Linear Classifiers

CS771: Introduction to Machine Learning

Purushottam Kar

Announcements

Please declare your assignment groups using the form below https://forms.gle/Zqe3yZyGv7rvzjm56

5 registered students per group (no auditors)

Deadline for filling this form August 09 (Friday), 11:59PM IST

Assignment 1 will be released soon thereafter and we will simply assign remaining students to arbitrary groups after above deadline

Groups that are not of proper size (5) may be reorganized by us

Please fill the form only once per project group

Auditor groups should not fill this form

Project groups will be listed on the website

Recap of Last Lecture

- Looked at a *weighted* variant of the NN algorithm
- Studied decision trees which have several benefits
  - Can be thought of as a speed up way to perform NN classification
  - Offer extremely fast prediction times
  - Can handle non-numeric, discrete, categorical features with ease
- Some challenges with decision trees
  - Can be bulky i.e. model size can be large
  - Deciding how to split a node can be time consuming
  - Learning the *best* decision tree is an intractable (NP hard) problem

Decision Boundaries

Some interesting discussion on Piazza

Earlier definition of decision boundary (points where classifier gets confused is simple but not general enough)

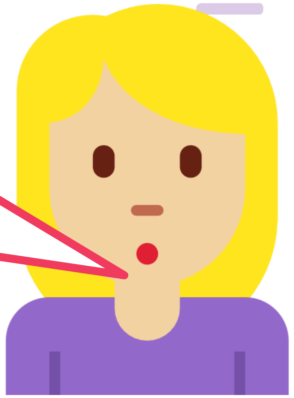
More robust definition of decision boundary: locations where classifier decision abruptly changes from one class to another class

All classifiers have such a decision boundary

Easy to detect whether a test point is at decision boundary for linear classifiers – difficult to do so for most other classifiers, e.g. deep nets

Decision Boundaries

Indeed, since we would have to not only predict for that data point, but also for other data points around it!



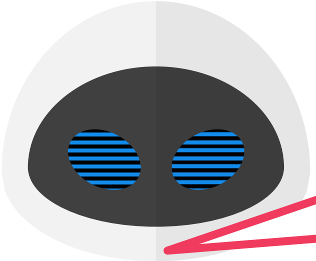
Some interesting discussion on Piazza

Earlier definition of decision boundary (points where classifier gets confused is simple but not general enough)

More robust definition of decision boundary: locations where classifier decision abruptly changes from one class to another class

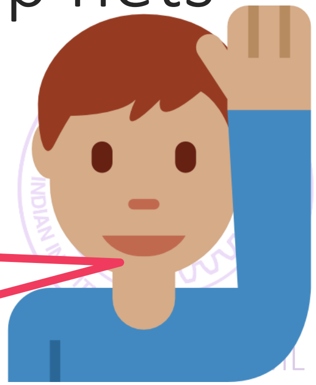
All classifiers have such a decision boundary

Easy to detect whether a test point is at decision boundary for linear classifiers – difficult to do so for most other classifiers, e.g. deep nets



It might still get confused if there are 4 points equally close 2 of them red and 2 green ☺

For example, kNN will never get confused if k = 3 (or some odd number) and 2 classes



Linear Classifiers

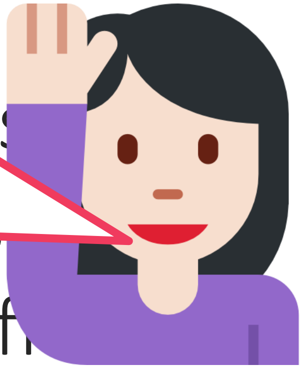
Keep appearing again and again in various methods
- LwP with 2 classes, Euclidean metric always gives a linear classifier
- Even if Mahalanobis metric used, still LwP gives a linear classifier
- Decision stumps with a single feature also give a linear classifier

Extremely popular in ML
- Very small model size – just one vector (and one bias value)
- Very fast $\mathcal{O}(d)$ prediction time
- Used to build DTs, deep nets etc

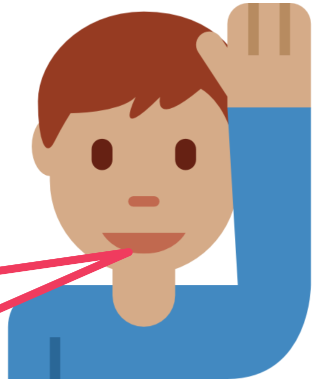
Linear Classifiers

Keep appearing again and ag
- LwP with 2 classes, Euclidea las
- Even if Mahalanobis metric ss
- Decision stumps with a single feature also give a linear classif

