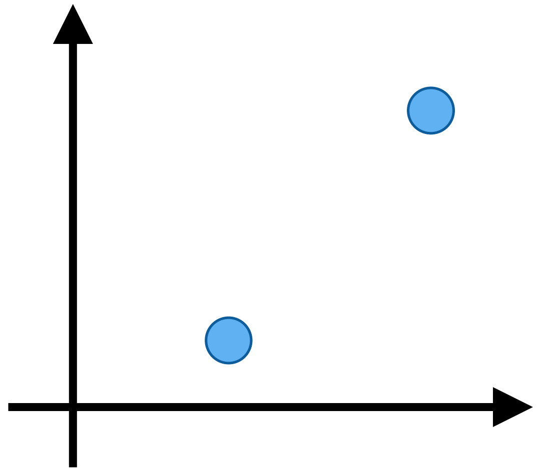
Taking $A = I_d$ i.e. identity matrix, gives us the usual Euclidean distance

Making LwP more Powerful

The Euclidean distance is nice but gives all features equal weight

Also does not allow features to talk to each other

E.g. $\|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^{d} \mathbf{x}_j^2}$ has no $\mathbf{x}_j \mathbf{x}_k$ term for $j \neq k$



Using a different distance function really helps

Metric learning: learn this distance function as well

A very popular family of metrics – Mahalanobis metrics

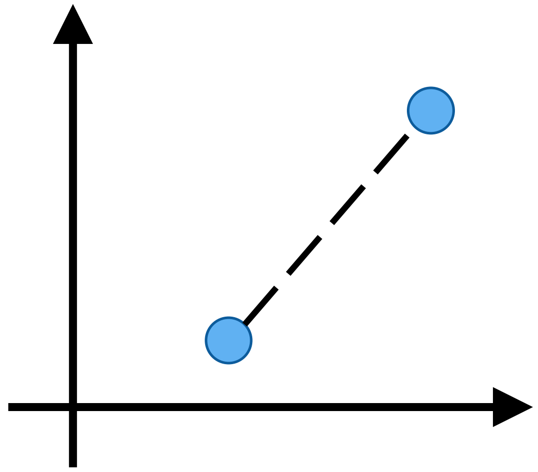
Given a symmetric $d \times d$ matrix $A \in \mathbb{R}^{d \times d}$, we define a distance

$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top A (\mathbf{x} - \mathbf{y})}$$

Taking $A = I_d$ i.e. identity matrix, gives us the usual Euclidean distance

Making LwP more Powerful

The Euclidean distance is nice but gives all features equal weight



Also does not allow features to talk to each other

E.g. $\|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^{d} \mathbf{x}_j^2}$ has no $\mathbf{x}_j \mathbf{x}_k$ term for $j \neq k$

Using a different distance function really helps

Metric learning: learn this distance function as well

A very popular family of metrics – Mahalanobis metrics

Given a symmetric $d \times d$ matrix $A \in \mathbb{R}^{d \times d}$, we define a distance

$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top A (\mathbf{x} - \mathbf{y})}$$

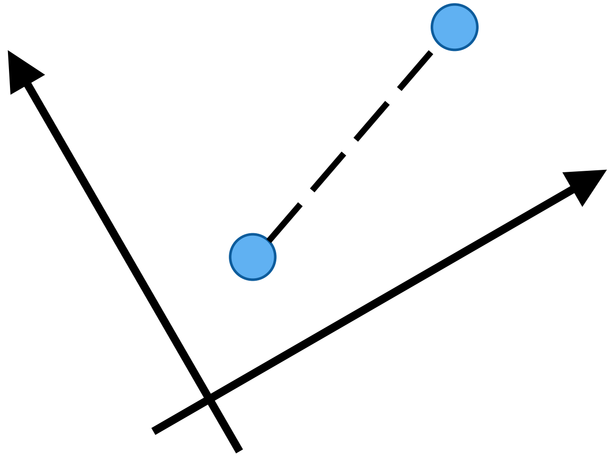
Taking $A = I_d$ i.e. identity matrix, gives us the usual Euclidean distance

Making LwP more Powerful

The Euclidean distance is nice but gives all features equal weight

Also does not allow features to talk to each other

E.g. $\|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^{d} \mathbf{x}_j^2}$ has no $\mathbf{x}_j \mathbf{x}_k$ term for $j \neq k$



Using a different distance function really helps

Metric learning: learn this distance function as well

A very popular family of metrics – Mahalanobis metrics

Given a symmetric $d \times d$ matrix $A \in \mathbb{R}^{d \times d}$, we define a distance

$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top A (\mathbf{x} - \mathbf{y})}$$

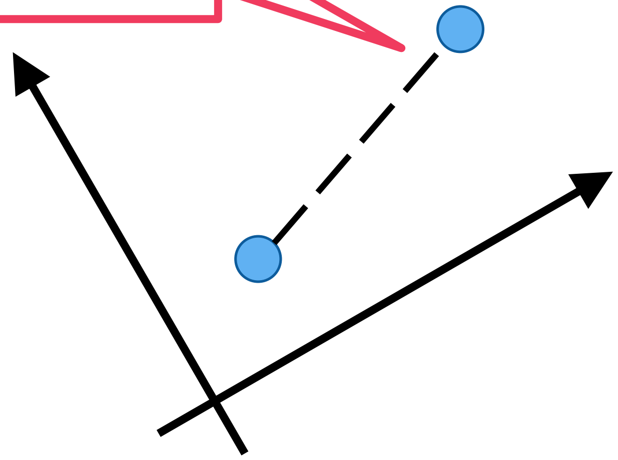
Taking $A = I_d$ i.e. identity matrix, gives us the usual Euclidean distance

Making LwP more Powerful

The Euclidean distance is nice but … eight

Also does not allow features to tal…

Euclidean distance does not change even if axes are rotated



E.g. $\|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^{d} \mathbf{x}_j^2}$ has no $\mathbf{x}_j\mathbf{x}_k$ term for $j \neq k$

Using a different distance function really helps

Metric learning: learn this distance function as well

A very popular family of metrics – Mahalanobis metrics

Given a symmetric $d \times d$ matrix $A \in \mathbb{R}^{d\times d}$, we define a distance

$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^\top A(\mathbf{x}-\mathbf{y})}$$

Taking $A = I_d$ i.e. identity matrix, gives us the usual Euclidean distance

LwP with Mahalanobis metric still linear!!

A matrix A that satisfies a property called *positive semi-definiteness* (PSD has several other nice properties too

For all vectors $\mathbf{x}$, we must have $\mathbf{x}^\top A\mathbf{x} \geq 0$

$$d_A(\mathbf{x}, \boldsymbol{\mu}^+) < d_A(\mathbf{x}, \boldsymbol{\mu}^-) \Leftrightarrow 2\mathbf{x}^\top A(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-) + {\boldsymbol{\mu}^-}^\top A\boldsymbol{\mu}^- - {\boldsymbol{\mu}^+}^\top A\boldsymbol{\mu}^+ > 0$$

$$\equiv \langle \mathbf{x}, \mathbf{w} \rangle + b > 0 \text{ where } \mathbf{w} = 2A(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-), b = {\boldsymbol{\mu}^-}^\top A\boldsymbol{\mu}^- - {\boldsymbol{\mu}^+}^\top A\boldsymbol{\mu}^+$$

We can write $A = LL^\top$ where $L \in \mathbb{R}^{d\times d}$ (L need not be sym or PSD)

$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^\top A(\mathbf{x}-\mathbf{y})} = \sqrt{(\mathbf{x}-\mathbf{y})^\top LL^\top(\mathbf{x}-\mathbf{y})}$$

$$= \|L^\top\mathbf{x} - L^\top\mathbf{y}\|_2$$

LwP with Mahalanobis metric still linear!!

A matrix A that satisfies a property called *positive semi-definiteness* (PSD has several other nice properties too

For all vectors $\mathbf{x}$, we must have $\mathbf{x}^\top A\mathbf{x} \geq 0$

$$d_A(\mathbf{x}, \boldsymbol{\mu}^+) < d_A(\mathbf{x}, \boldsymbol{\mu}^-) \Leftrightarrow 2\mathbf{x}^\top A(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-) + {\boldsymbol{\mu}^-}^\top A\boldsymbol{\mu}^- - {\boldsymbol{\mu}^+}^\top A\boldsymbol{\mu}^+ > 0$$

$$\equiv \langle \mathbf{x}, \mathbf{w} \rangle + b > 0 \text{ where } \mathbf{w} = 2A(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-), b = {\boldsymbol{\mu}^-}^\top A\boldsymbol{\mu}^- - {\boldsymbol{\mu}^+}^\top A\boldsymbol{\mu}^+$$

We can write $A = LL^\top$ where $L \in \mathbb{R}^{d\times d}$ (L need not be sym or PSD)

$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^\top A(\mathbf{x}-\mathbf{y})} = \sqrt{(\mathbf{x}-\mathbf{y})^\top LL^\top(\mathbf{x}-\mathbf{y})}$$

$$= \|L^\top\mathbf{x} - L^\top\mathbf{y}\|_2$$

LwP with Mahalanobis metric still linear!!

A matrix A that satisfies a property called *positive semi-definiteness* (PSD has several other nice properties too

For all vectors $\mathbf{x}$, we must have $\mathbf{x}^\top A\mathbf{x} \geq 0$

$$d_A(\mathbf{x}, \boldsymbol{\mu}^+) < d_A(\mathbf{x}, \boldsymbol{\mu}^-) \Leftrightarrow 2\mathbf{x}^\top A(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-) + {\boldsymbol{\mu}^-}^\top A\boldsymbol{\mu}^- - {\boldsymbol{\mu}^+}^\top A\boldsymbol{\mu}^+ > 0$$

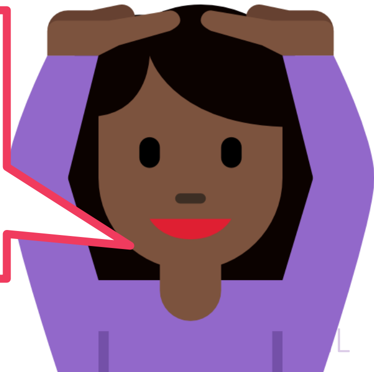
$$\equiv \langle \mathbf{x}, \mathbf{w} \rangle + b > 0 \text{ where } \mathbf{w} = 2A(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-), b = {\boldsymbol{\mu}^-}^\top A\boldsymbol{\mu}^- - {\boldsymbol{\mu}^+}^\top A\boldsymbol{\mu}^+$$

We can write $A = LL^\top$ where $L \in \mathbb{R}^{d \times d}$ (L need not be sym or PSD)

$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top A(\mathbf{x} - \mathbf{y})} = \sqrt{(\mathbf{x} - \mathbf{y})^\top LL^\top(\mathbf{x} - \mathbf{y})}$$

$$= \|L^\top \mathbf{x} - L^\top \mathbf{y}\|$$

Nice! This means that $d_A(\mathbf{x}, \mathbf{y}) \geq 0$ for all $\mathbf{x}, \mathbf{y}$ i.e. this will never give us negative distances which don't make sense



LwP with Mahalanobis metric still linear!!

A matrix A that satisfies a property called *positive semi-definiteness* (PSD has several other nice properties too

For all vectors $\mathbf{x}$, we must have $\mathbf{x}^\top A\mathbf{x} \geq 0$

$$d_A(\mathbf{x}, \boldsymbol{\mu}^+) < d_A(\mathbf{x}, \boldsymbol{\mu}^-) \Leftrightarrow 2\mathbf{x}^\top A(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-) + {\boldsymbol{\mu}^-}^\top A\boldsymbol{\mu}^- - {\boldsymbol{\mu}^+}^\top A\boldsymbol{\mu}^+ > 0$$

$$\equiv \langle \mathbf{x}, \mathbf{w} \rangle + b > 0 \text{ where } \mathbf{w} = 2A(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-), b = {\boldsymbol{\mu}^-}^\top A\boldsymbol{\mu}^- - {\boldsymbol{\mu}^+}^\top A\boldsymbol{\mu}^+$$

We can write $A = LL^\top$ where $L \in \mathbb{R}^{d\times d}$ (L need not be sym or PSD)

$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^\top A(\mathbf{x}-\mathbf{y})} = \sqrt{(\mathbf{x}-\mathbf{y})^\top LL^\top(\mathbf{x}-\mathbf{y})}$$

$$= \|L^\top\mathbf{x} - L^\top\mathbf{y}\|_2$$

LwP with Mahalanobis metric still linear!!

A matrix A that satisfies a property called *positive semi-definiteness* (PSD has several other nice properties too

For all vectors $\mathbf{x}$, we must have $\mathbf{x}^\top A\mathbf{x} \geq 0$

$$d_A(\mathbf{x}, \boldsymbol{\mu}^+) < d_A(\mathbf{x}, \boldsymbol{\mu}^-) \Leftrightarrow 2\mathbf{x}^\top A(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-) + {\boldsymbol{\mu}^-}^\top A\boldsymbol{\mu}^- - {\boldsymbol{\mu}^+}^\top A\boldsymbol{\mu}^+ > 0$$

$$\equiv \langle \mathbf{x}, \mathbf{w} \rangle + b > 0 \text{ where } \mathbf{w} = 2A(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-), b = {\boldsymbol{\mu}^-}^\top A\boldsymbol{\mu}^- - {\boldsymbol{\mu}^+}^\top A\boldsymbol{\mu}^+$$

We can write $A = LL^\top$ where $L \in \mathbb{R}^{d \times d}$ (L need not be sym or PSD)

$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top A(\mathbf{x} - \mathbf{y})} = \sqrt{(\mathbf{x} - \mathbf{y})^\top LL^\top(\mathbf{x} - \mathbf{y})}$$

$$= \|L^\top\mathbf{x} - L^\top\mathbf{y}\|_2$$

LwP with Mahalanobis metric still linear!!

A matrix A that satisfies a property called *positive semi-definiteness* (PSD has several other nice properties too

For all vectors $\mathbf{x}$, we must have $\mathbf{x}^\top A\mathbf{x} \geq 0$

$$d_A(\mathbf{x}, \boldsymbol{\mu}^+) < d_A(\mathbf{x}, \boldsymbol{\mu}^-) \Leftrightarrow 2\mathbf{x}^\top A(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-) + {\boldsymbol{\mu}^-}^\top A\boldsymbol{\mu}^- - {\boldsymbol{\mu}^+}^\top A\boldsymbol{\mu}^+ > 0$$

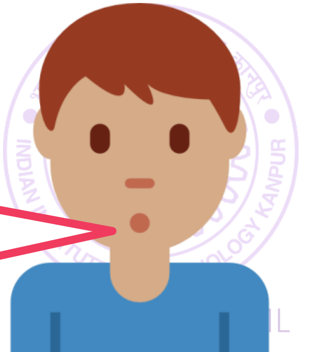
$$\equiv \langle \mathbf{x}, \mathbf{w} \rangle + b > 0 \text{ where } \mathbf{w} = 2A(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-), b = {\boldsymbol{\mu}^-}^\top A\boldsymbol{\mu}^- - {\boldsymbol{\mu}^+}^\top A\boldsymbol{\mu}^+$$

We can write $A = LL^\top$ where $L \in \mathbb{R}^{d\times d}$ (L need not be sym or PSD)

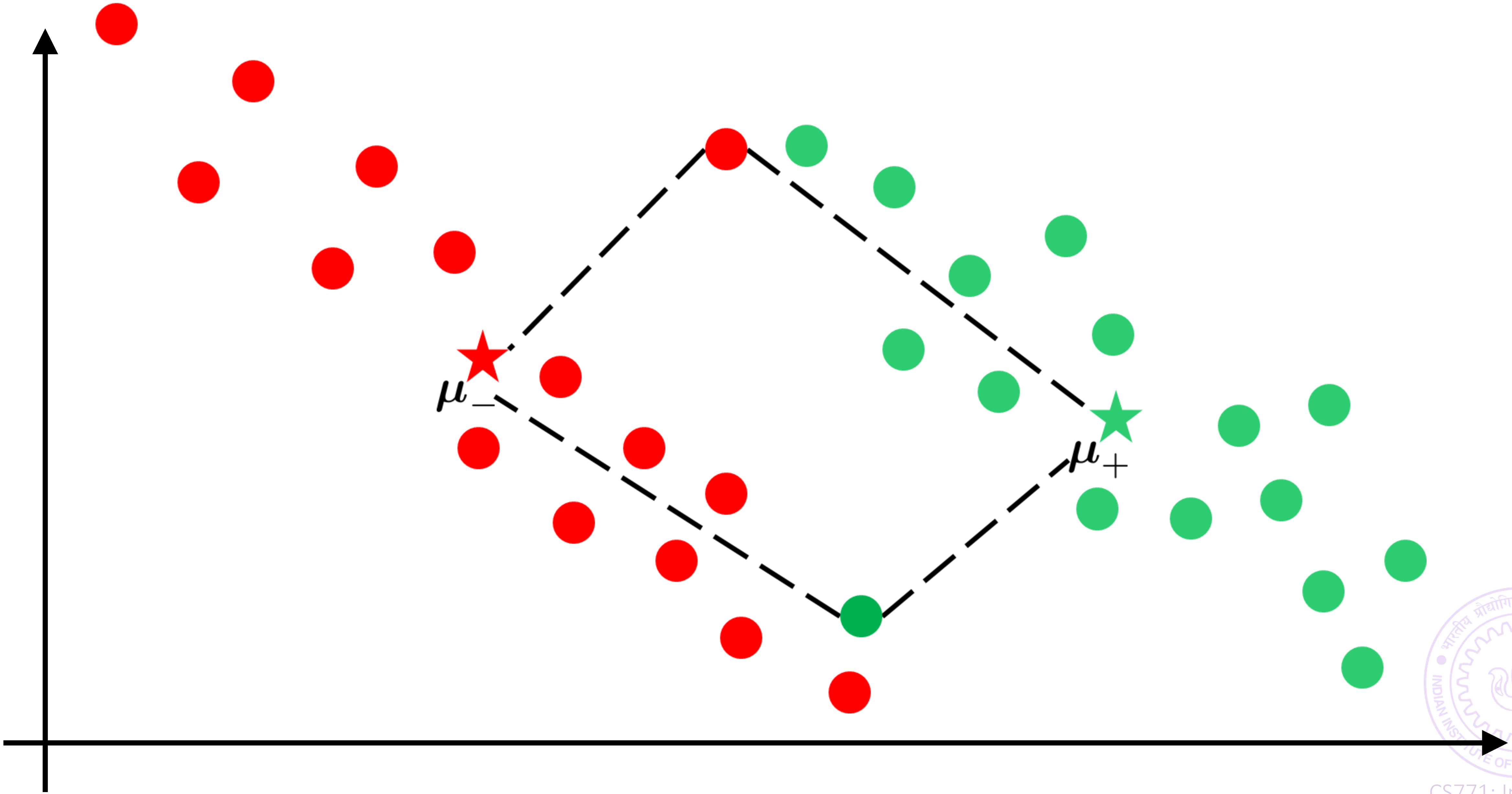
$$d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^\top A(\mathbf{x}-\mathbf{y})} = \sqrt{(\mathbf{x}-\mathbf{y})^\top LL^\top(\mathbf{x}-\mathbf{y})}$$

$$= \|L^\top\mathbf{x} - L^\top\mathbf{y}\|_2$$

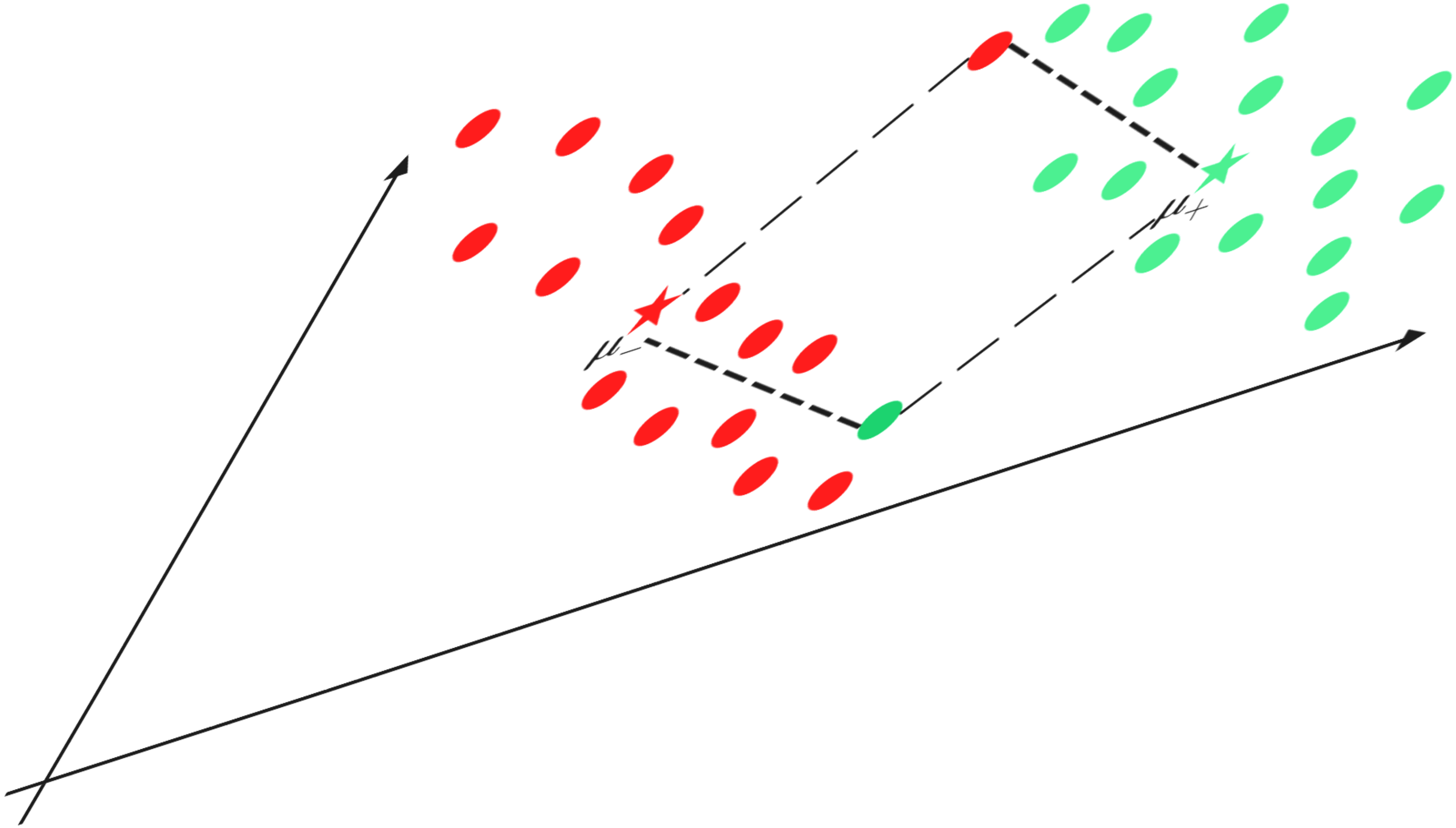
Oh! So the Mahalanobis distance is just Euclidean distance if we transform the vectors as $\mathbf{x} \mapsto L\mathbf{x}$



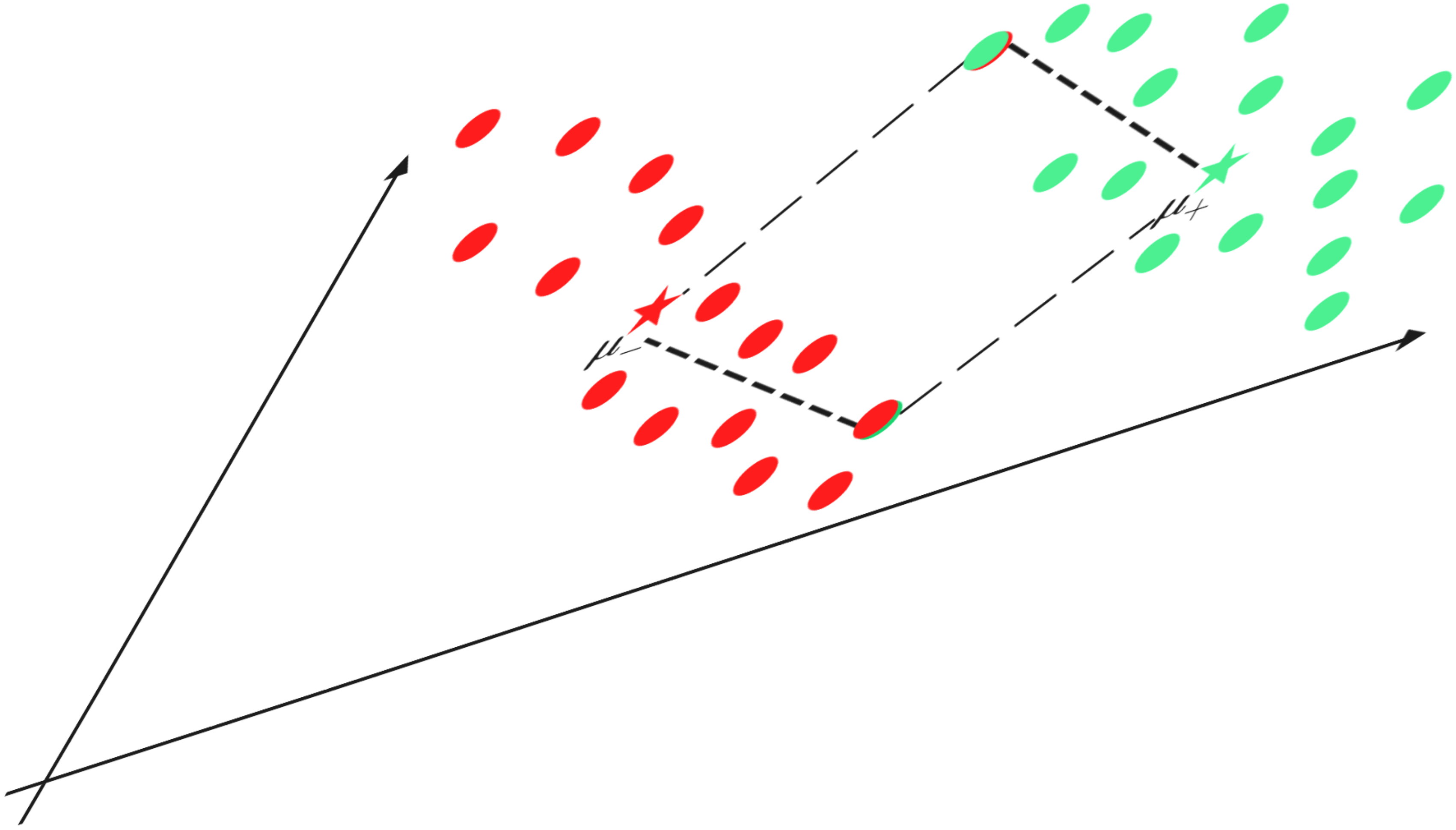
Why metric learning works



Why metric learning works

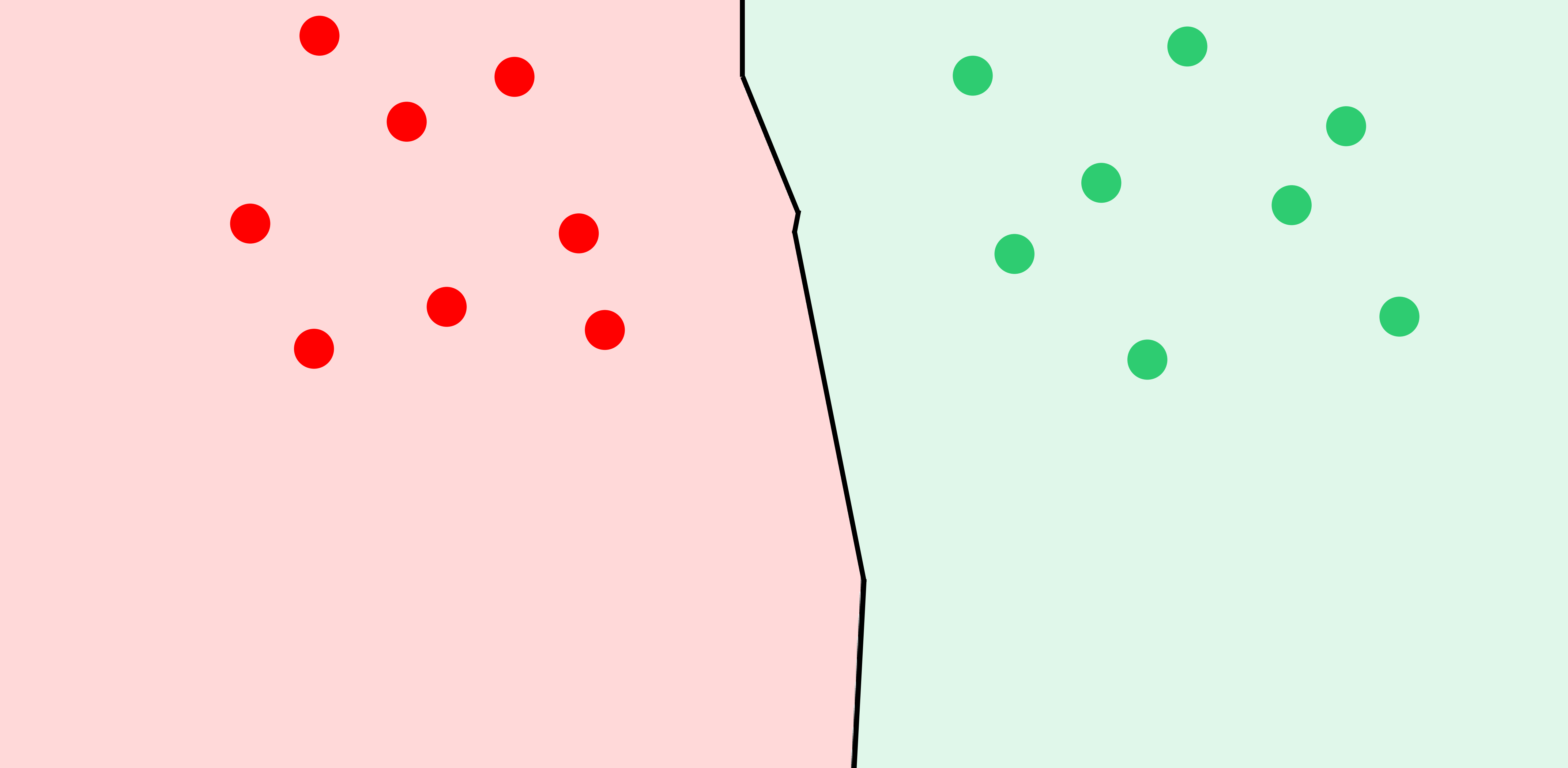


Why metric learning works

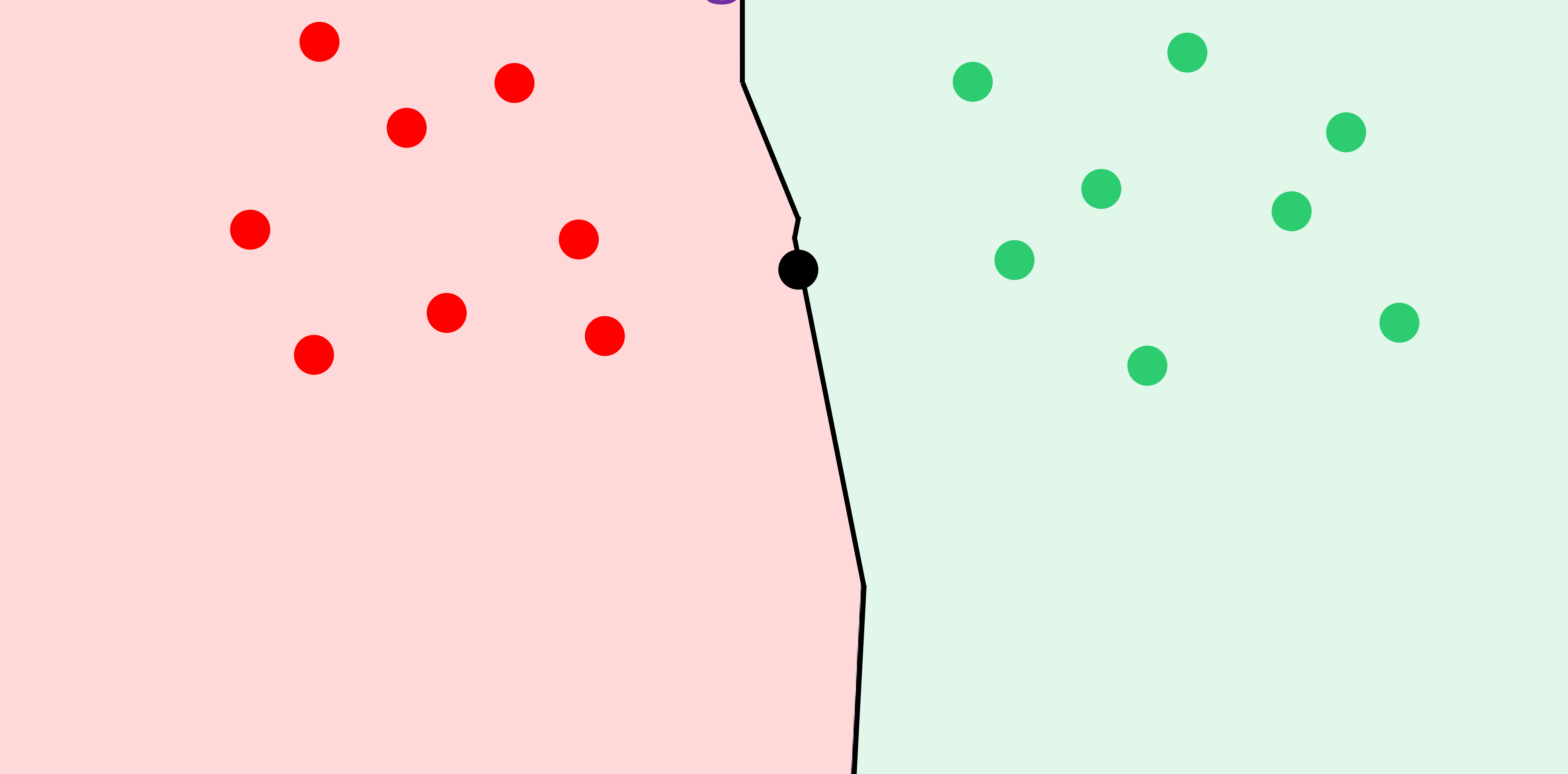


Back to Nearest Neighbors - kNN

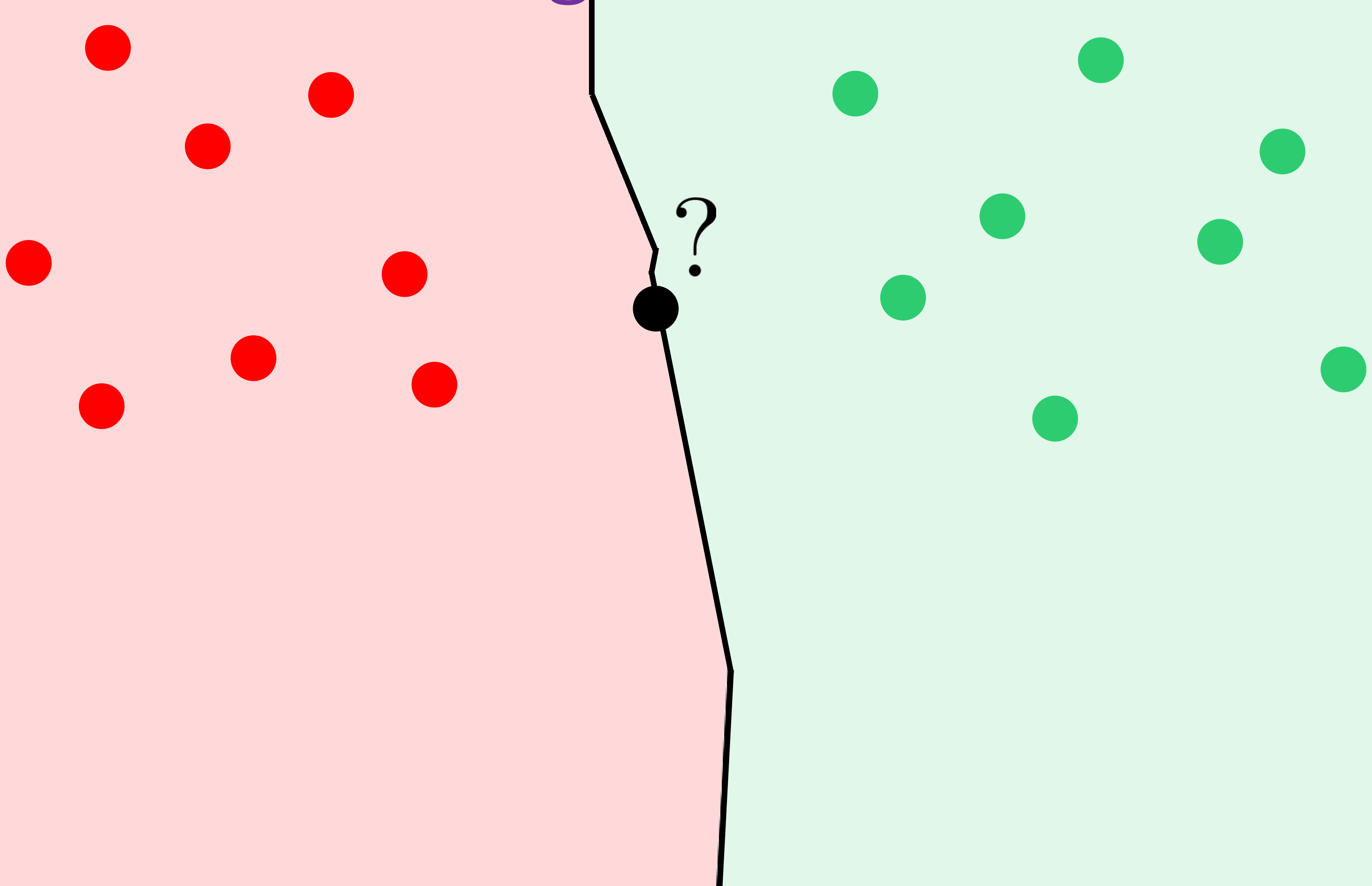
Back to Nearest Neighbors - kNN



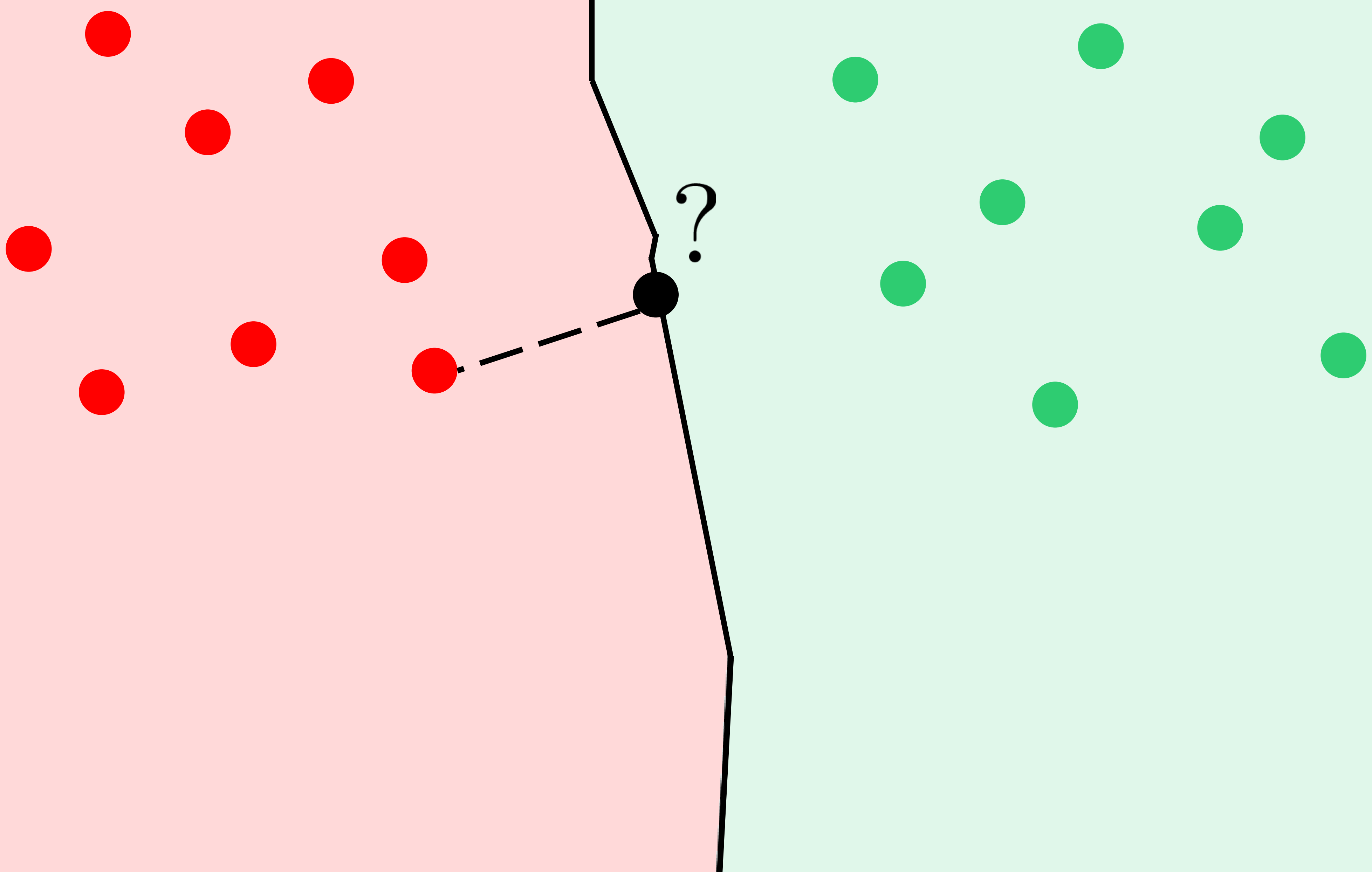
Back to Nearest Neighbors - kNN



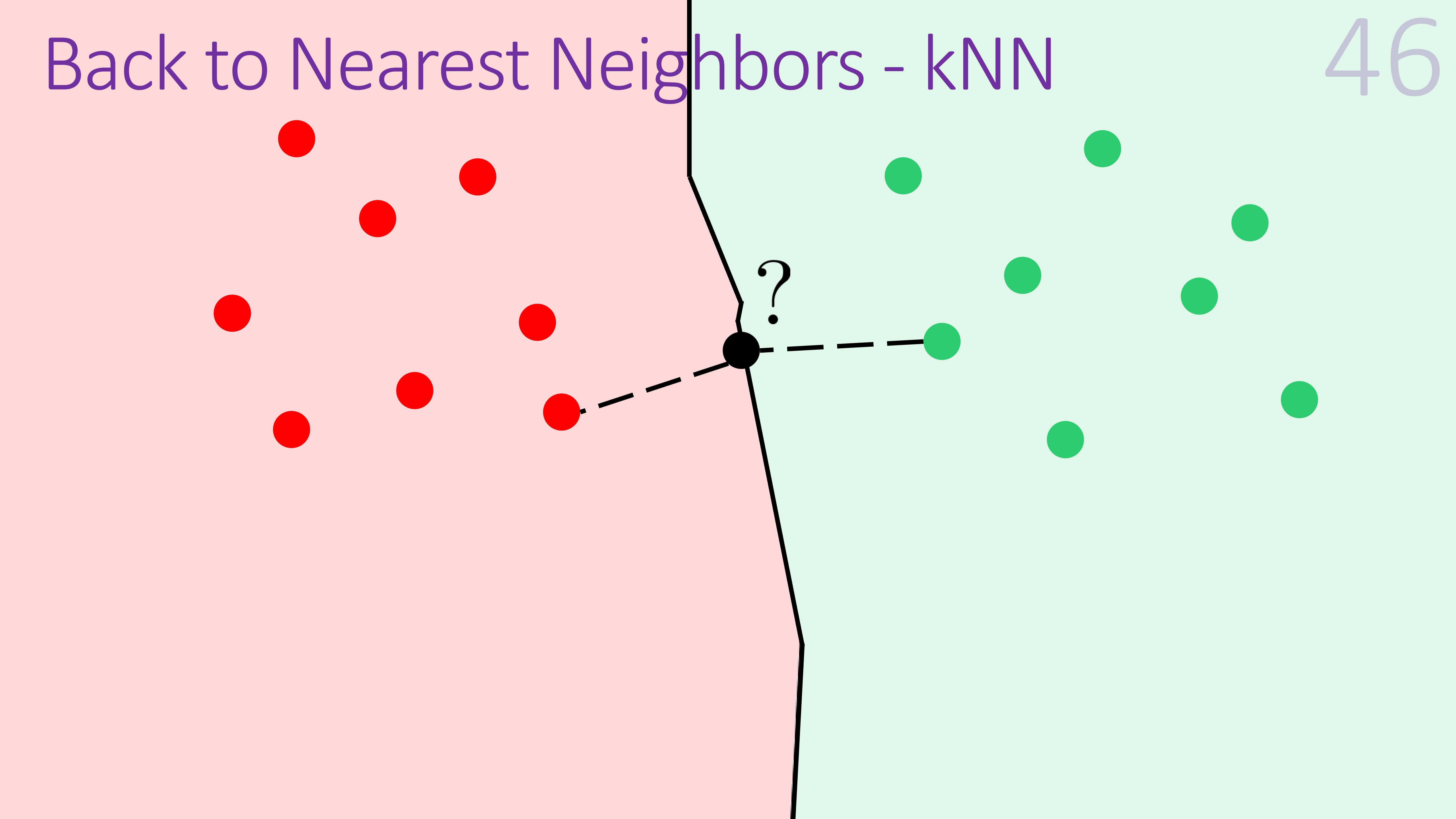
Back to Nearest Neighbors - kNN



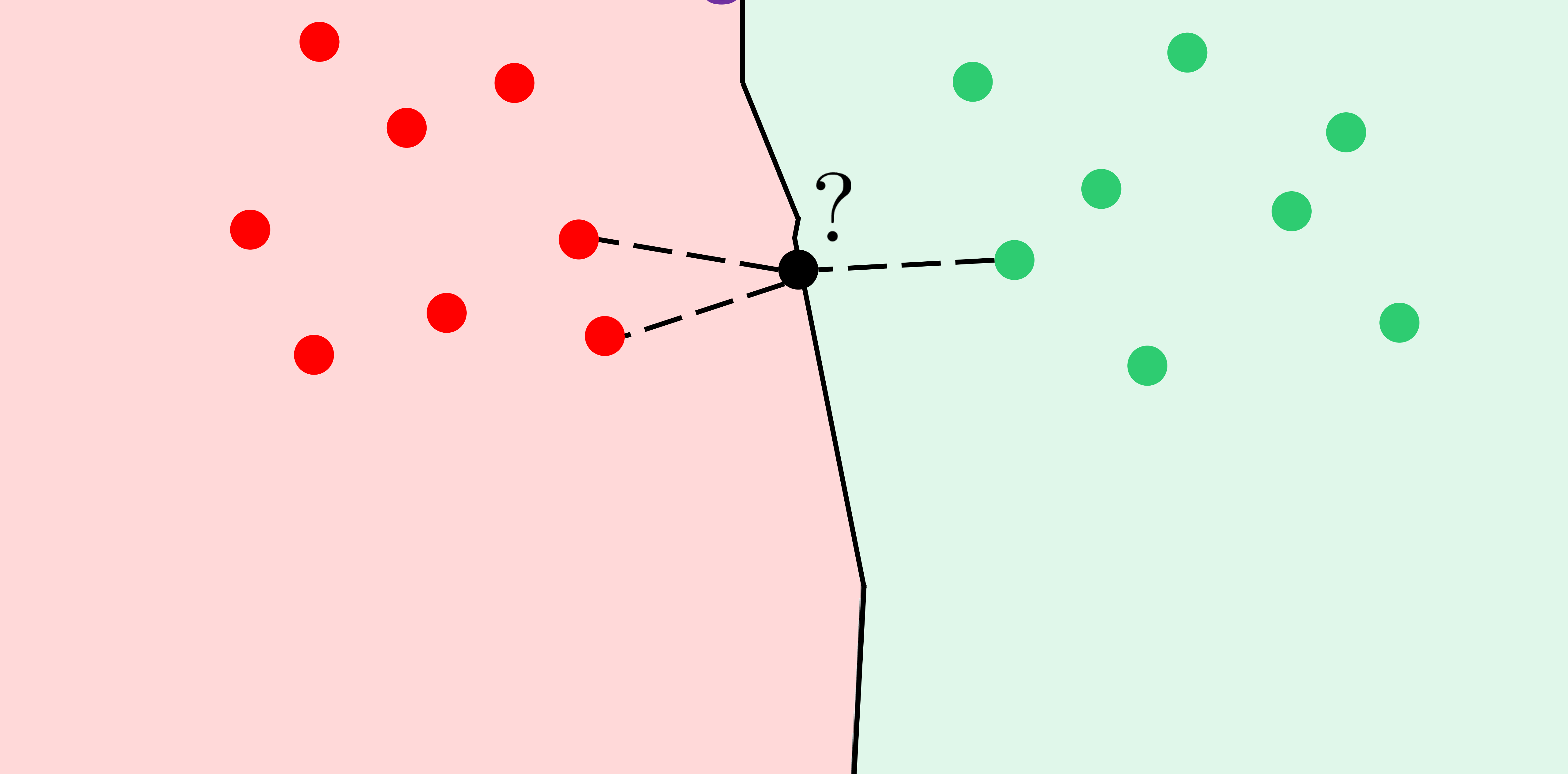
Back to Nearest Neighbors - kNN



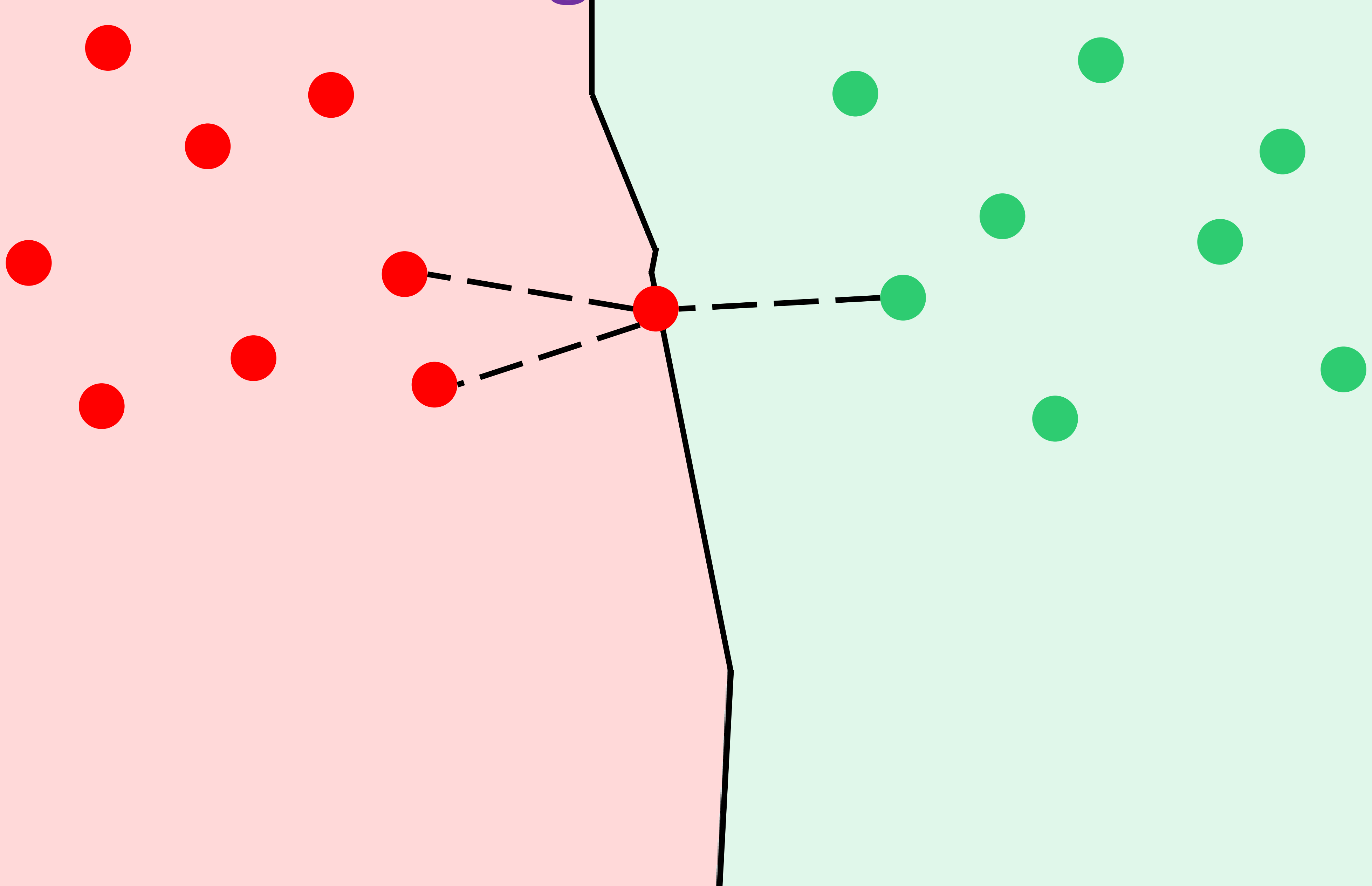
Back to Nearest Neighbors - kNN



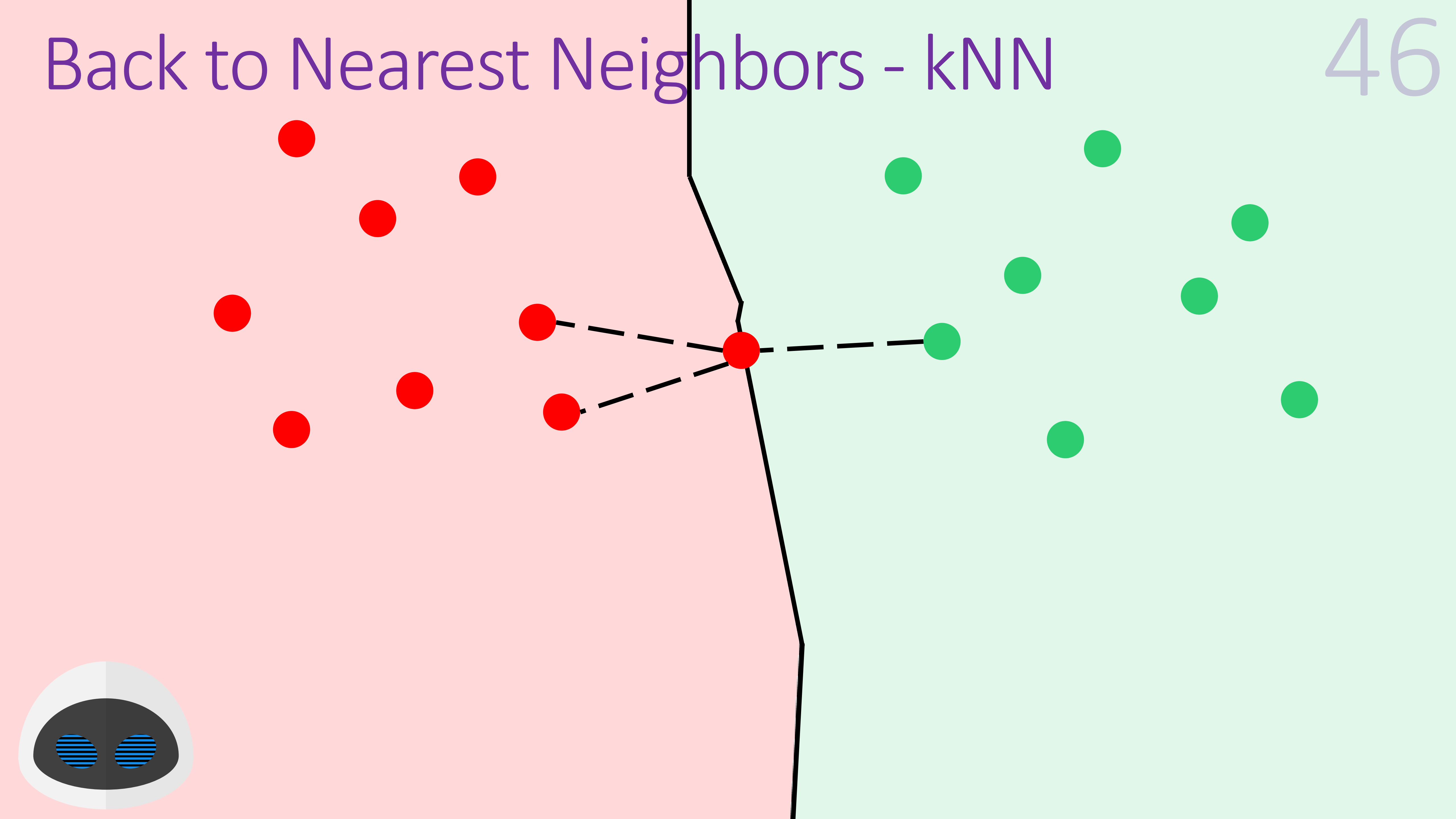
Back to Nearest Neighbors - kNN



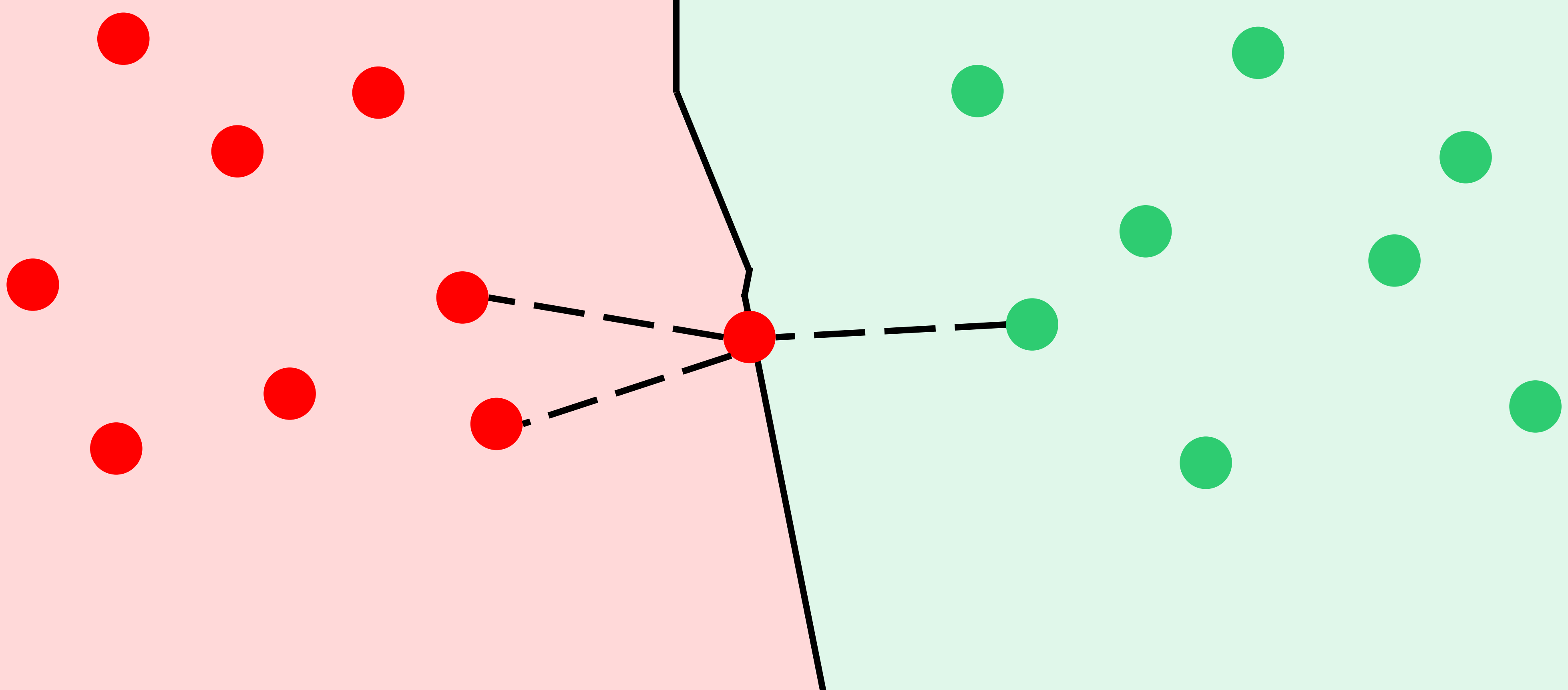
Back to Nearest Neighbors - kNN

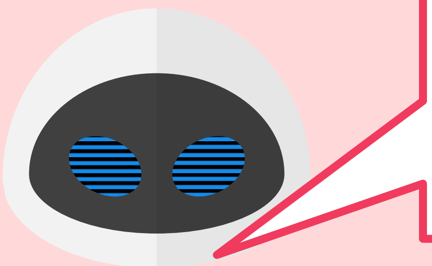


Back to Nearest Neighbors - kNN