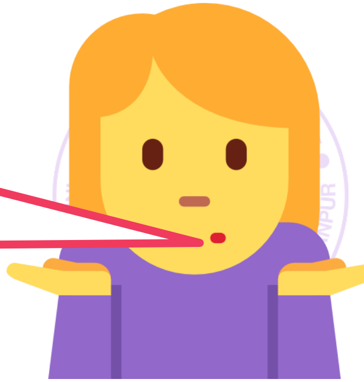
Extremely popular in ML
- Very small model size – just one vector (and one bias value)
- Very fast $\mathcal{O}(d)$ prediction time
- Used to build DTs, deep nets etc



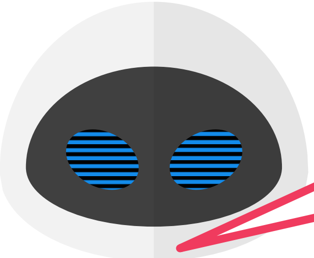
Before going forward, recall that linear classifiers are those that have a line or a plane as the decision boundary. A linear classifier is given by a model that looks like $(\mathbf{w}, b)$ and it makes predictions by looking at whether $\mathbf{w}^\top \mathbf{x} + b > 0$ or not



Learning classifiers directly will allow us to control many useful properties about them!



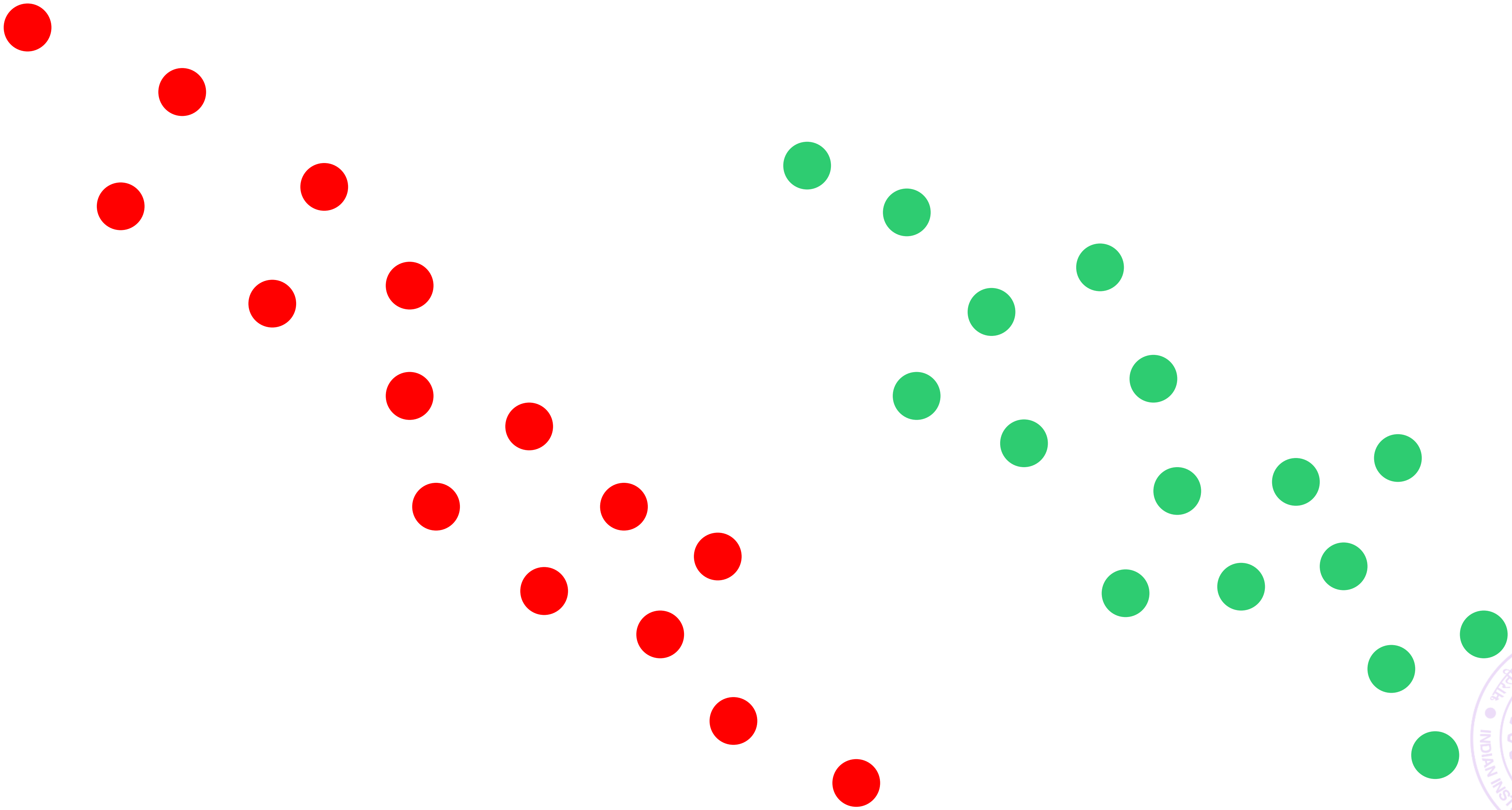
Instead of indirectly getting a linear classifier via LwP + Mahalanobis etc etc, can't we learn one directly?



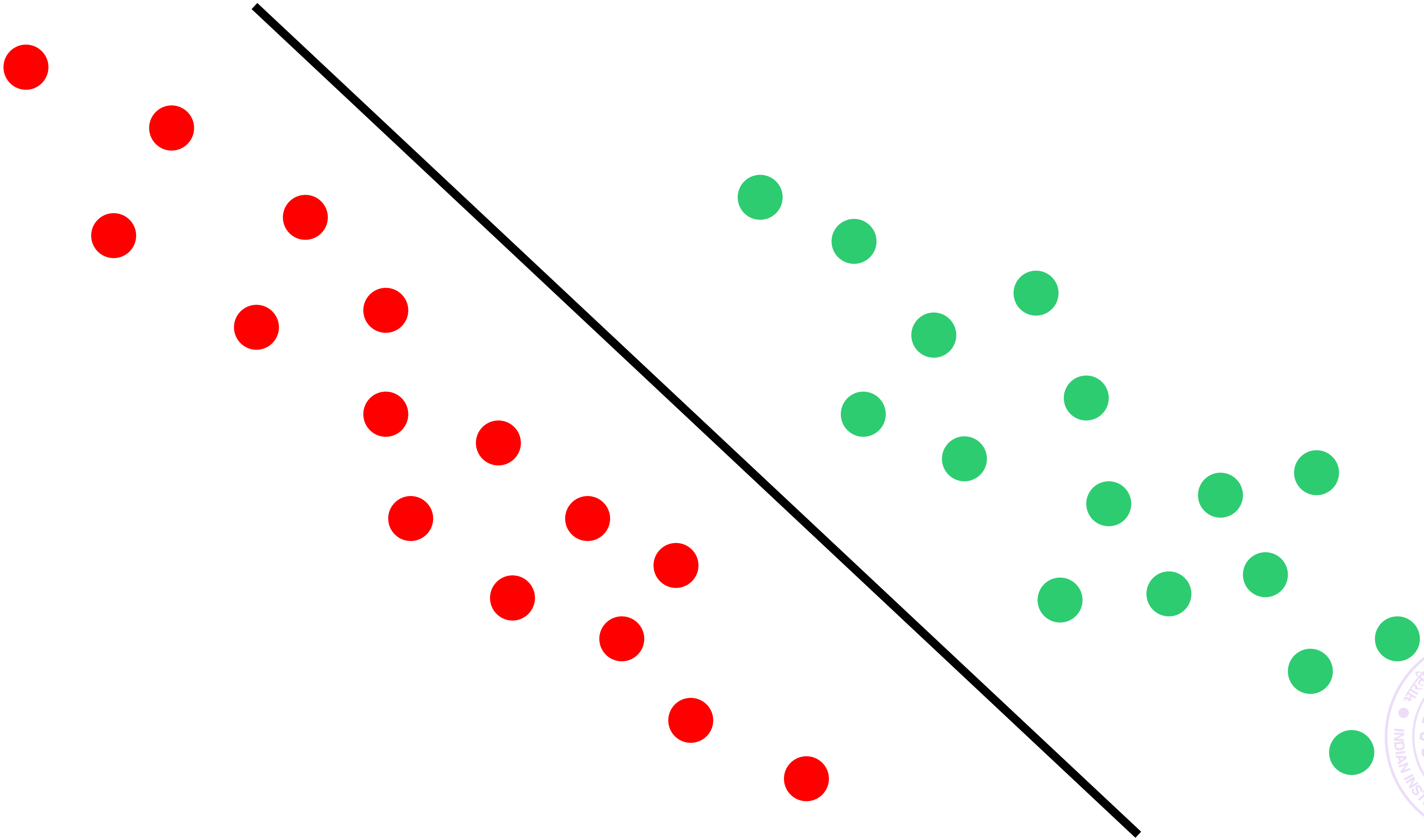
That is exactly what we will do today!