Back to Nearest Neighbors - kNN

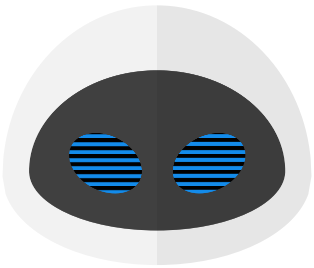




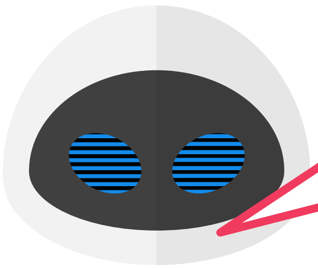
Instead of looking at just the label of the nearest neighbour, look at the labels of the k nearest neighbors and choose the one that is most popular

Back to Nearest Neighbors - rNN

Back to Nearest Neighbors - rNN

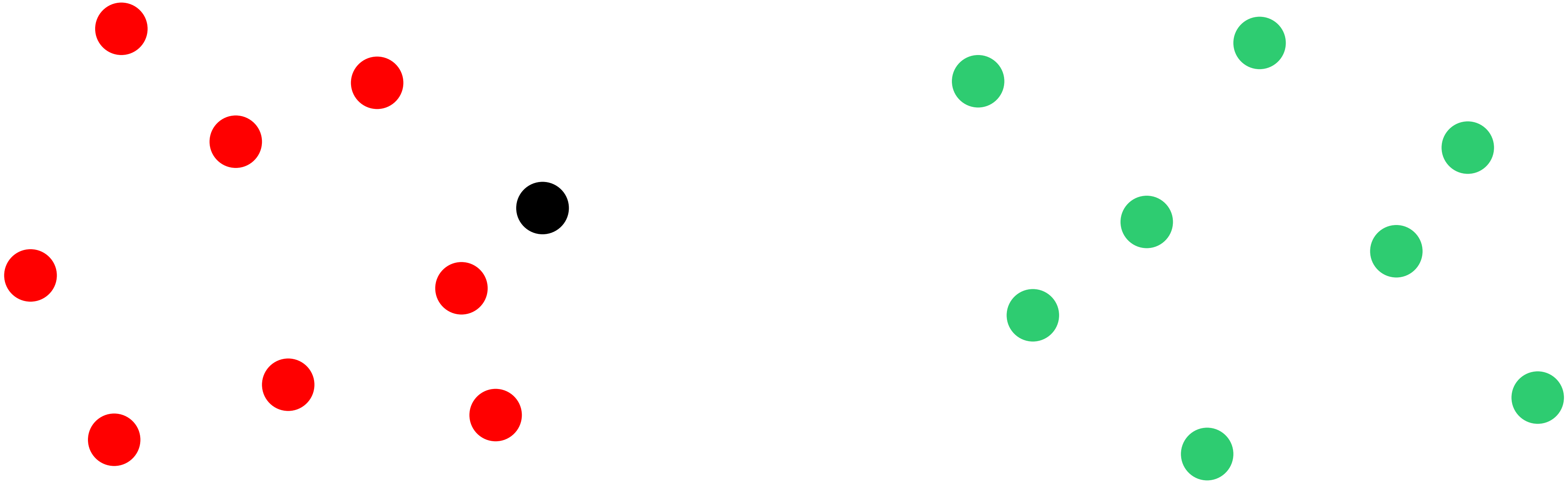


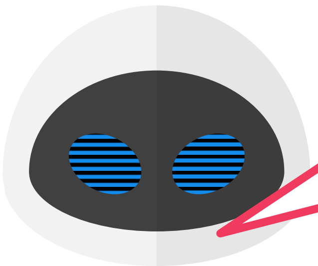
Back to Nearest Neighbors - rNN



Look at all neighbors who are within an r radius of the test point and choose the label most popular among the neighbors

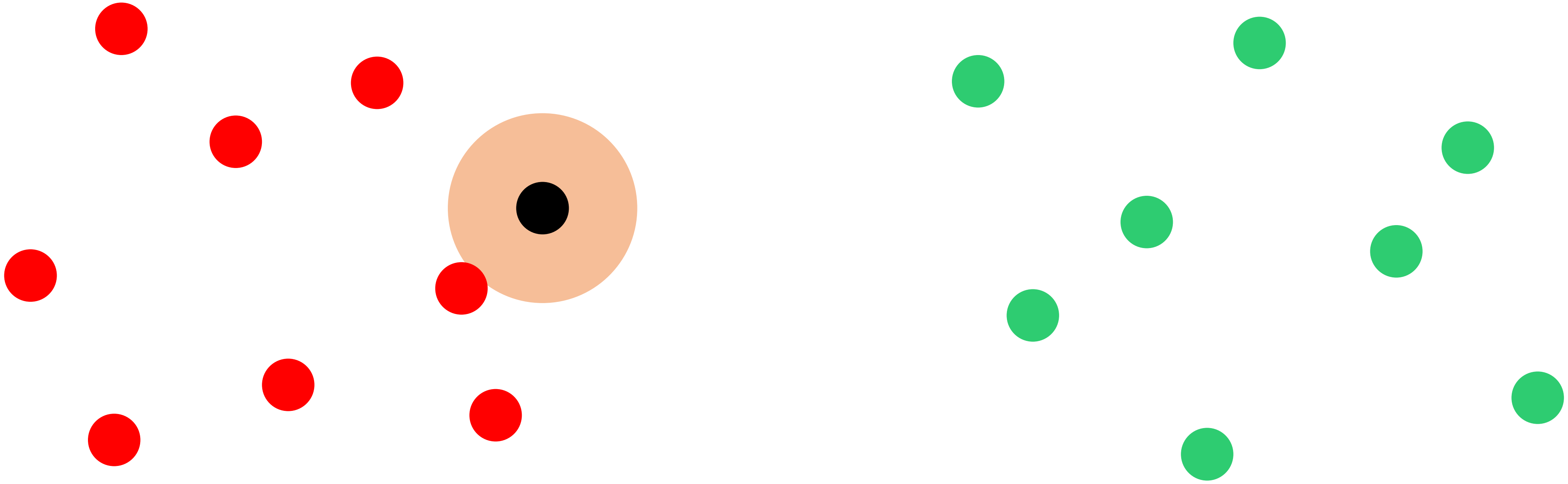
Back to Nearest Neighbors - rNN

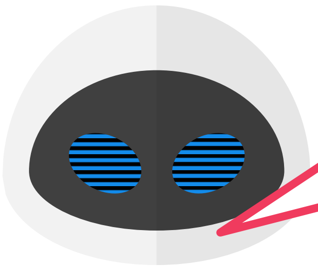




Look at all neighbors who are within an r radius of the test point and choose the label most popular among the neighbors

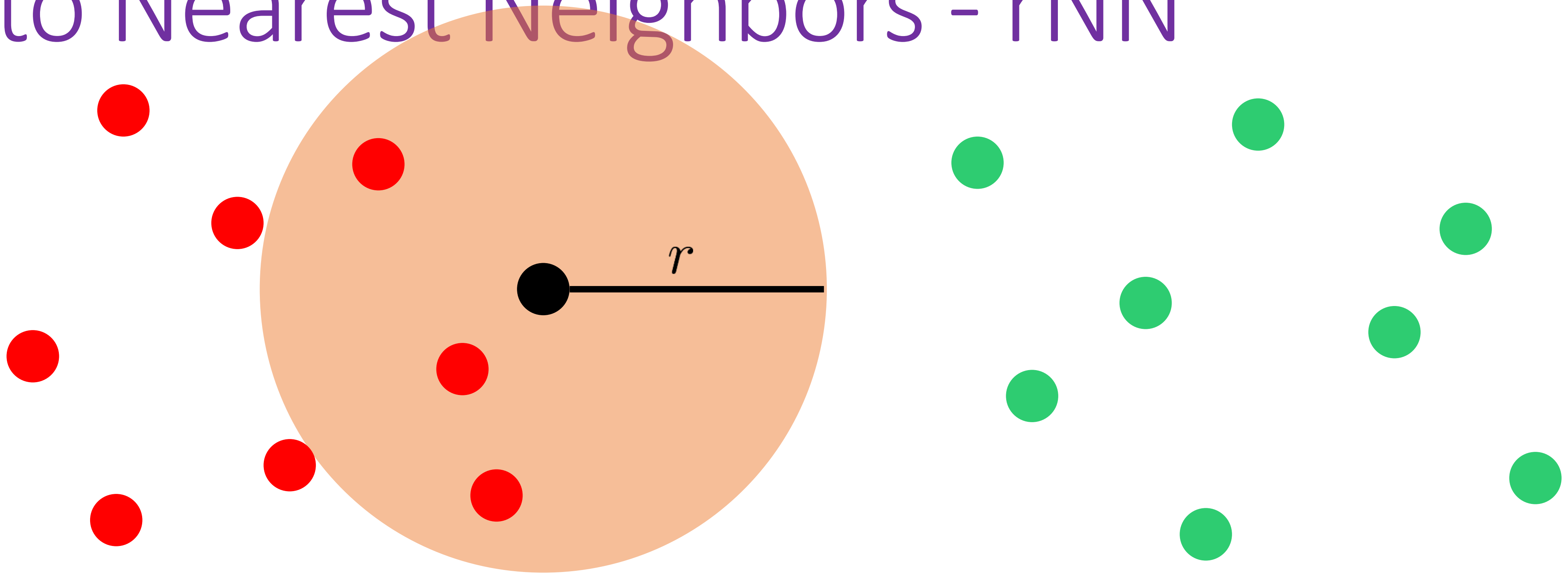
Back to Nearest Neighbors - rNN

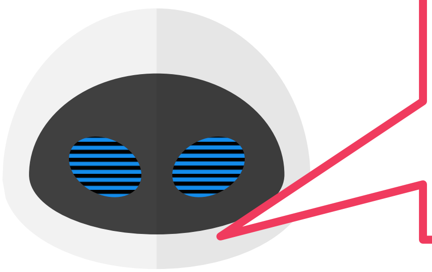




Look at all neighbors who are within an r radius of the test point and choose the label most popular among the neighbors

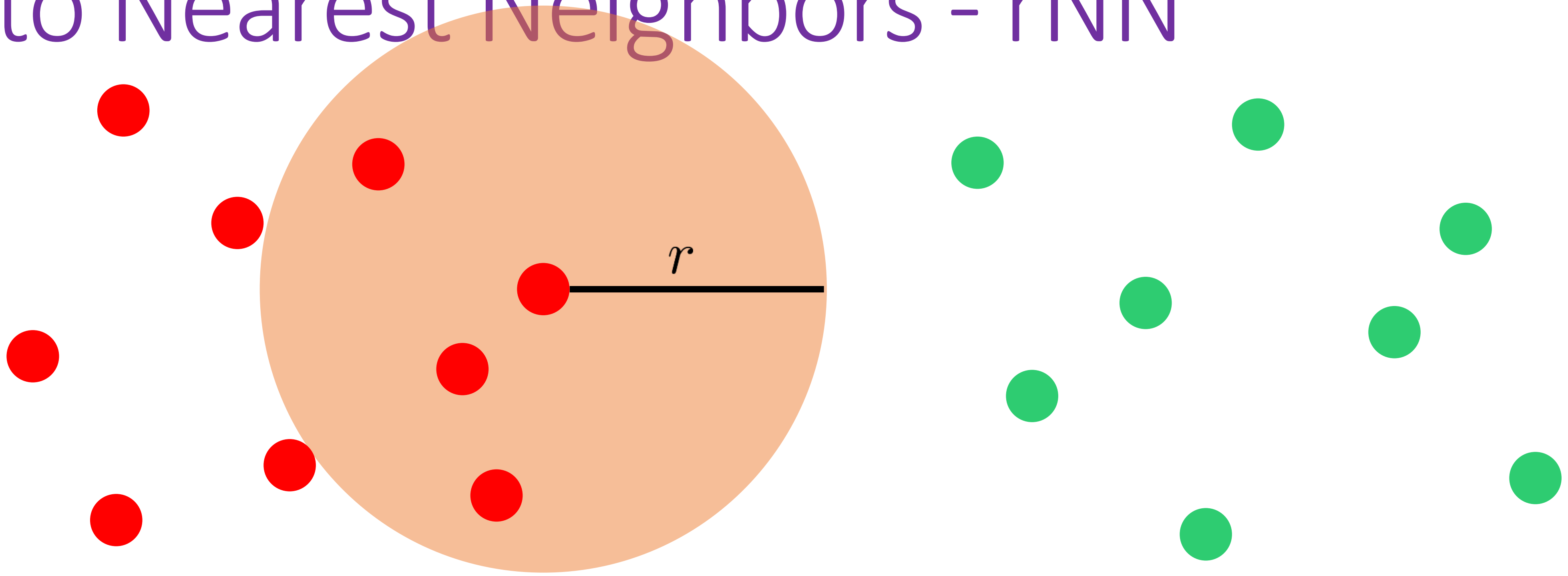
Back to Nearest Neighbors - rNN

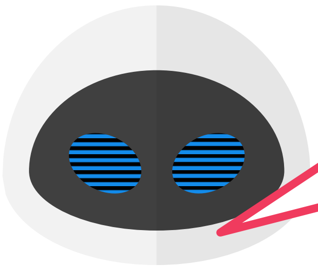




Look at all neighbors who are within an r radius of the test point and choose the label most popular among the neighbors

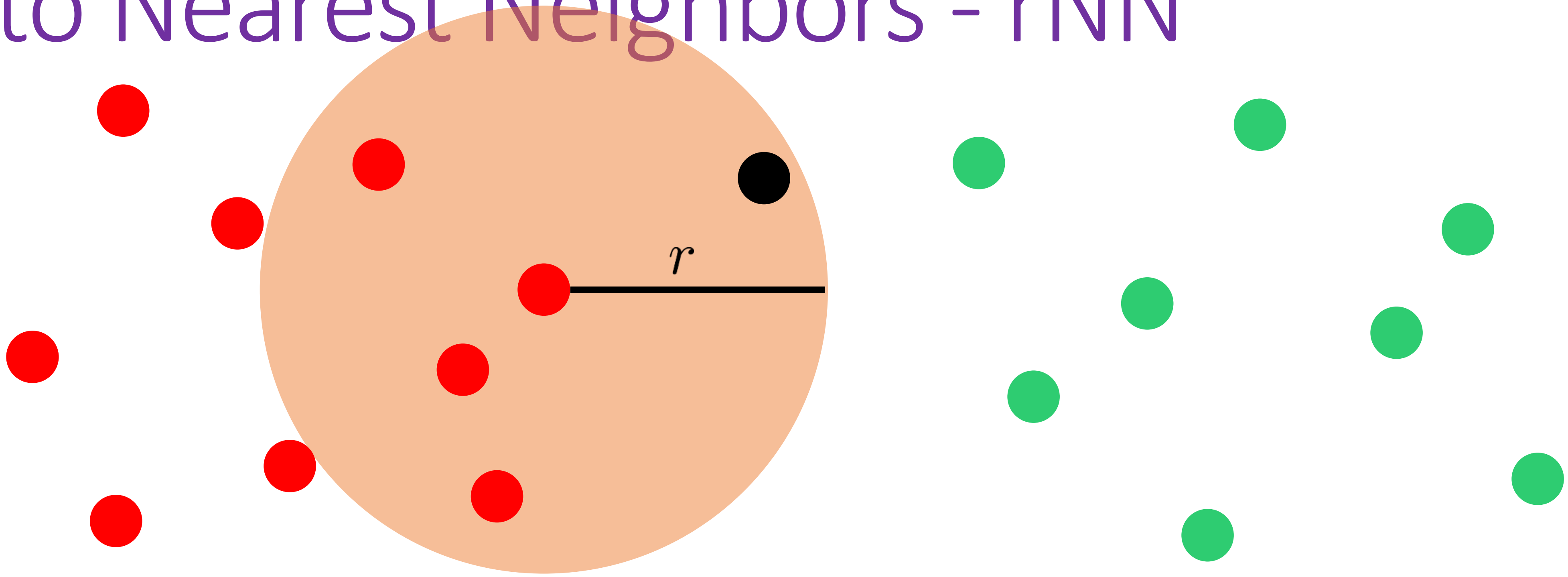
Back to Nearest Neighbors - rNN

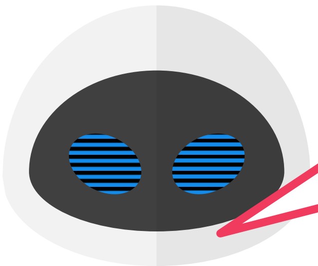




Look at all neighbors who are within an r radius of the test point and choose the label most popular among the neighbors

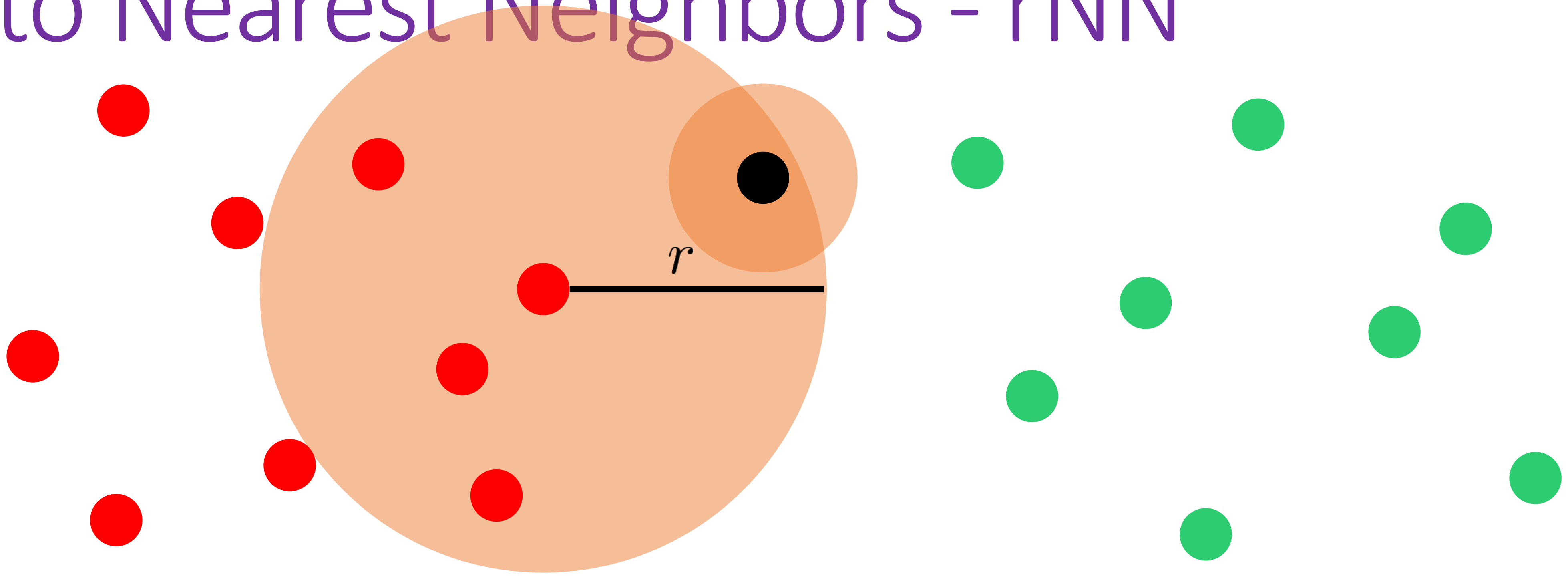
Back to Nearest Neighbors - rNN

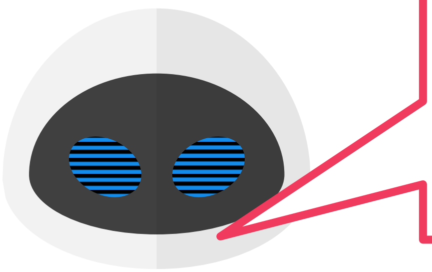




Look at all neighbors who are within an r radius of the test point and choose the label most popular among the neighbors

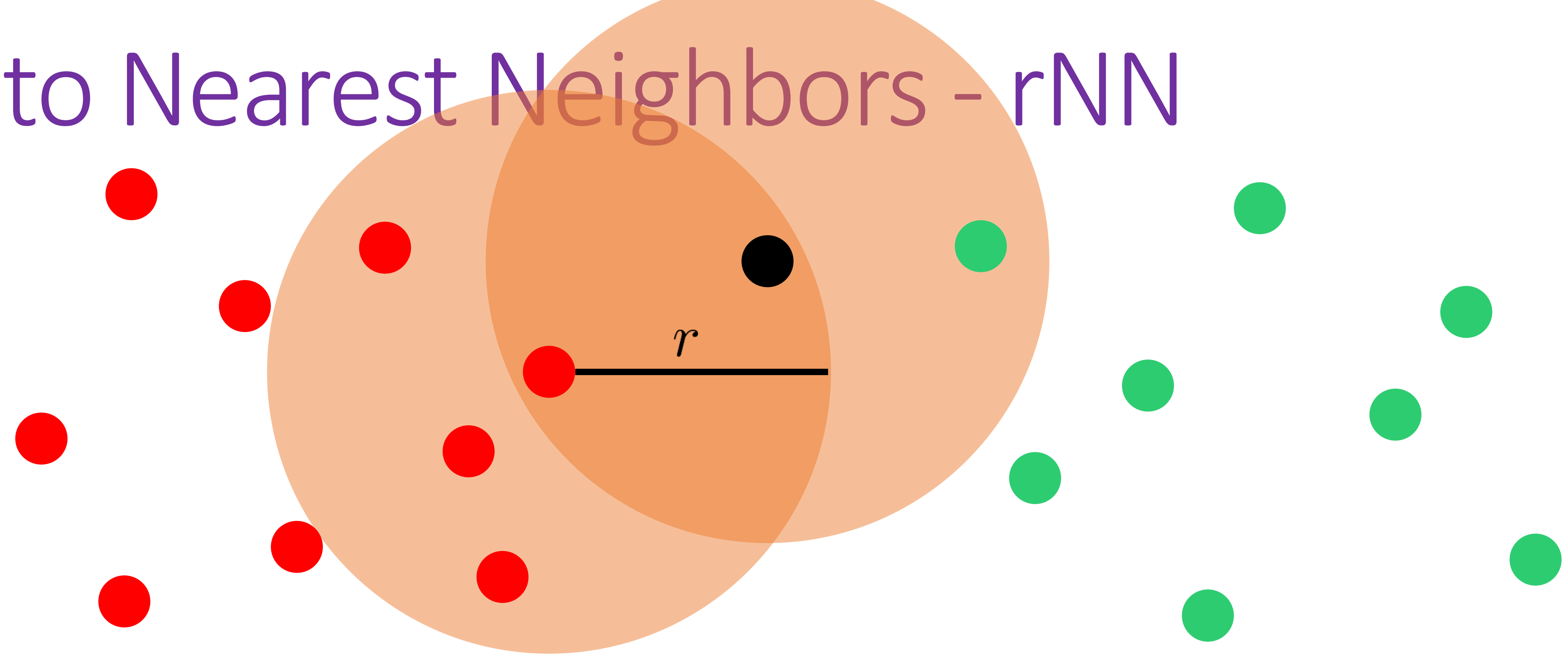
Back to Nearest Neighbors - rNN

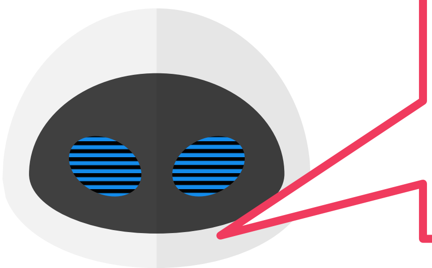




Look at all neighbors who are within an r radius of the test point and choose the label most popular among the neighbors

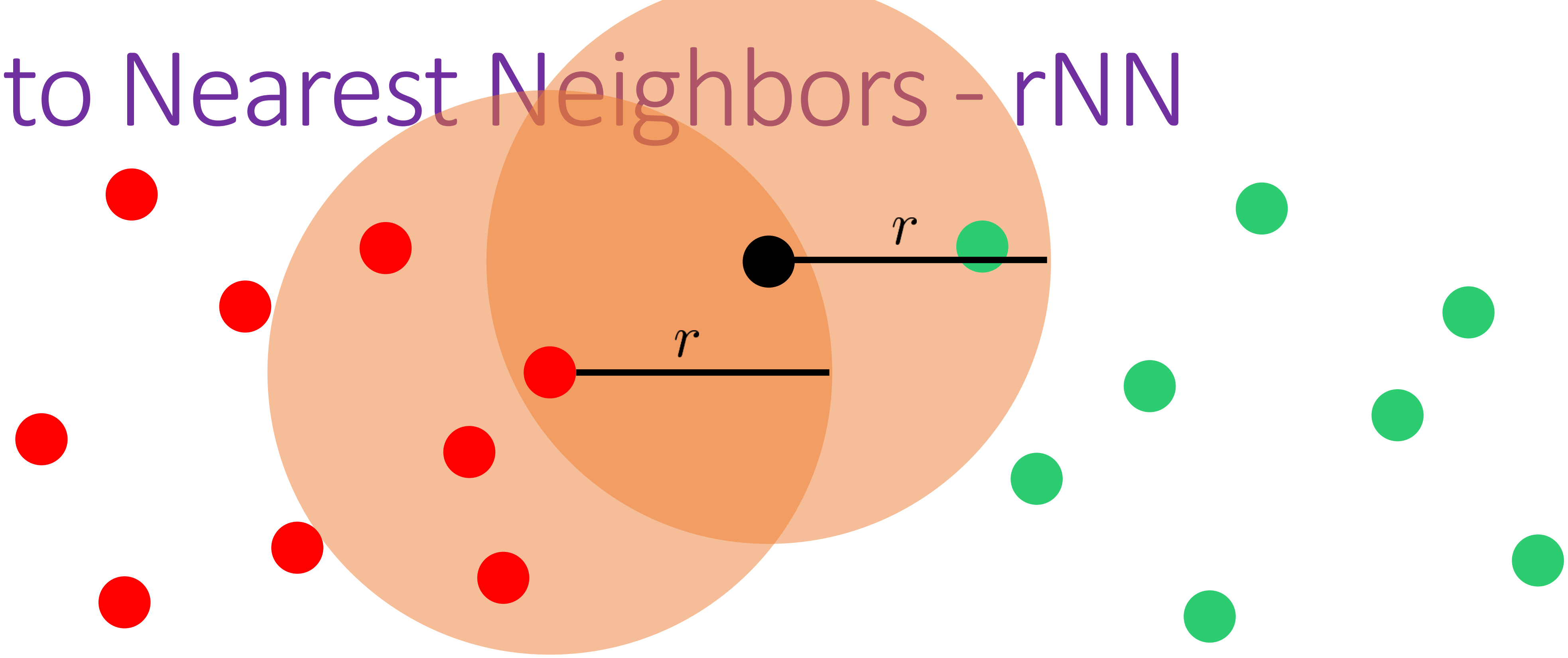
Back to Nearest Neighbors - rNN

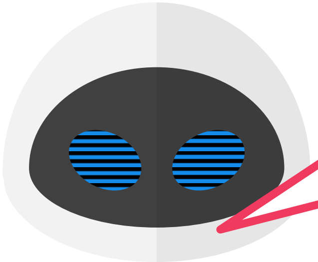




Look at all neighbors who are within an r radius of the test point and choose the label most popular among the neighbors