The “best” Linear Classifier

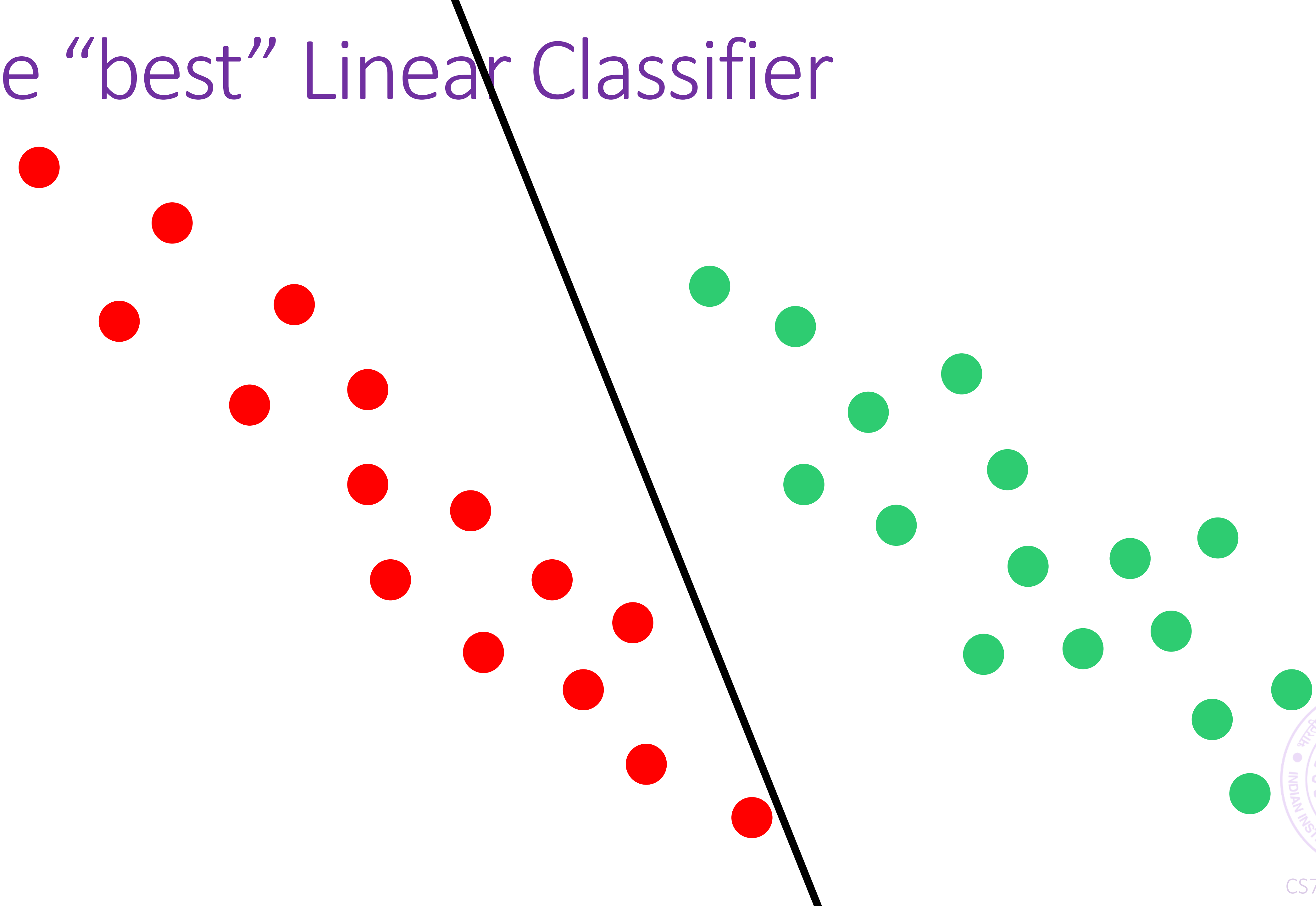
The “best” Linear Classifier