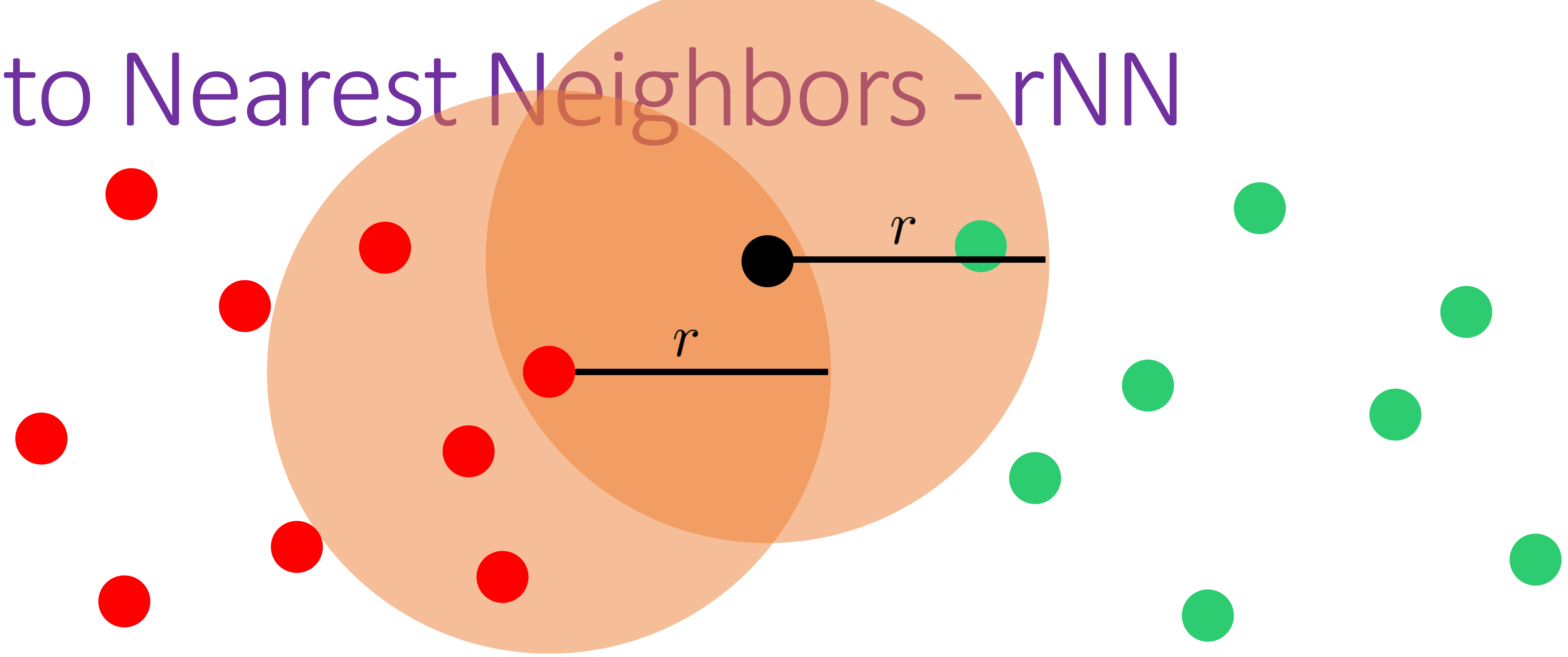
Back to Nearest Neighbors - rNN

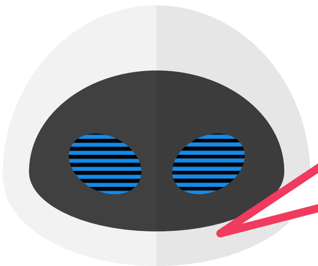




Look at all neighbors who are within an r radius of the test point and choose the label most popular among the neighbors

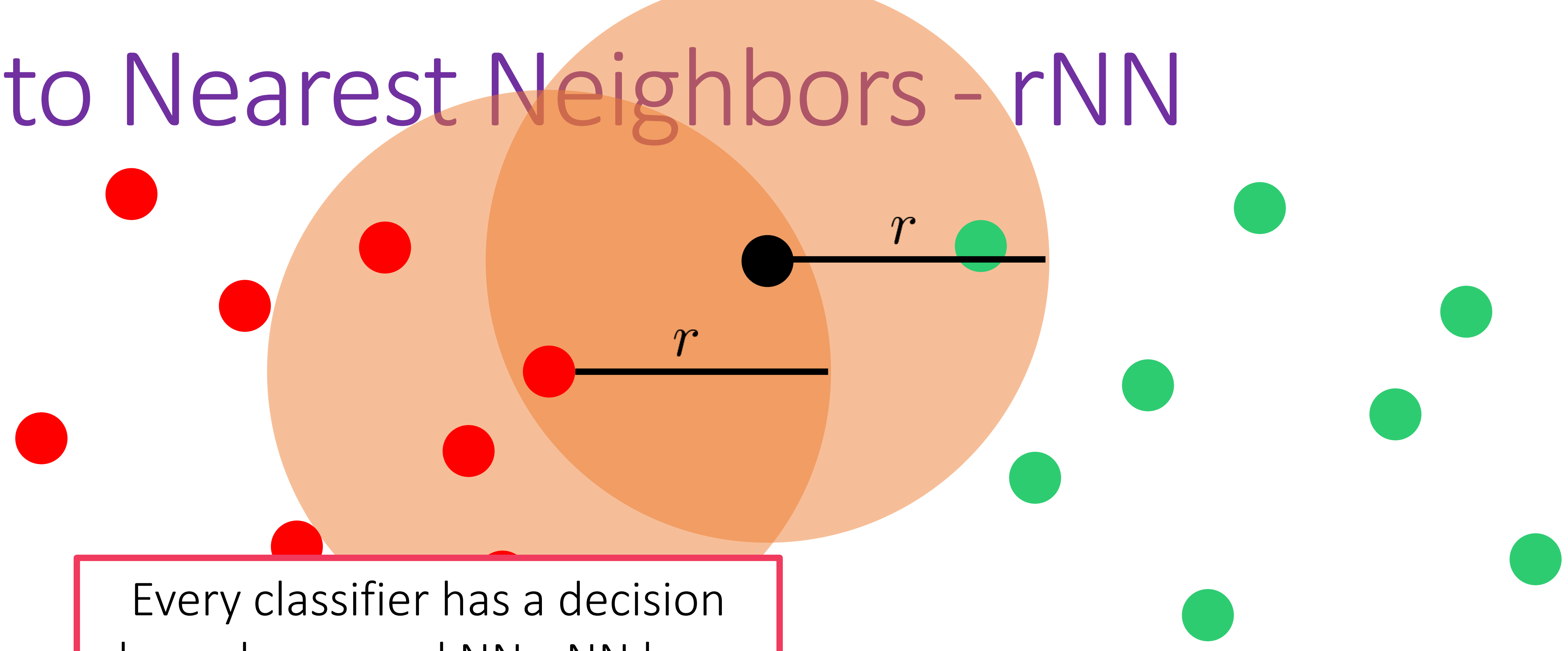
Back to Nearest Neighbors - rNN





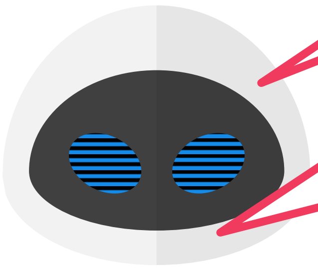
Look at all neighbors who are within an r radius of the test point and choose the label most popular among the neighbors

Back to Nearest Neighbors - rNN

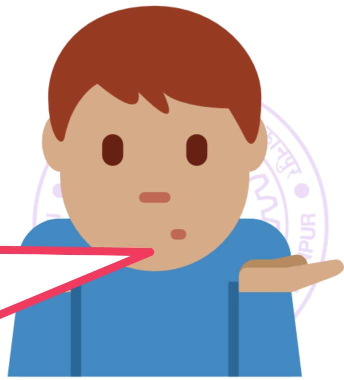


Every classifier has a decision boundary even kNN, rNN have some decision boundary (which we hope are better than 1NN's)

Look at all neighbors who are within an r radius of the test point and choose the label most popular among the neighbors



How should I decide which value of k or r to use? Also, should I use kNN or rNN?



Hyperparameter Tuning in ML

Constants like k in kNN, r in rNN, or even the metric to use in LwP are called *hyperparameters* of an ML algorithm

Just a fancy name (model vectors like $\mathbf{w}$ are called *parameters*)

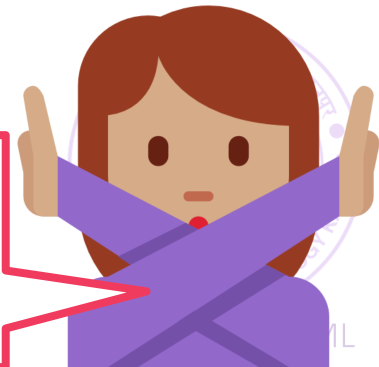
We usually tune these hyperparameters by setting them to a value that gives us highest test accuracy

Take out a part of the training data and pretend it is test data for the purpose of hyperparameter tuning ☺

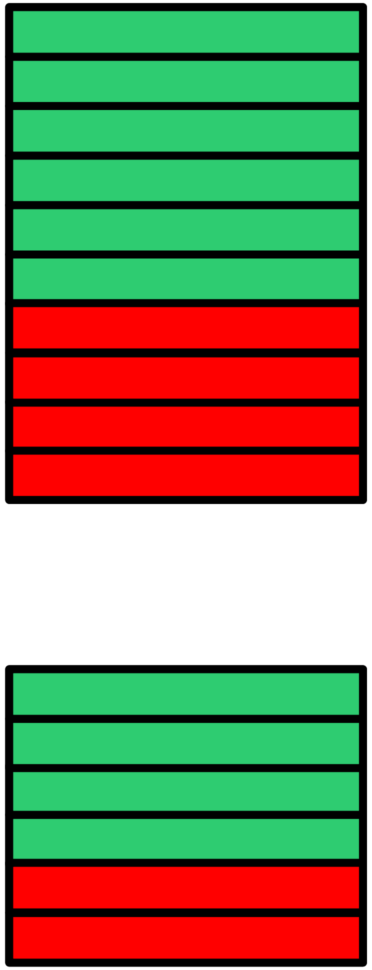
This part of data that is a mock test for us is called *validation data*

Let us look at the two most popular ways of creating such validation datasets

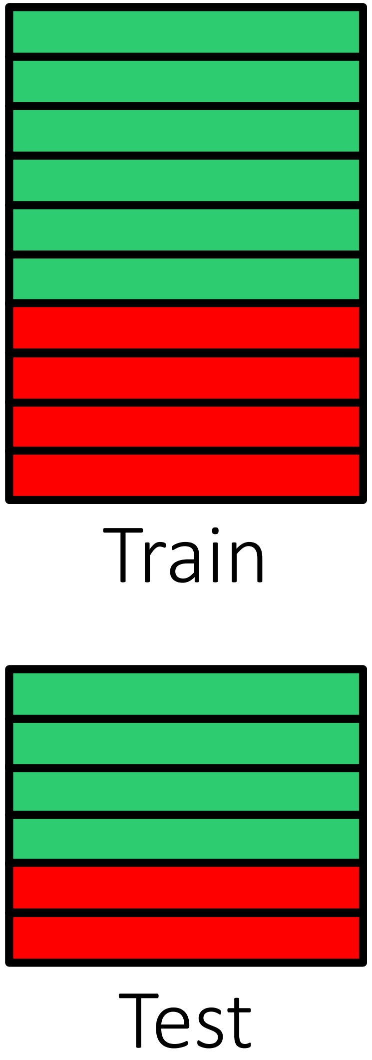
Looking at test data during training is an execution-worthy crime!



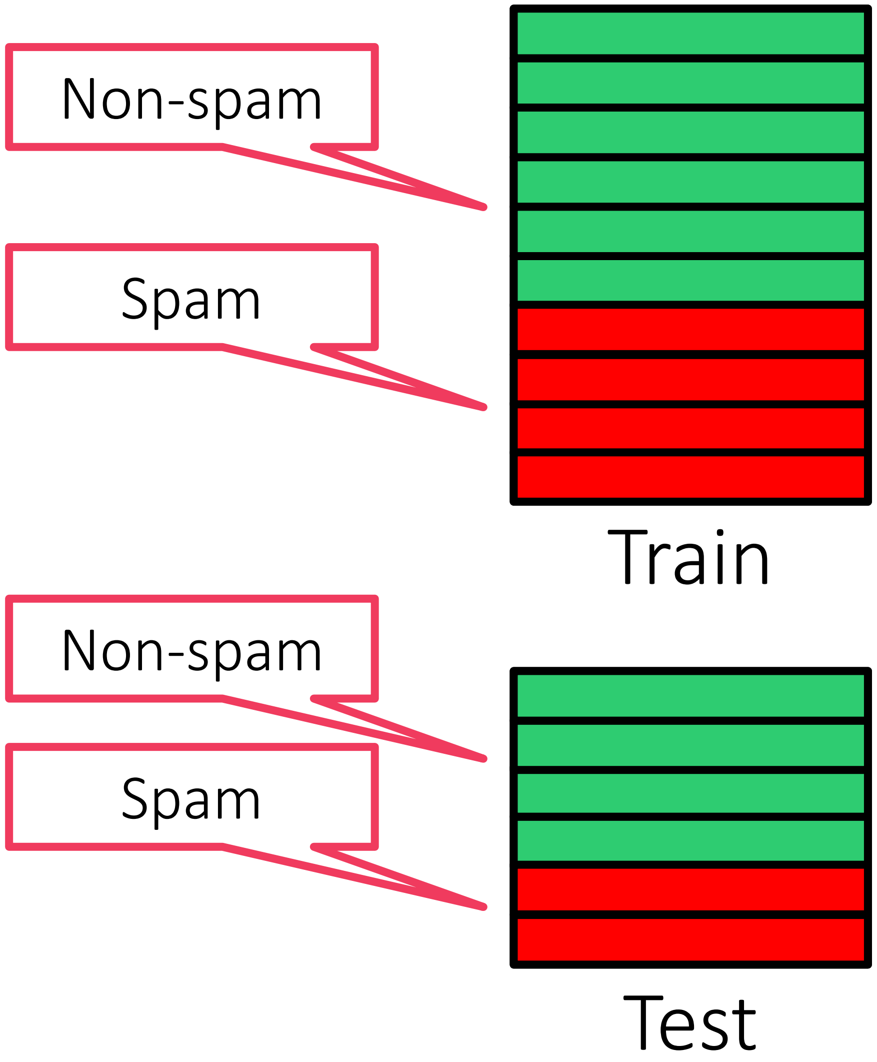
Held-out Validation



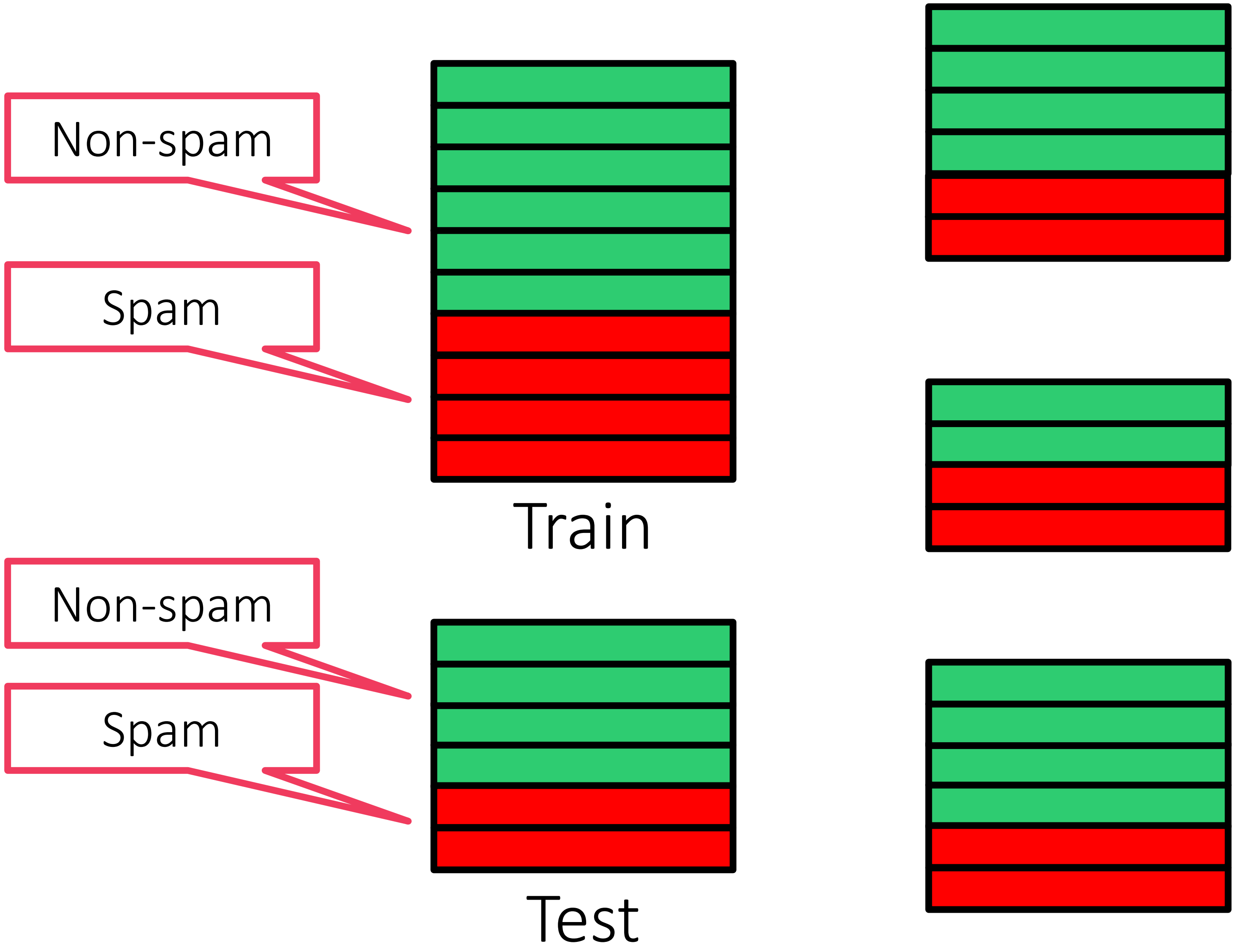
Held-out Validation



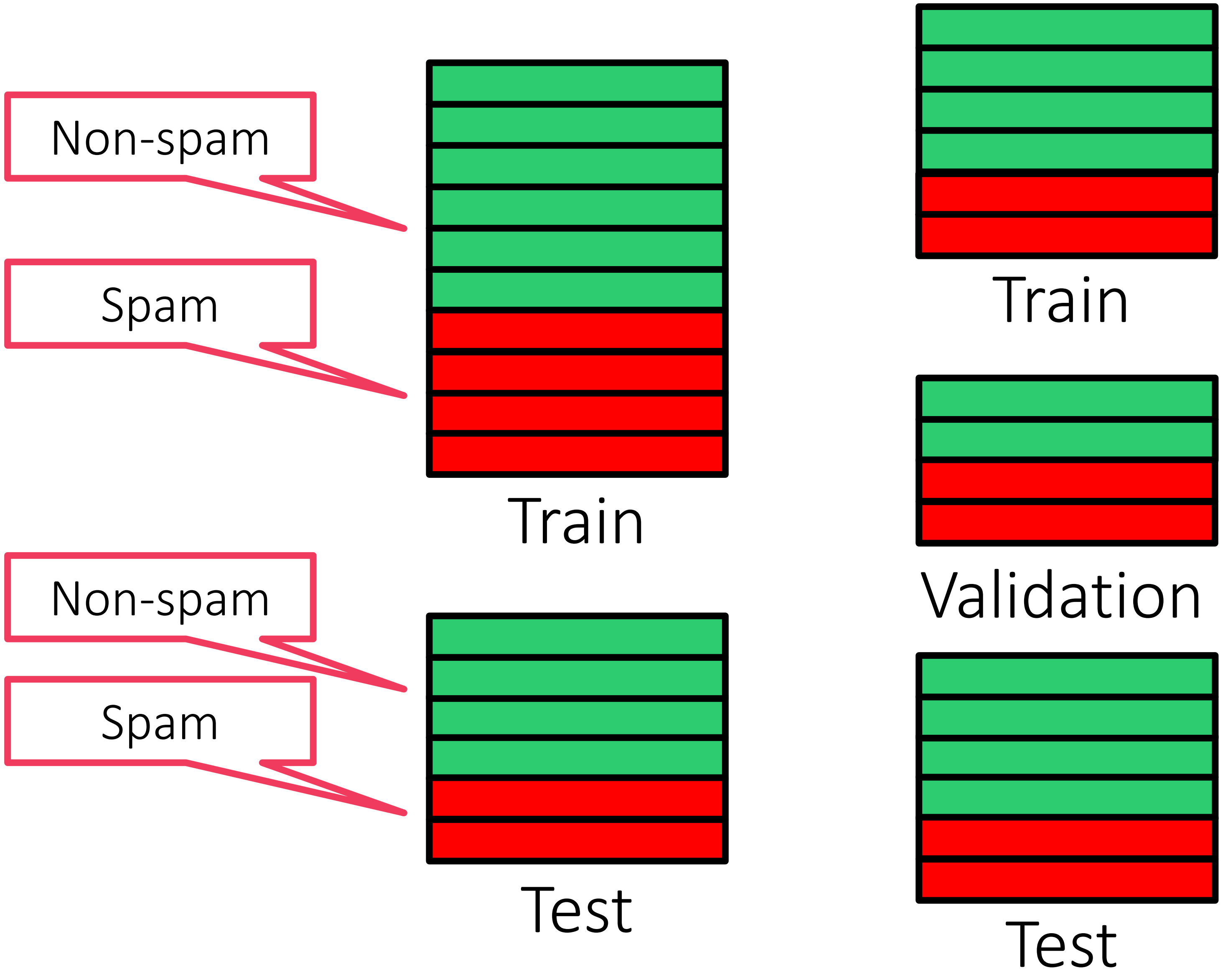
Held-out Validation



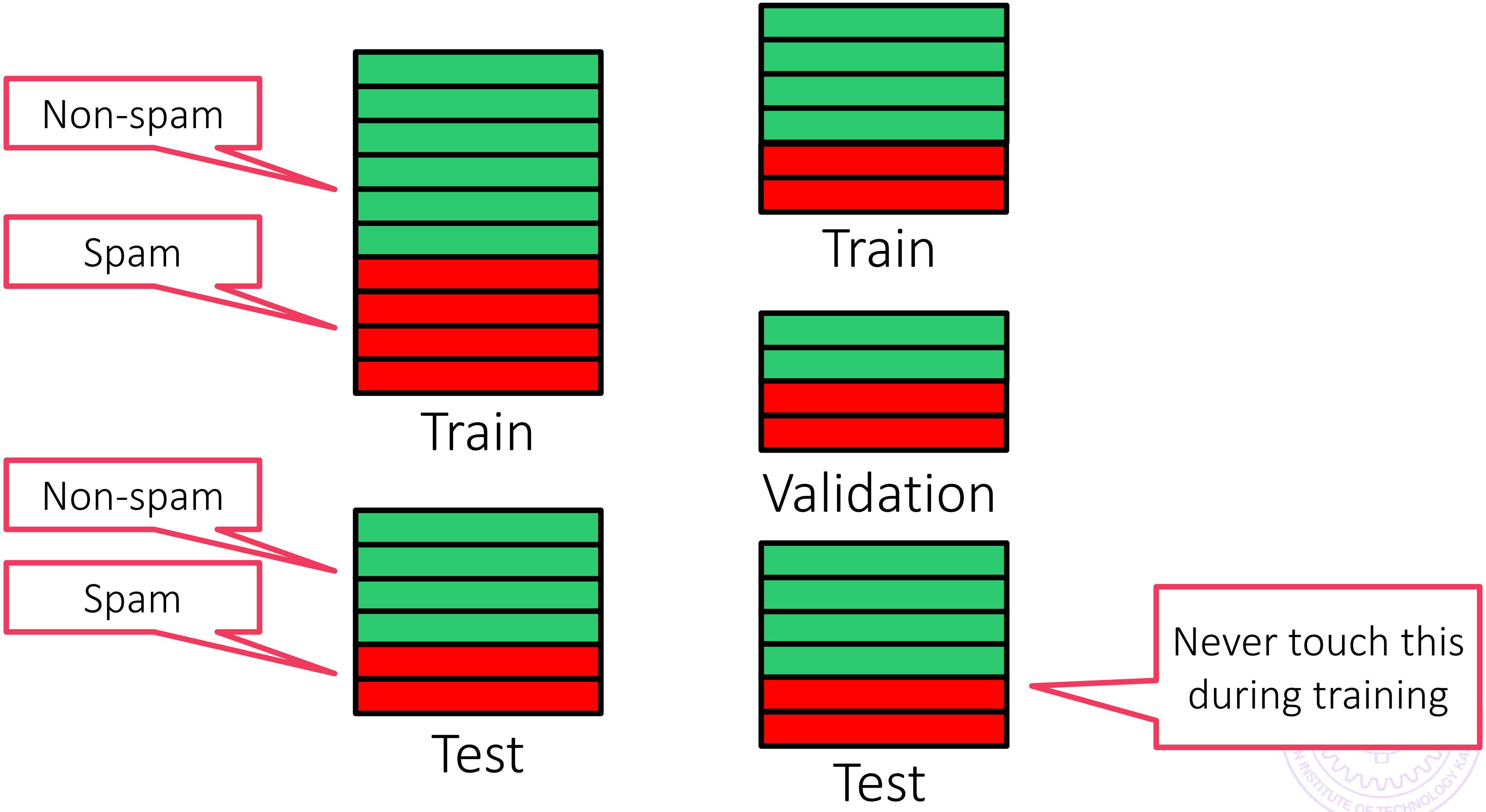
Held-out Validation



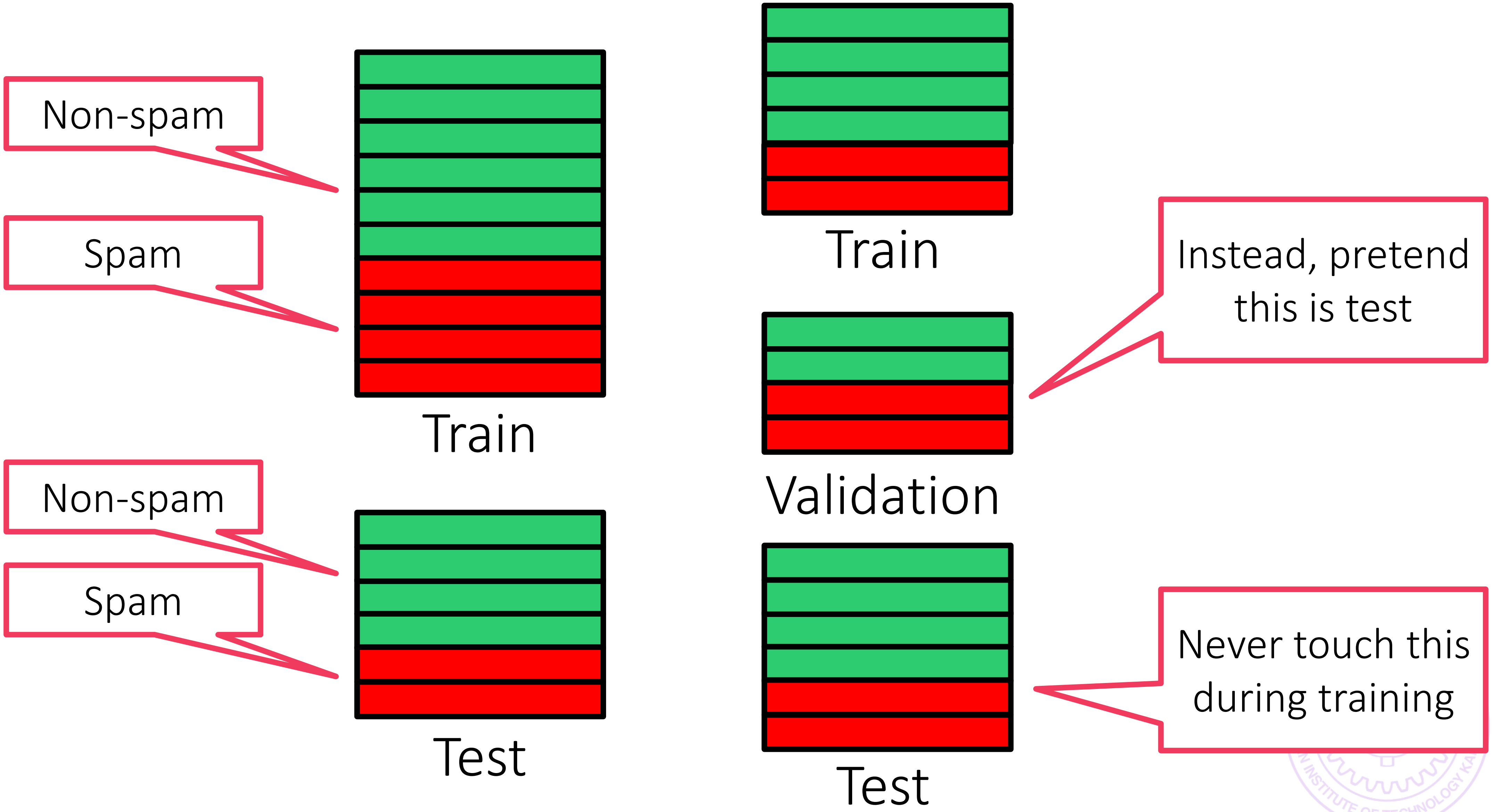
Held-out Validation



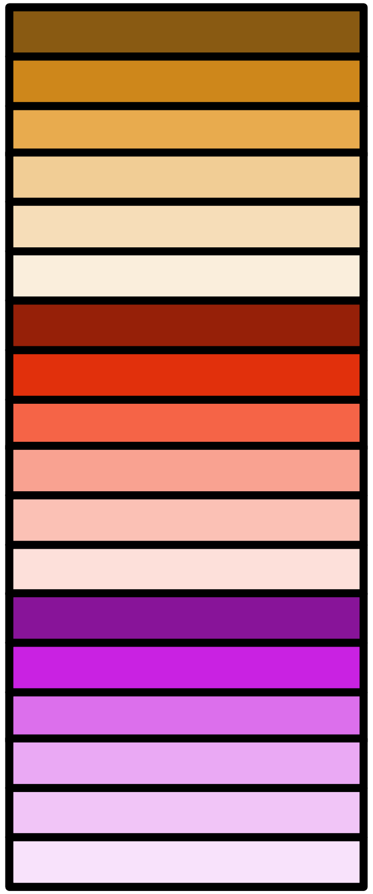
Held-out Validation



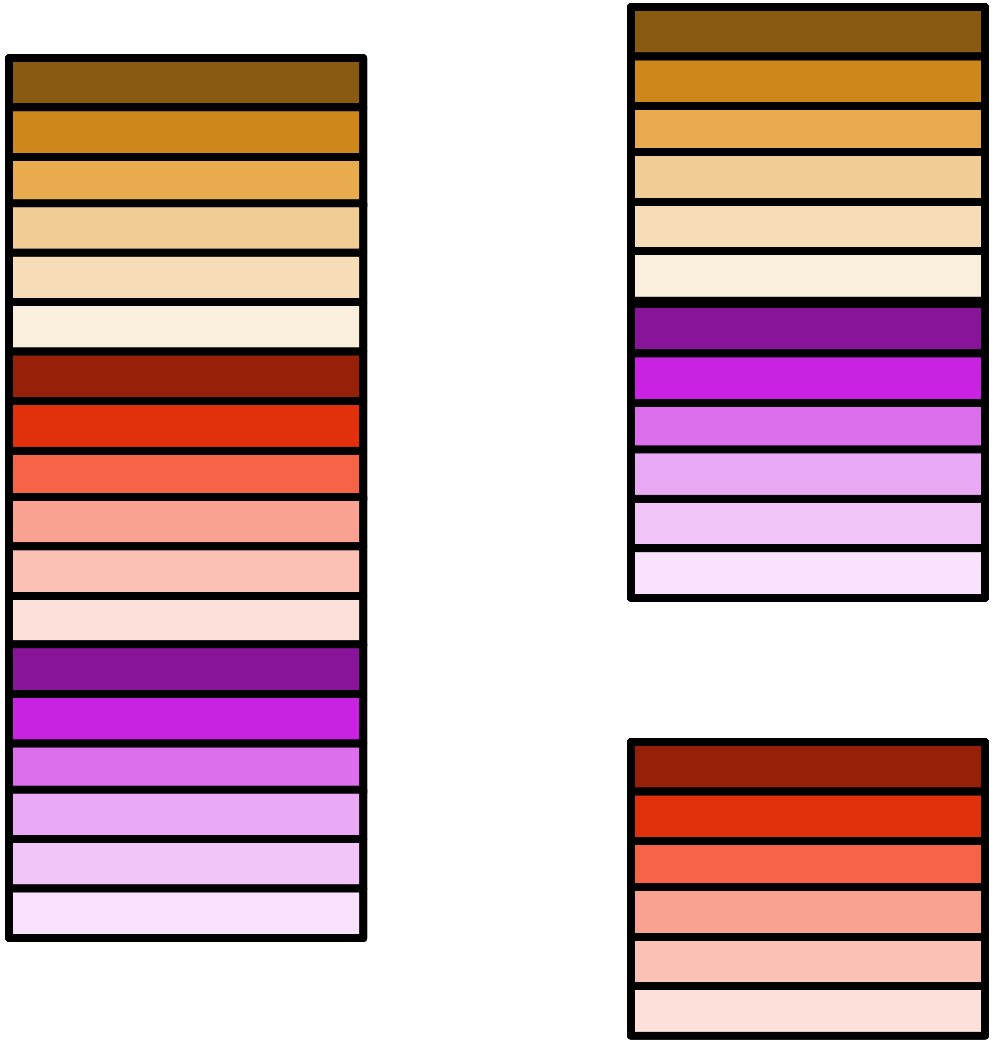
Held-out Validation



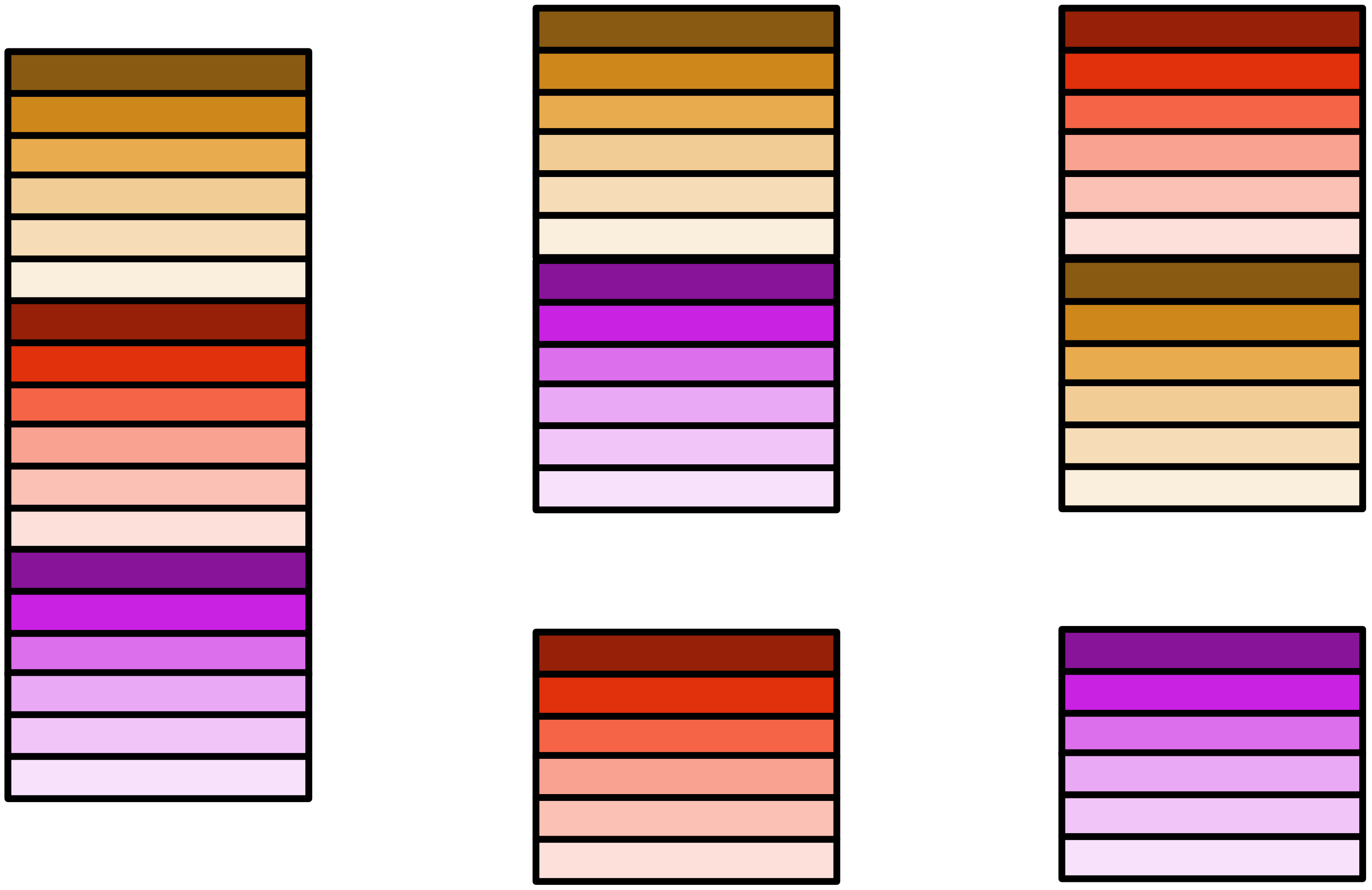
Multi-*fold* Cross Validation



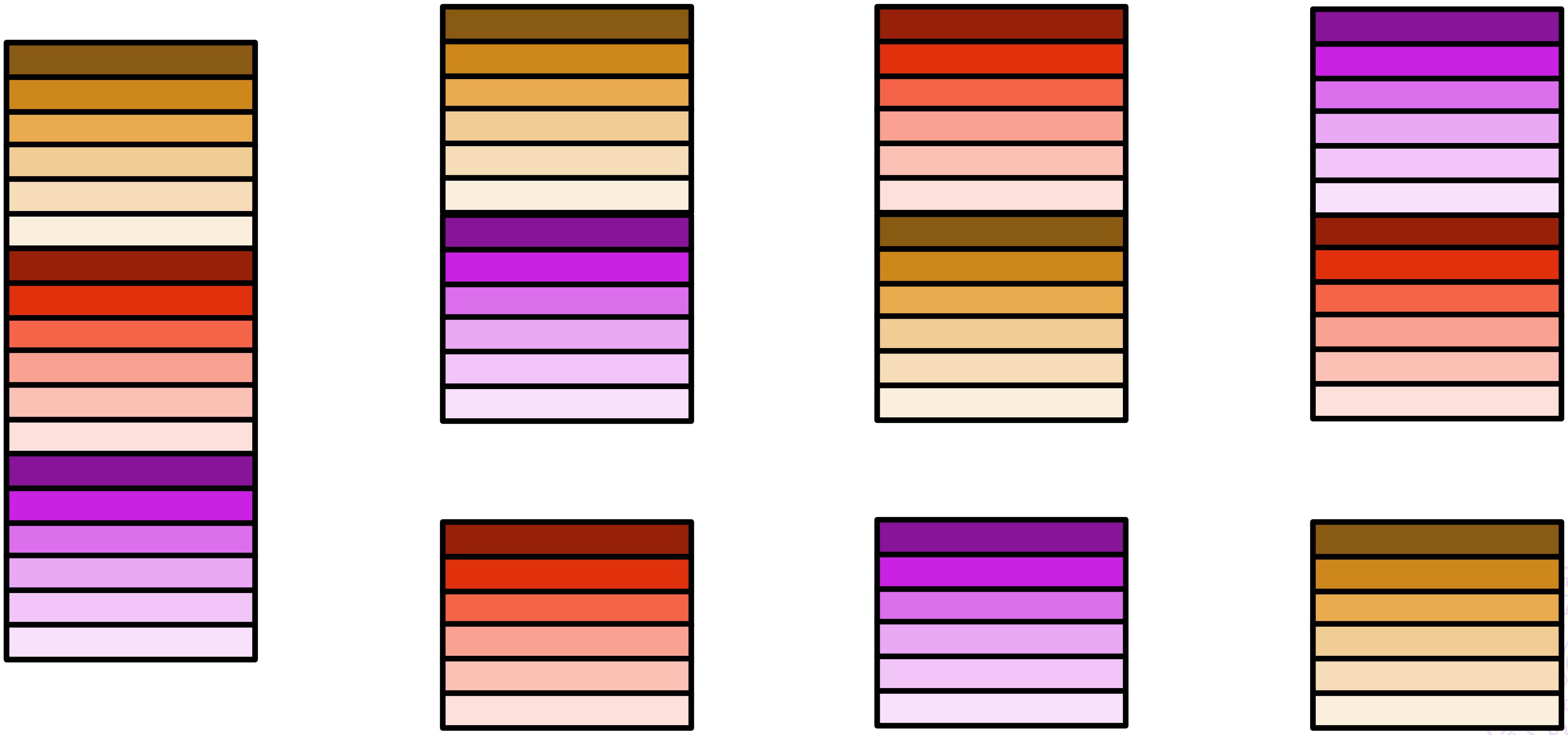
Multi-*fold* Cross Validation



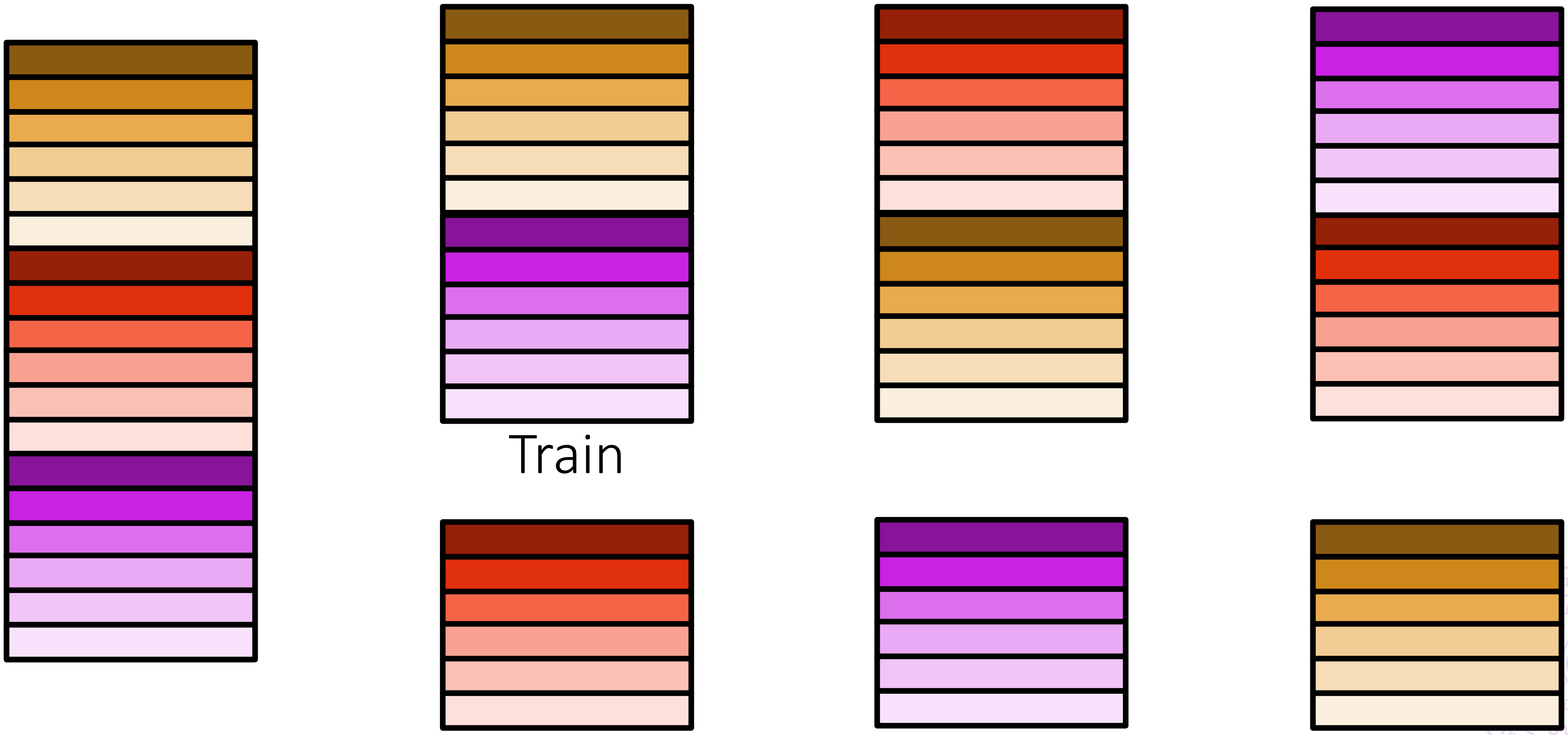
Multi-*fold* Cross Validation



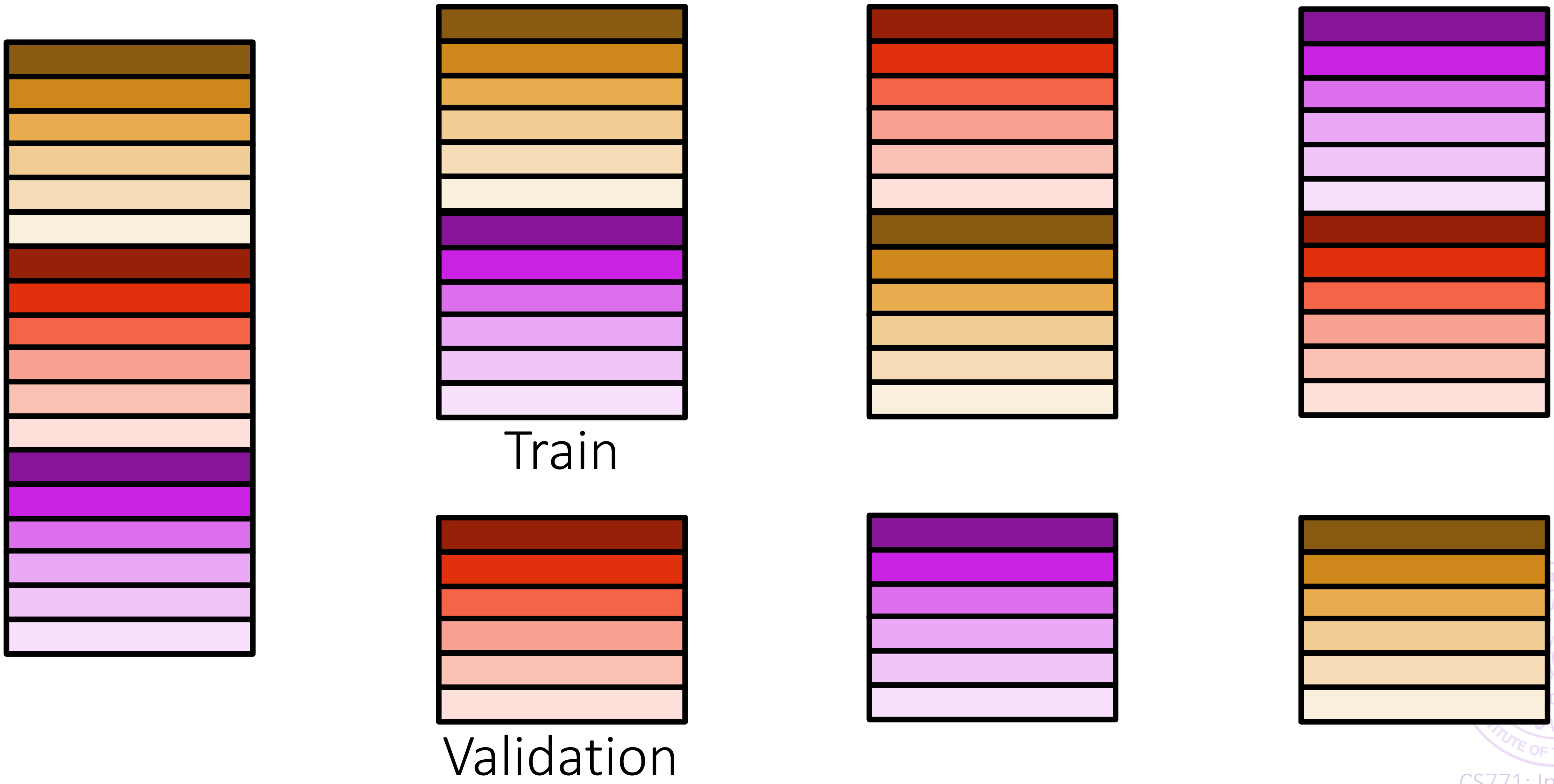
Multi-*fold* Cross Validation



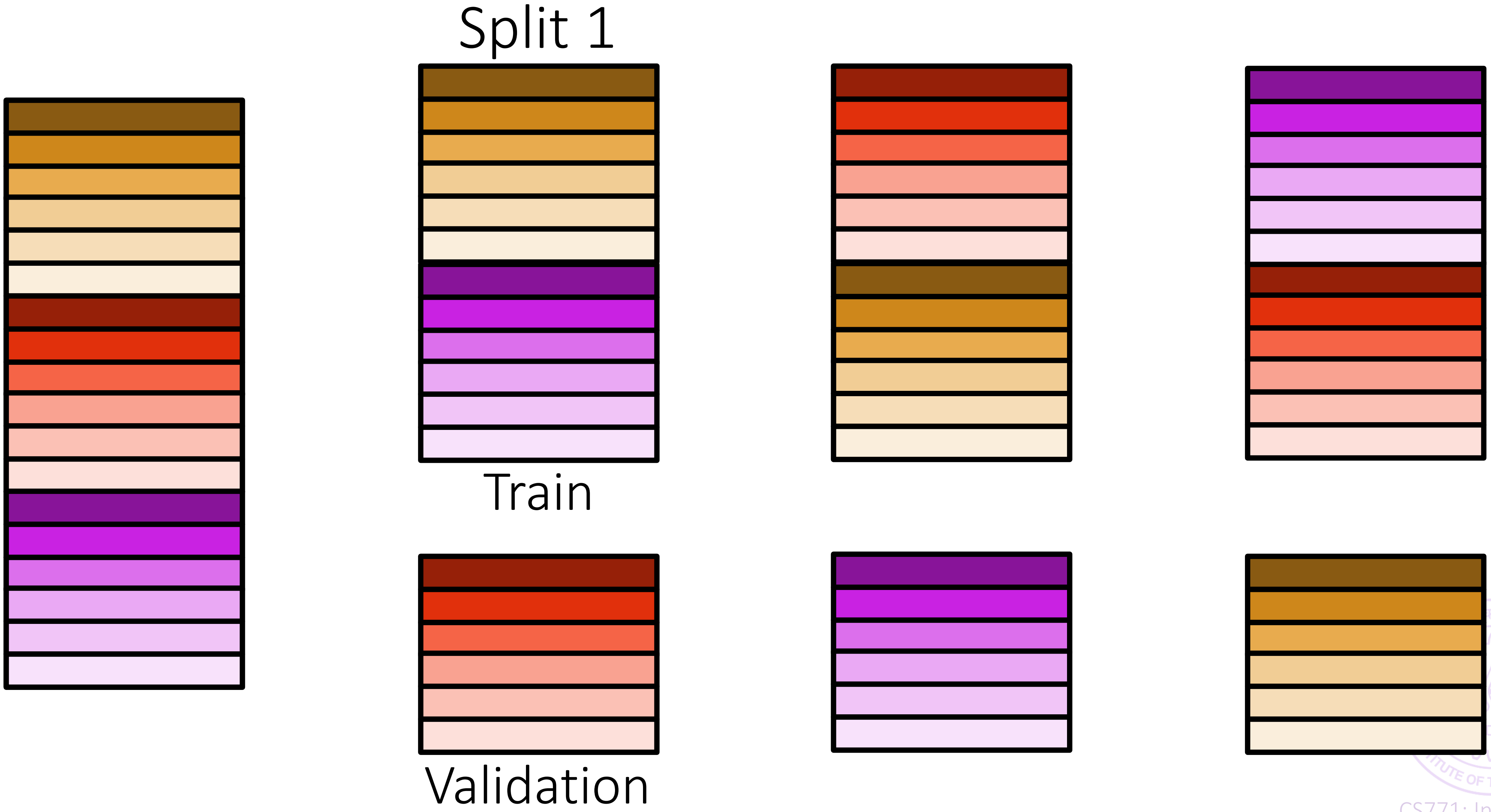
Multi-*fold* Cross Validation



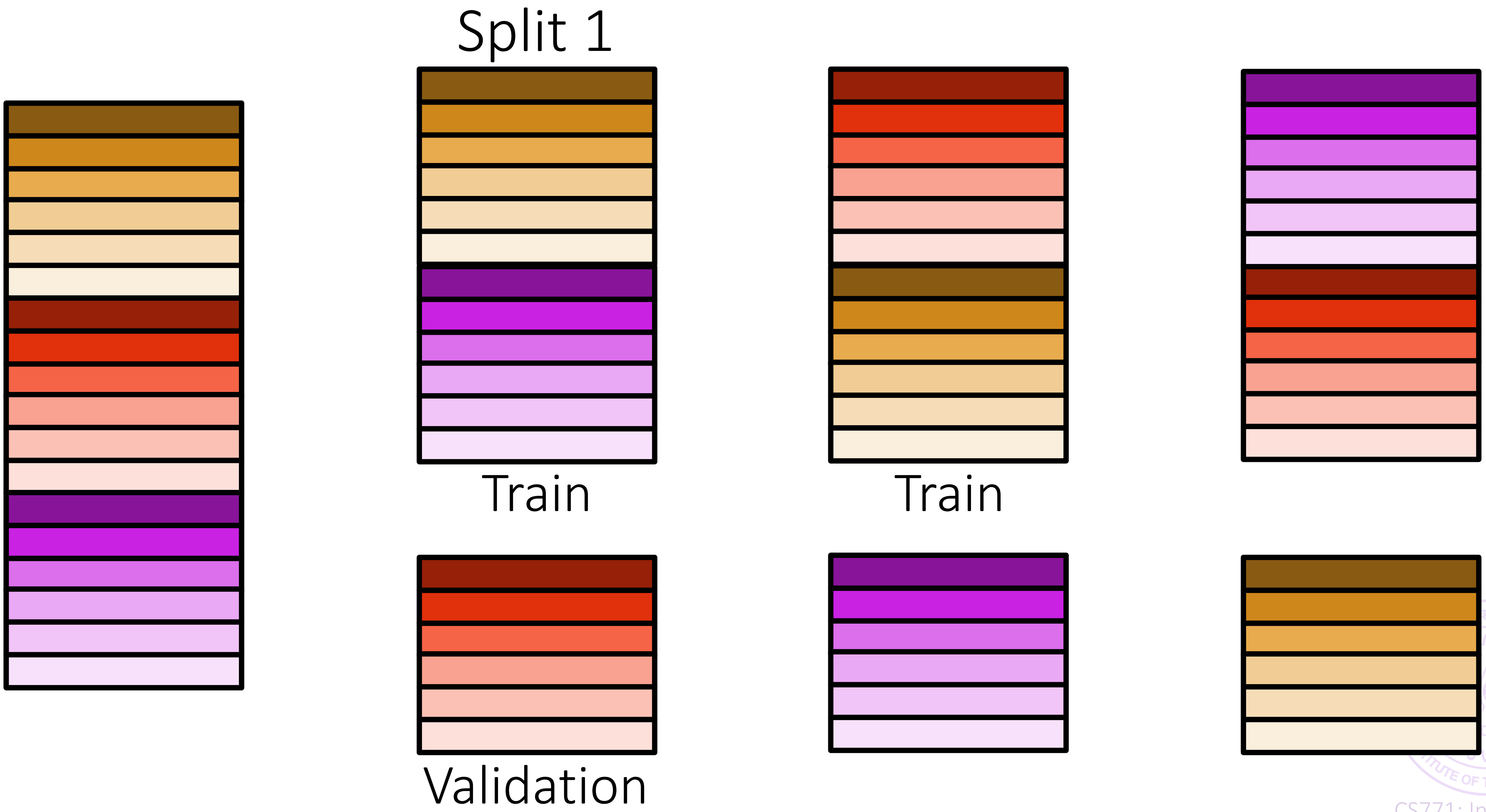
Multi-*fold* Cross Validation



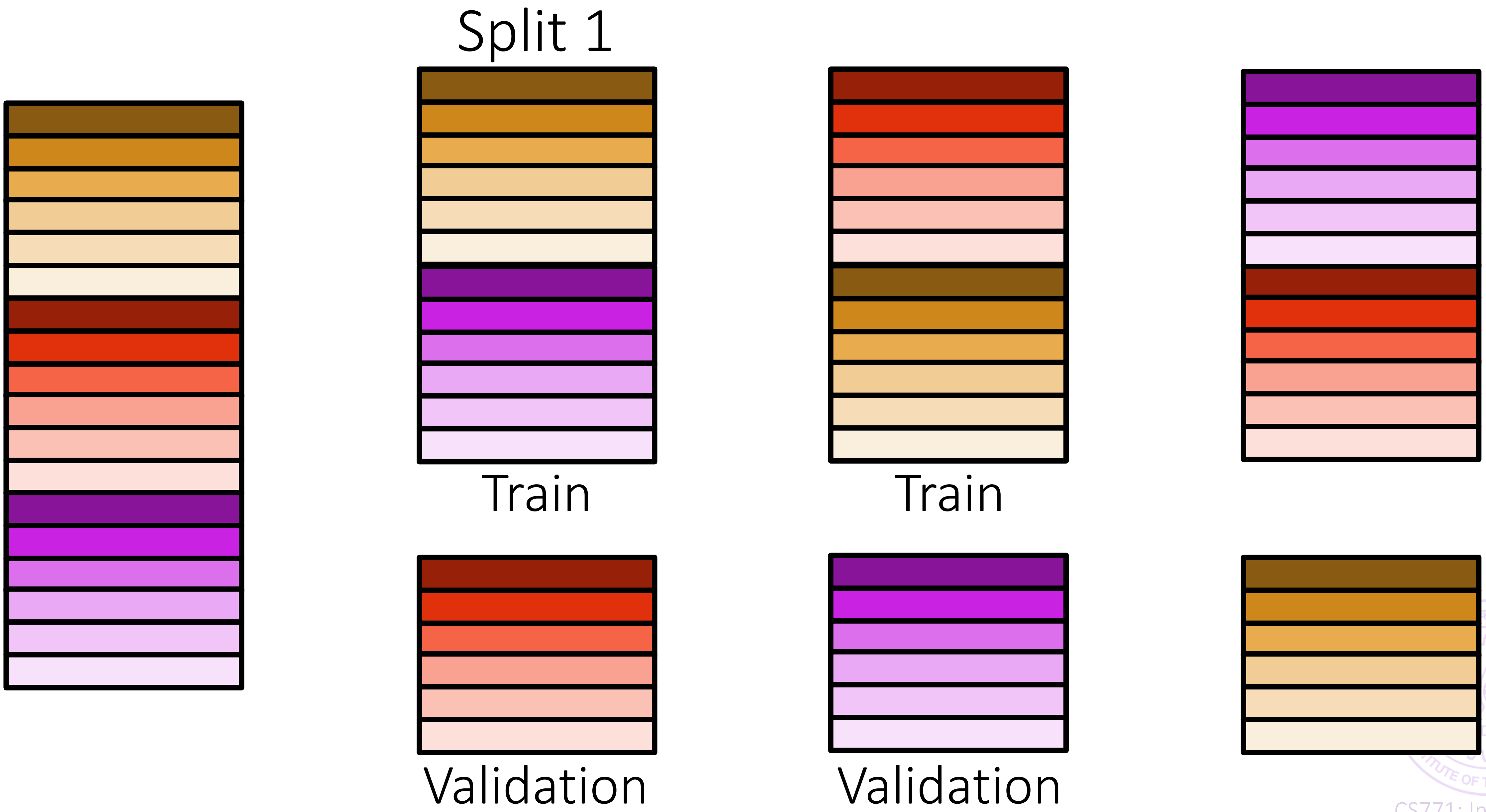
Multi-*fold* Cross Validation



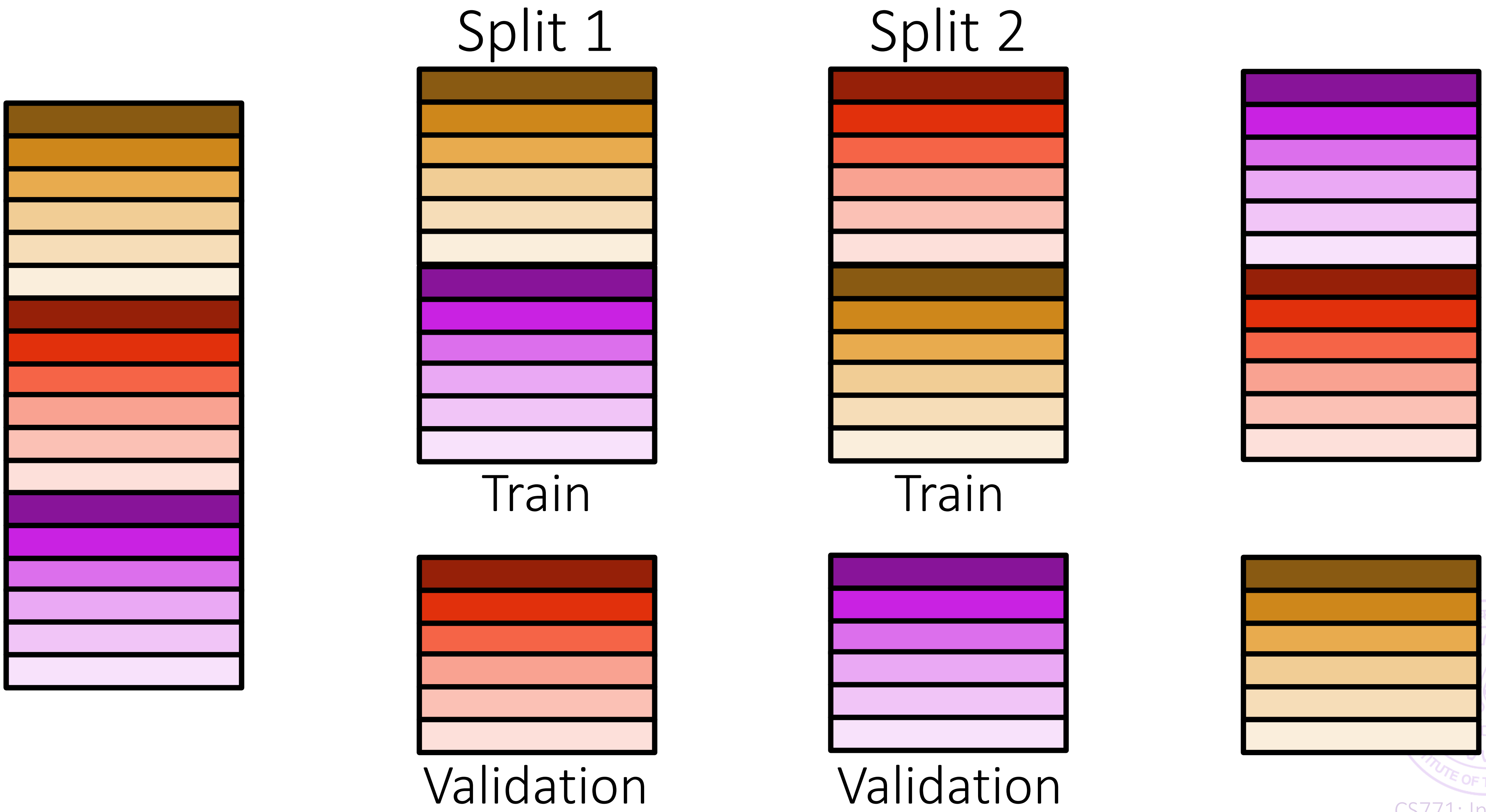
Multi-*fold* Cross Validation



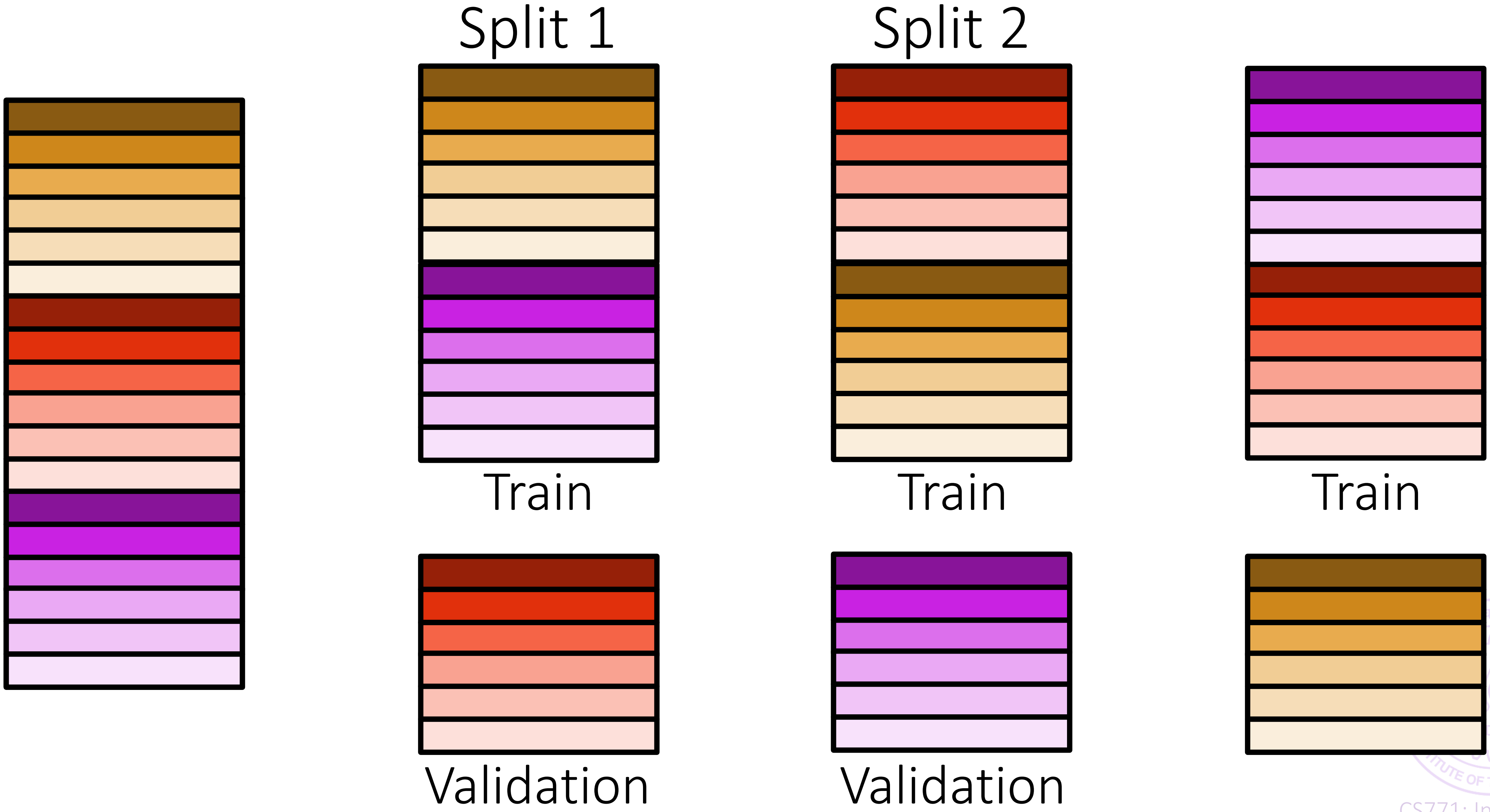
Multi-*fold* Cross Validation



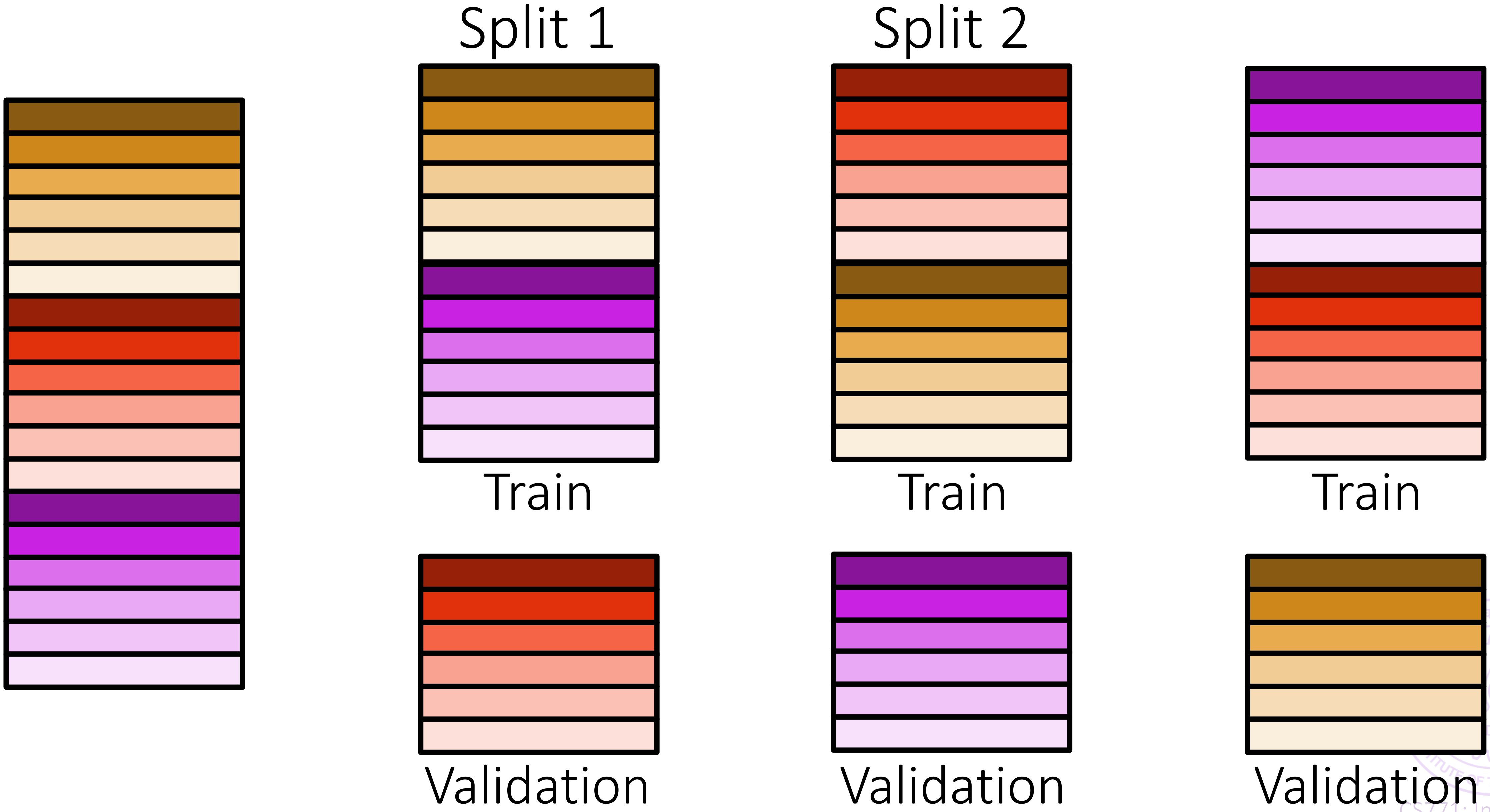
Multi-*fold* Cross Validation



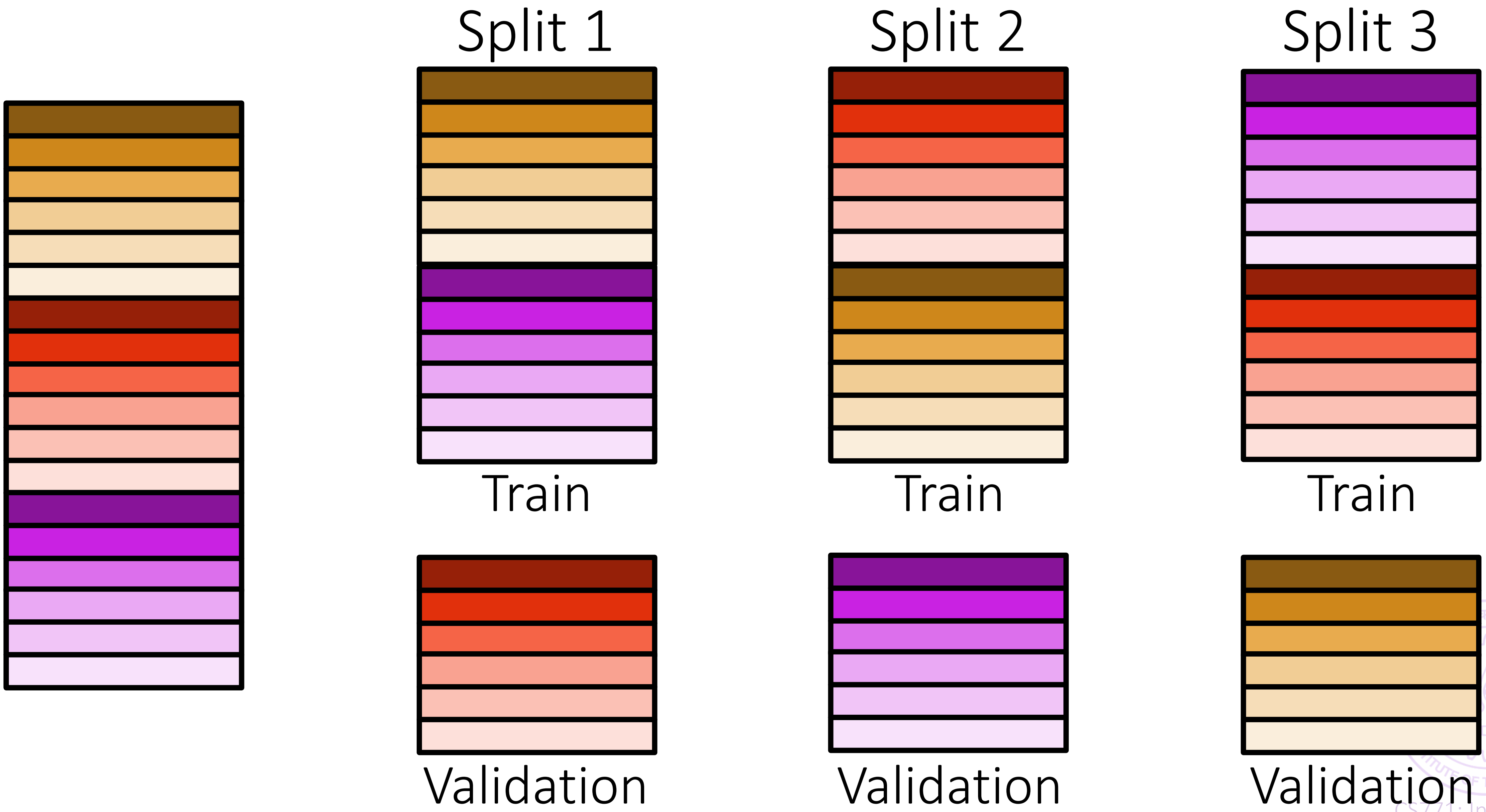
Multi-*fold* Cross Validation



Multi-*fold* Cross Validation



Multi-*fold* Cross Validation



Learning with NN – Lessons

One of the oldest learning algorithms - Fix and Hodges (1951)

Very intuitive, in fact – theoretically, it is the best algorithm possible

In practice it performs well if there is lots and lots of training data

However, not used directly since it takes a lot of time to make a prediction on new test point (finding nearest neighbour expensive)

Instead, clever ways used to speed up calculation of nearest neighbor

Learning with NN – Lessons

One of the oldest learning algorithms - Fix and Hodges (1951)

Very intuitive, in fact – theoretically, it is the best algorithm possible

In practice it performs well if there is lots and lots of training data

However, not used directly since it takes a lot of time to make a prediction on new test point (finding nearest neighbour expensive)

Instead, clever ways used to speed up calculation of nearest neighbor

Learning with NN – Lessons

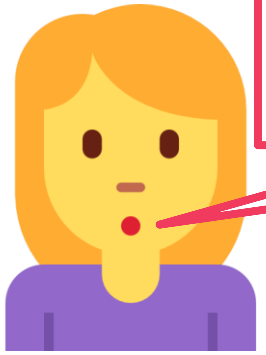
One of the oldest learning algorithms - Fix and Hodges (1951)

Very intuitive, in fact – theoretically, it is the best algorithm possible

In practice it performs well if there is lots and lots of training data

However, not used directly since it takes a lot of time to make a prediction on new test point (finding nearest neighbour expensive)

Instead, clever ways used to speed up calculation of nearest neighbor



Makes sense. If there are 2M training points, each a 10K-dim vector, then naively finding nearest neighbor takes 20B operations i.e. ~20 seconds @ 2GHz

Learning with NN – Lessons

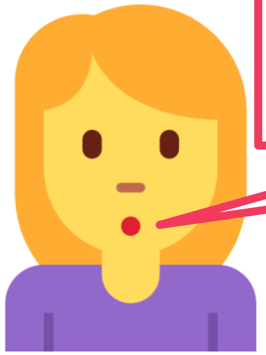
One of the oldest learning algorithms - Fix and Hodges (1951)

Very intuitive, in fact – theoretically, it is the best algorithm possible

In practice it performs well if there is lots and lots of training data

However, not used directly since it takes a lot of time to make a prediction on new test point (finding nearest neighbour expensive)

Instead, clever ways used to speed up calculation of nearest neighbor



Makes sense. If there are 2M training points, each a 10K-dim vector, then naively finding nearest neighbor takes 20B operations i.e. ~20 seconds @ 2GHz

Learning with NN – Lessons

One of the oldest learning algorithms - Fix and Hodges (1951)

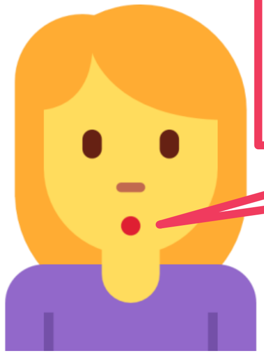
Very intuitive, in fact – theoretically, it is the best algorithm possible

In practice it performs well if there is lots and lots of training data