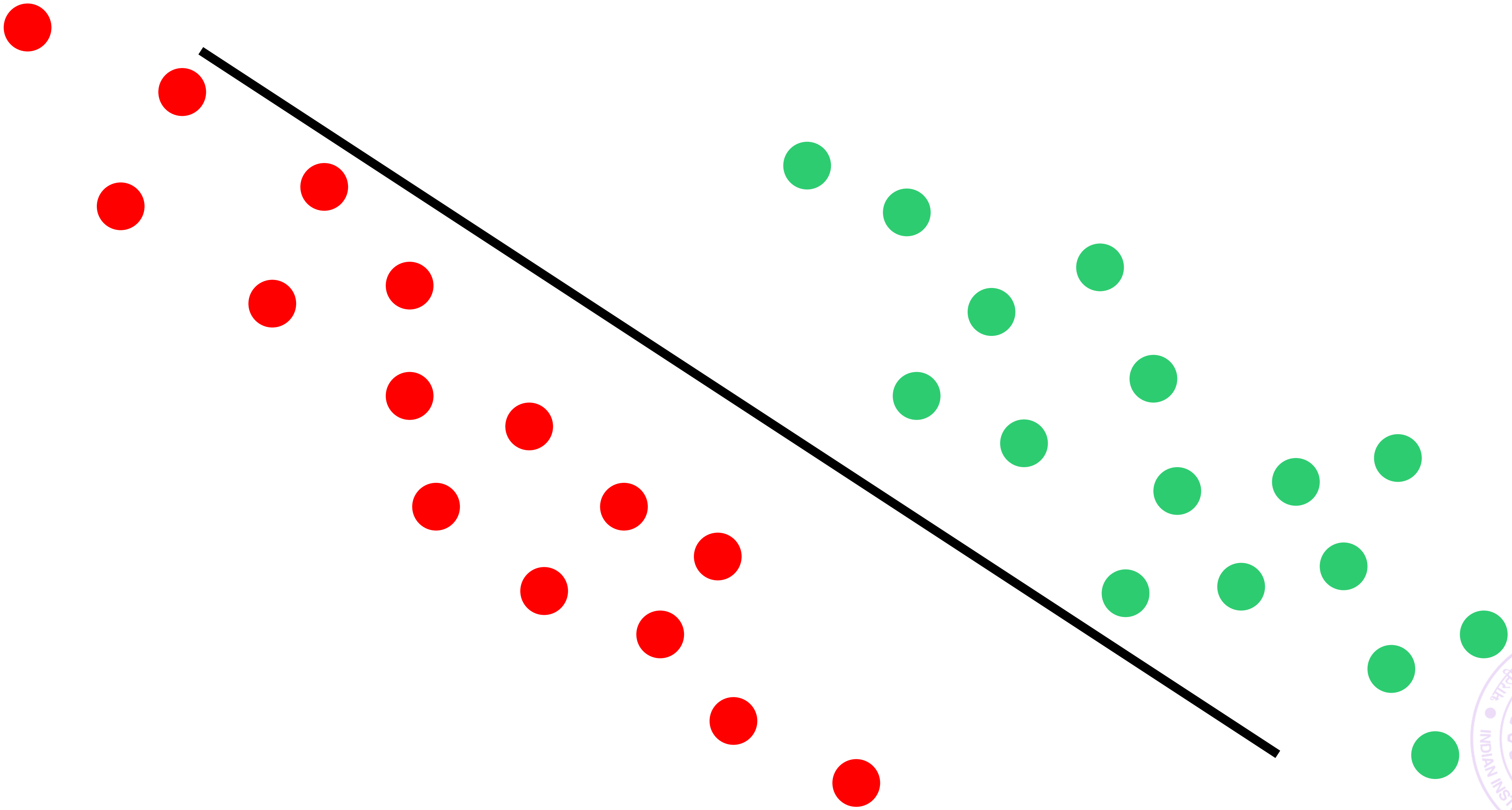
The “best” Linear Classifier