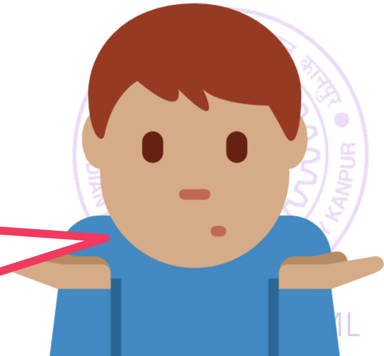
However, not used directly since it takes a lot of time to make a prediction on new test point (finding nearest neighbour expensive)

Instead, clever ways used to speed up calculation of nearest neighbor

Makes sense. If there are 2M training points, each a 10K-dim vector, then naively finding nearest neighbor takes 20B operations i.e. ~20 seconds @ 2GHz



Imagine if a bank website took 20 seconds to verify if a credit transaction is valid!



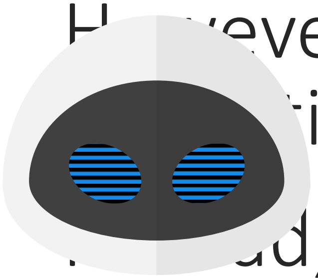
Learning with NN – Lessons

One of the oldest learning algorithms - Fix and Hodges (1951)

Very intuitive, in fact – theoretically, it is the best algorithm possible

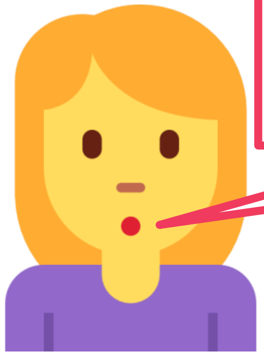
In practice it performs well if there is lots and lots of training data

H ever, not used directly since it takes a lot of time to make a
tion on new test point (finding nearest neighbour expensive)

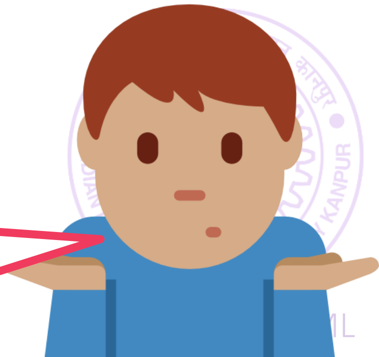


ad, clever ways used to speed up calculation of nearest neighbor

Makes sense. If there are 2M training points, each a 10K-dim vector, then naively finding nearest neighbor takes 20B operations i.e. ~20 seconds @ 2GHz



Imagine if a bank website took 20 seconds to verify if a credit transaction is valid!



Learning with NN – Lessons

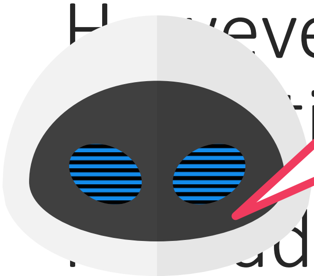
One of the oldest learning algorithms - Fix and Hodges (1951)

Very in… t is the best algorithm possible

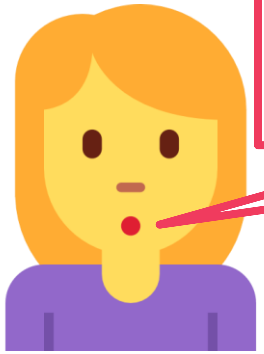
In prac… lots and lots of training data

However, n… ed directly since it takes a lot of time to make a …tic… on new test point (finding nearest neighbour expensive)

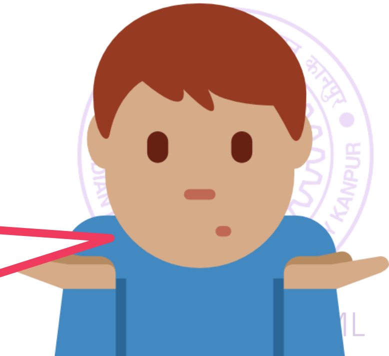
…d, clever ways used to speed up calculation of nearest neighbor



Also notice, with NN, all training need to be stored. In LwP, I could throw away training points – lightweight!



Makes sense. If there are 2M training points, each a 10K-dim vector, then naively finding nearest neighbor takes 20B operations i.e. ~20 seconds @ 2GHz



Imagine if a bank website took 20 seconds to verify if a credit transaction is valid!

Learning with NN – Lessons

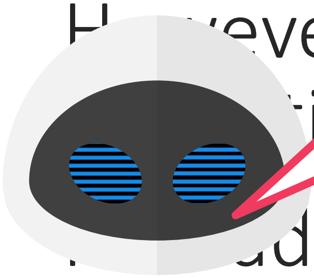
One of the oldest learning algorithms - Fix and Hodges (1951)

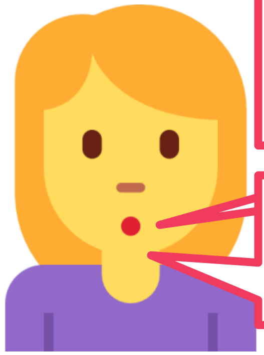
Very in… t is the best algorithm possible

In prac… lots and lots of training data

H…ever, n… …ed directly since it takes a lot of time to make a …tio… on new test point (finding nearest neighbour expensive)

…d, clever ways used to speed up calculation of nearest neighbor





Learning with NN – Lessons

One of the oldest learning algorithms - Fix and Hodges (1951)

Very in … is the best algorithm possible

In prac … lots and lots of training data

H…ever, n… …ed directly since it takes a lot of time to make a

…ti… on new test point (finding nearest neighbour expensive)

…d, clever ways used to speed up calculation of nearest neighbor

Also notice, with NN, all training need to be stored. In LwP, I could throw away training points – lightweight!

Makes sense. If there are 2M training points, each a 10K-dim vector, then naively finding nearest neighbor takes 20B operations i.e. ~20 seconds @ 2GHz

Wait! So what is the “model” for NN?

Imagine if a bank website took 20 seconds to verify if a credit transaction is valid!

Learning with NN – Lessons

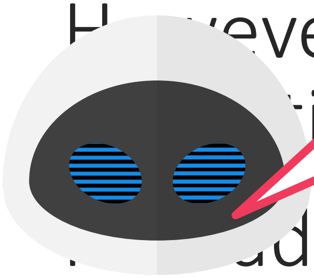
One of the oldest learning algorithms - Fix and Hodges (1951)

Very in[...] is the best algorithm possible

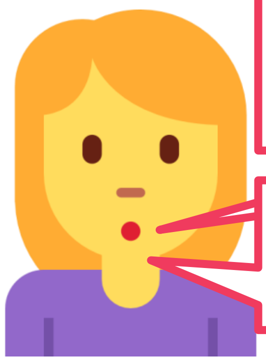
In prac[...] lots and lots of training data

H[...]ever, n[...]ed directly since it takes a lot of time to make a [...]ti[...]on new test point (finding nearest neighbour expensive)