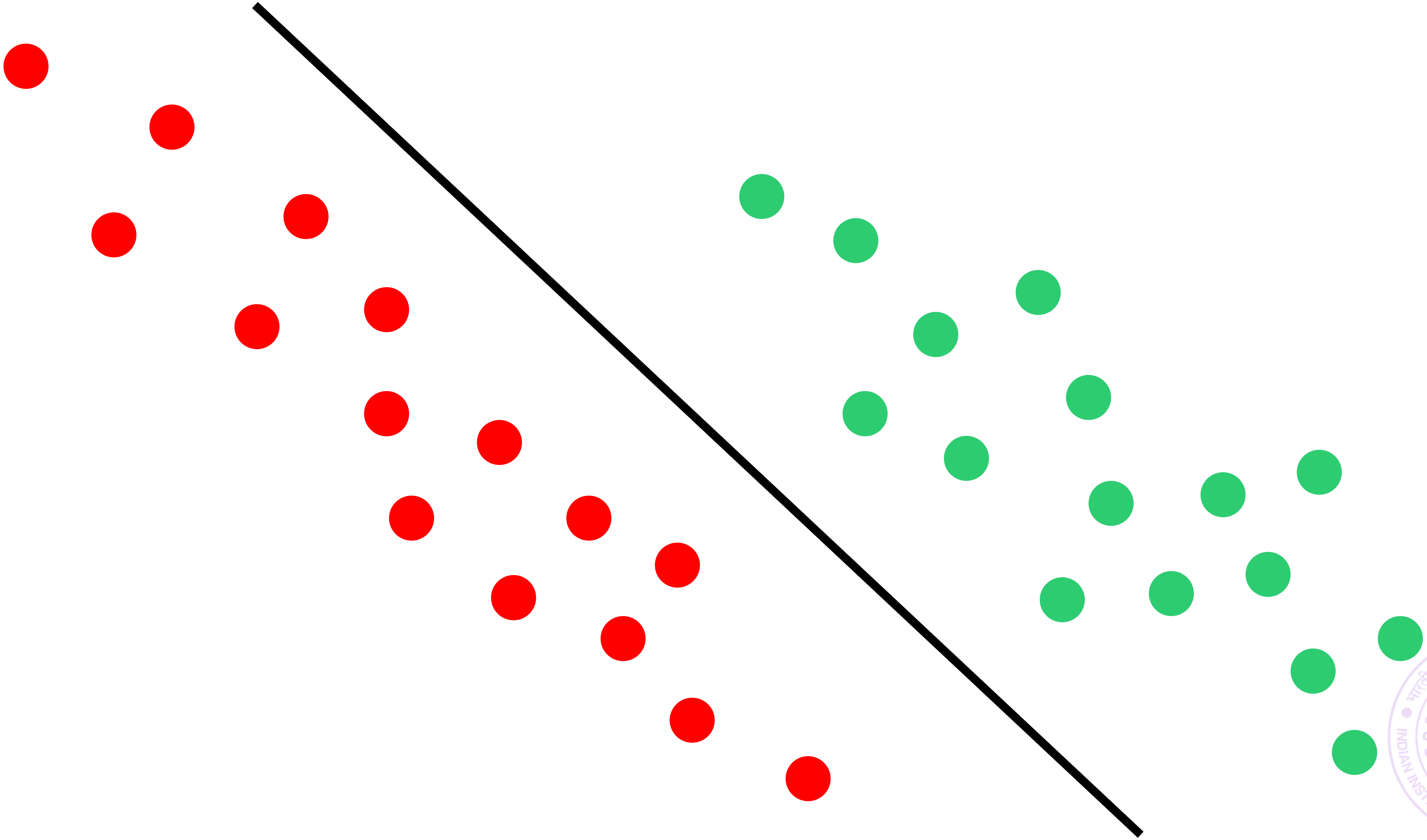
The "best" Linear Classifier



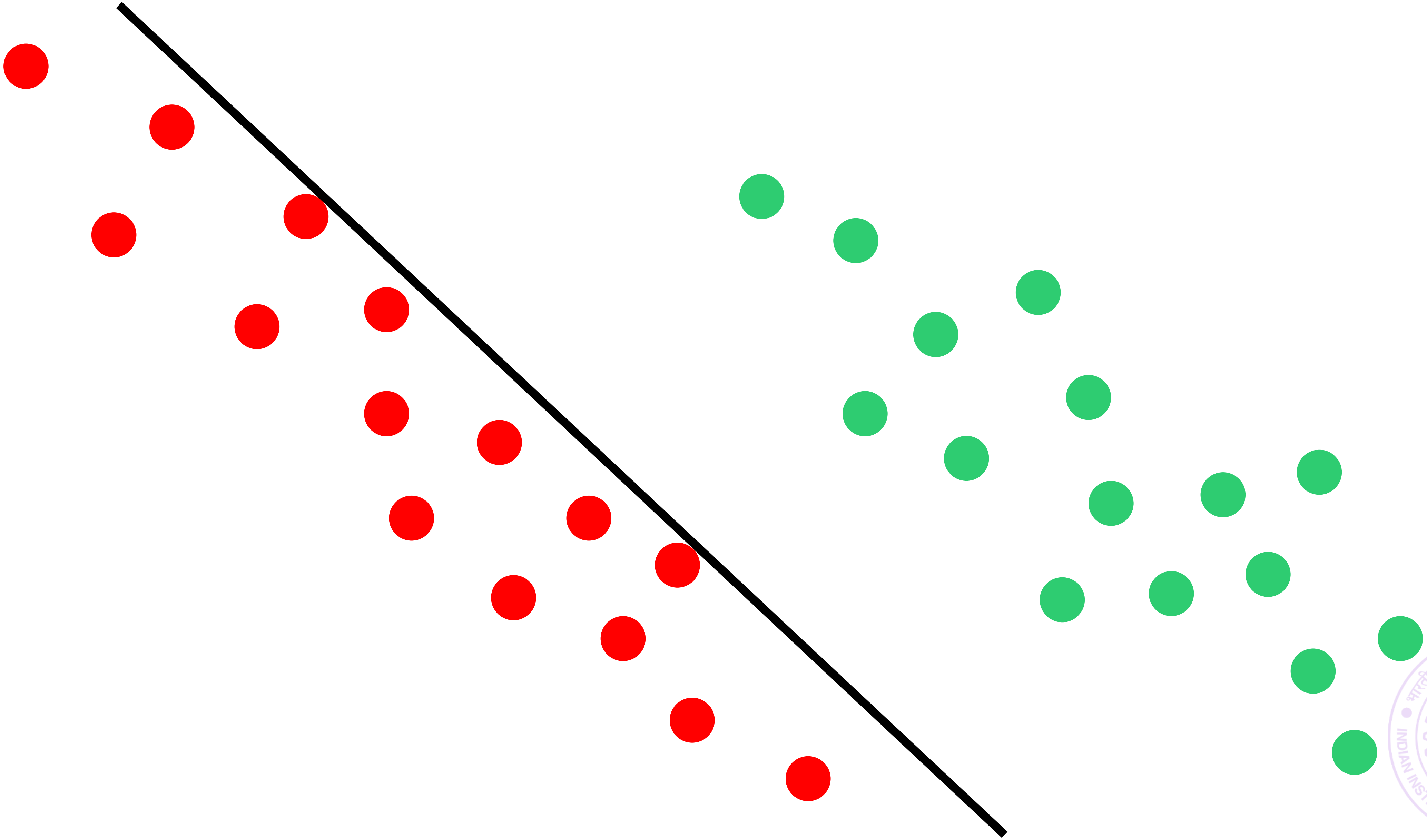
The “best” Linear Classifier



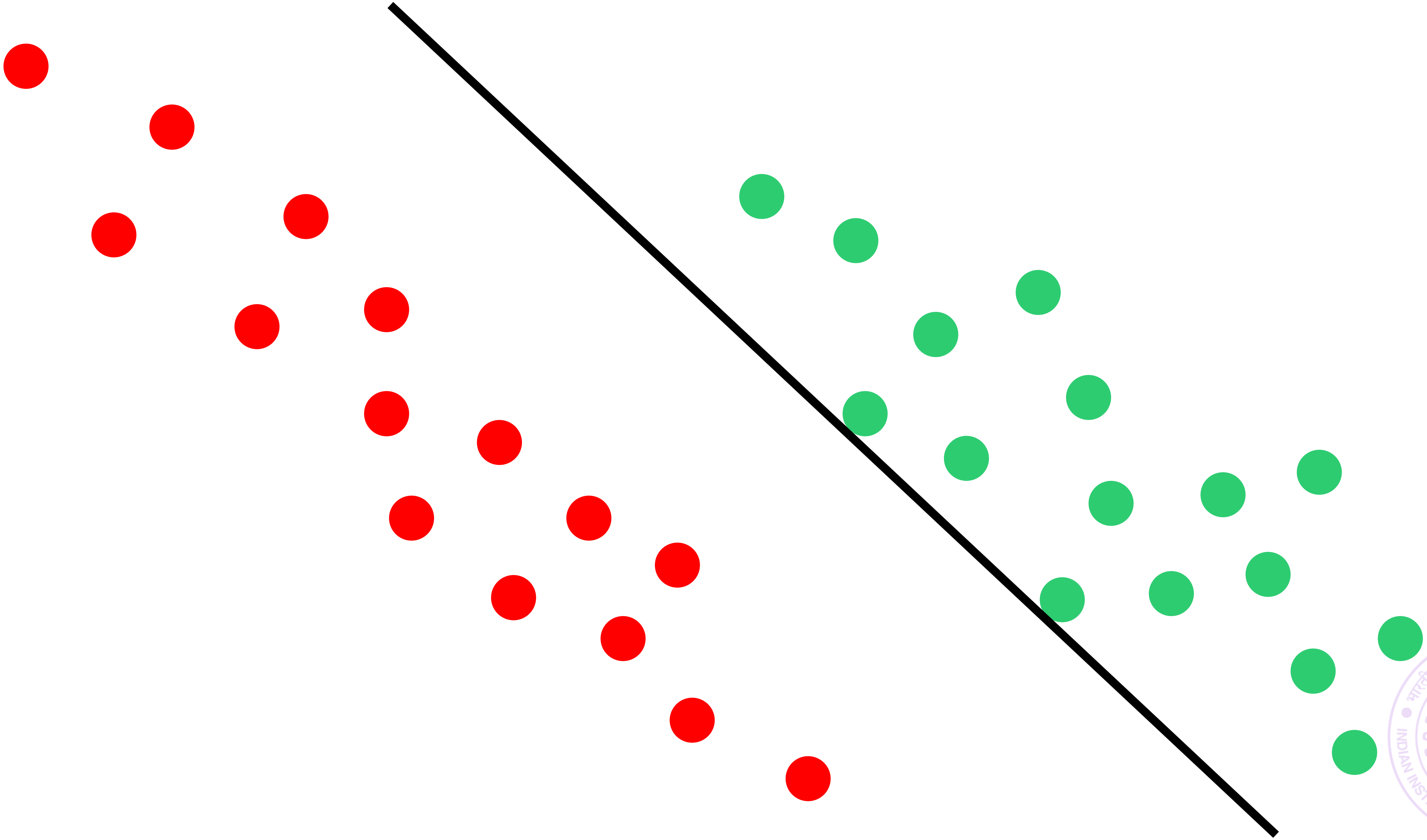