[...]d, clever ways used to speed up calculation of nearest neighbor

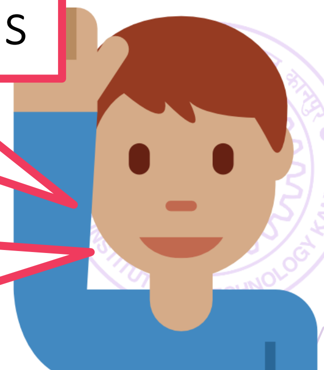


Also notice, with NN, all training need to be stored. In LwP, I could throw away training points – lightweight!



Makes sense. If there are 2M training points, each a 10K-dim vector, then naively finding nearest neighbor takes 20B operations i.e. ~20 seconds @ 2GHz

Wait! So what is the “model” for NN?



The entire training set is the model – NNs have huge models

Imagine if a bank website took 20 seconds to verify if a credit transaction is valid!

Learning with NN – Lessons

One of the oldest learning algorithms - Fix and Hodges (1951)

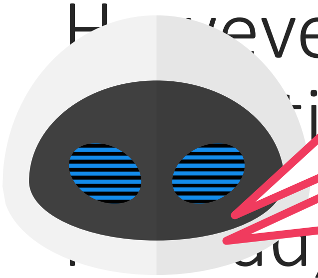
Very in… t is the best algorithm possible

In prac… lots and lots of training data

However, … a lot of time to make a
…rest neighbour expensive)

…ad, cl… lculation of nearest neighbor

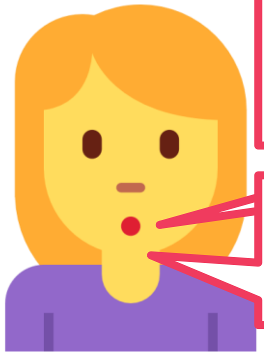
Also notice, with NN, all training need to be stored. In LwP, I could throw away training points – lightweight!



Correct! NN requires huge storage too! Note that the model size goes up with the amount of training data ☹

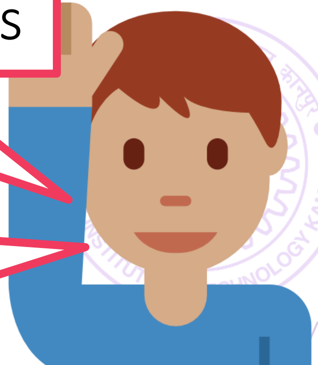
Makes sense. If there are 2M training points, each a 10K-dim vector, then naively finding nearest neighbor takes 20B operations i.e. ~20 seconds @ 2GHz

The entire training set is the model – NNs have huge models



Wait! So what is the "model" for NN?

Imagine if a bank website took 20 seconds to verify if a credit transaction is valid!



Learning with NN – Lessons

One of the oldest learning algorithms - Fix and Hodges (1951)

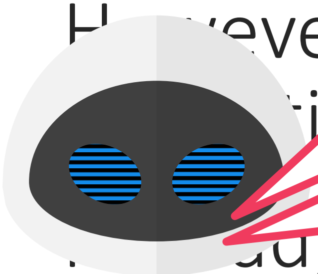
Very in… t is the best algorithm possible

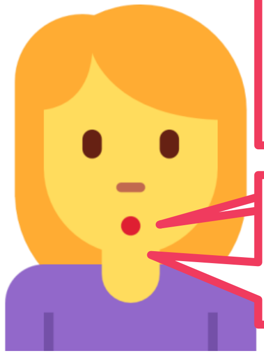
In prac… lots and lots of training data

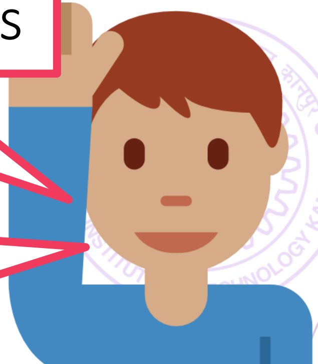
H…ever, … a lot of time to make a

…ti… …rest neighbour expens

…ad, c… …lculation of nearest nei







Learning with NN – Lessons

One of the oldest learning algorithms - Fix and Hodges (1951)

Very in… is the best algorithm possible

In prac… lots and lots of training data

Also notice, with NN, all training need to be stored. In LwP, I could throw away training points – lightweight!

Correct! NN requires huge storage too! Note that the model size goes up with the amount of training data ☹

Models whose size depends on the amount of training data are called *non-parametric* models

Makes sense. If there are 2M training points, each a 10K-dim vector, then naively finding nearest neighbor takes 20B operations i.e. ~20 seconds @ 2GHz

The entire training set is the model – NNs have huge models

Wait! So what is the “model” for NN?

Imagine if a bank website took 20 seconds to verify if a credit transaction is valid!

Learning with NN – Lessons

One of the oldest learning algorithms 1)

Very in possible

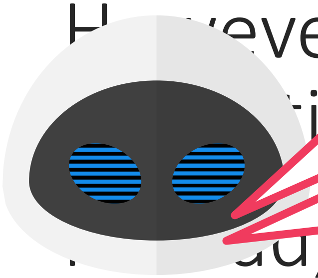
In prac data

However,

ud, cl

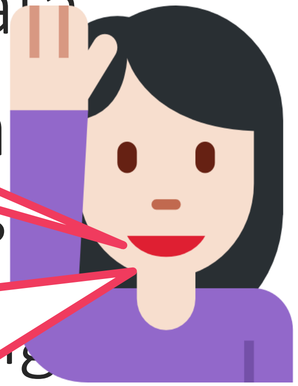
Also notice, with NN, all training need to be stored. In LwP, I could throw away training points – lightweight!

Compare this to LwP where model had 2 vectors no matter how many training points – such models are called *parametric models*



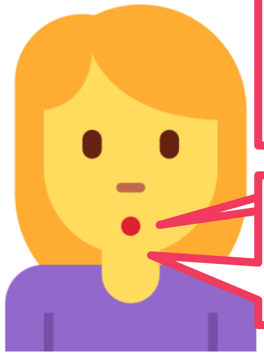
Correct! NN requires huge storage too! Note that the model size goes up with the amount of training data ☹

Models whose size depends on the amount of training data are called *non-parametric* models



Makes sense. If there are 2M training points, each a 10K-dim vector, then naively finding nearest neighbor takes 20B operations i.e. ~20 seconds @ 2GHz

The entire training set is the model – NNs have huge models



Wait! So what is the “model” for NN?

Imagine if a bank website took 20 seconds to verify if a credit transaction is valid!