The “best” Linear Classifier



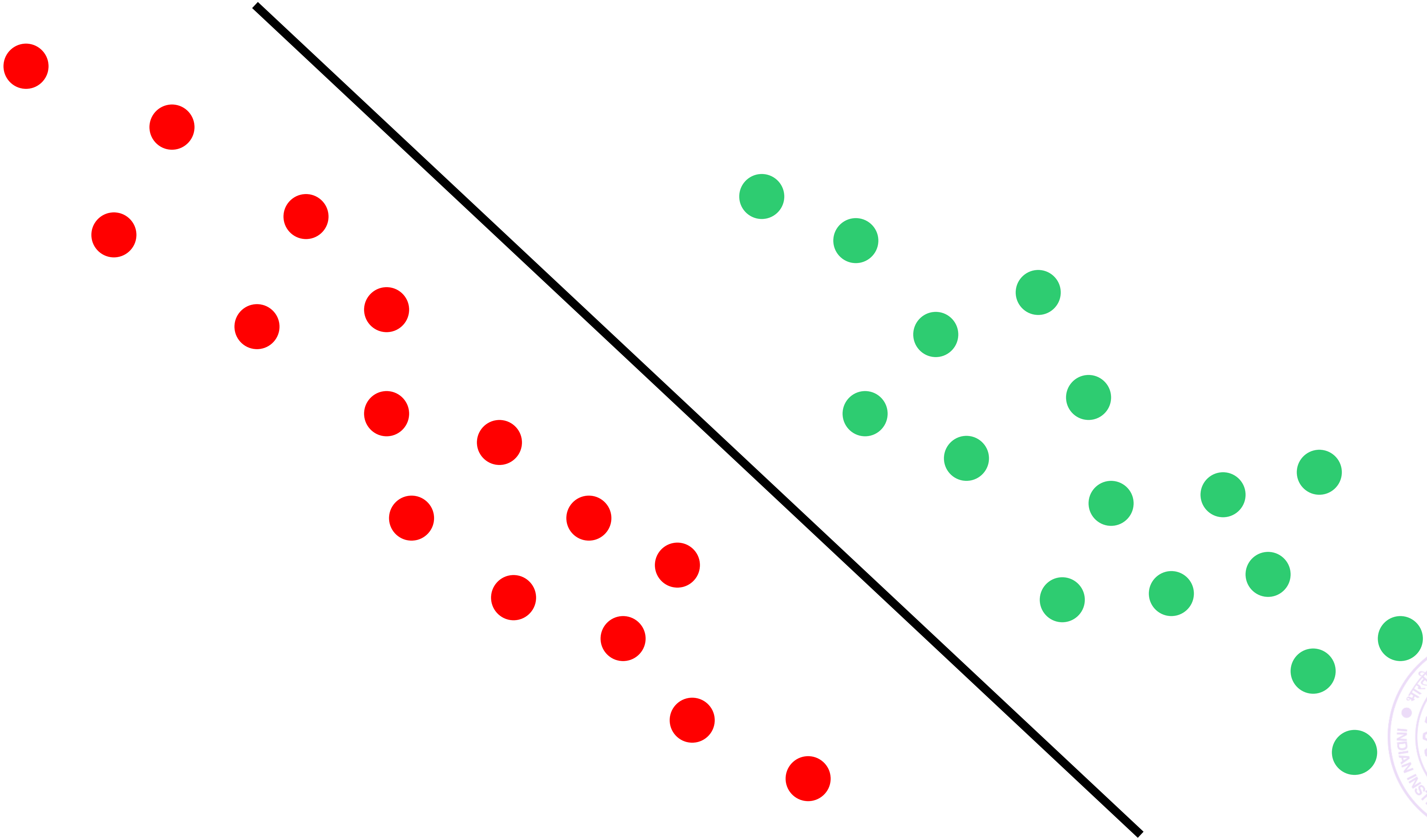
The "best" Linear Classifier



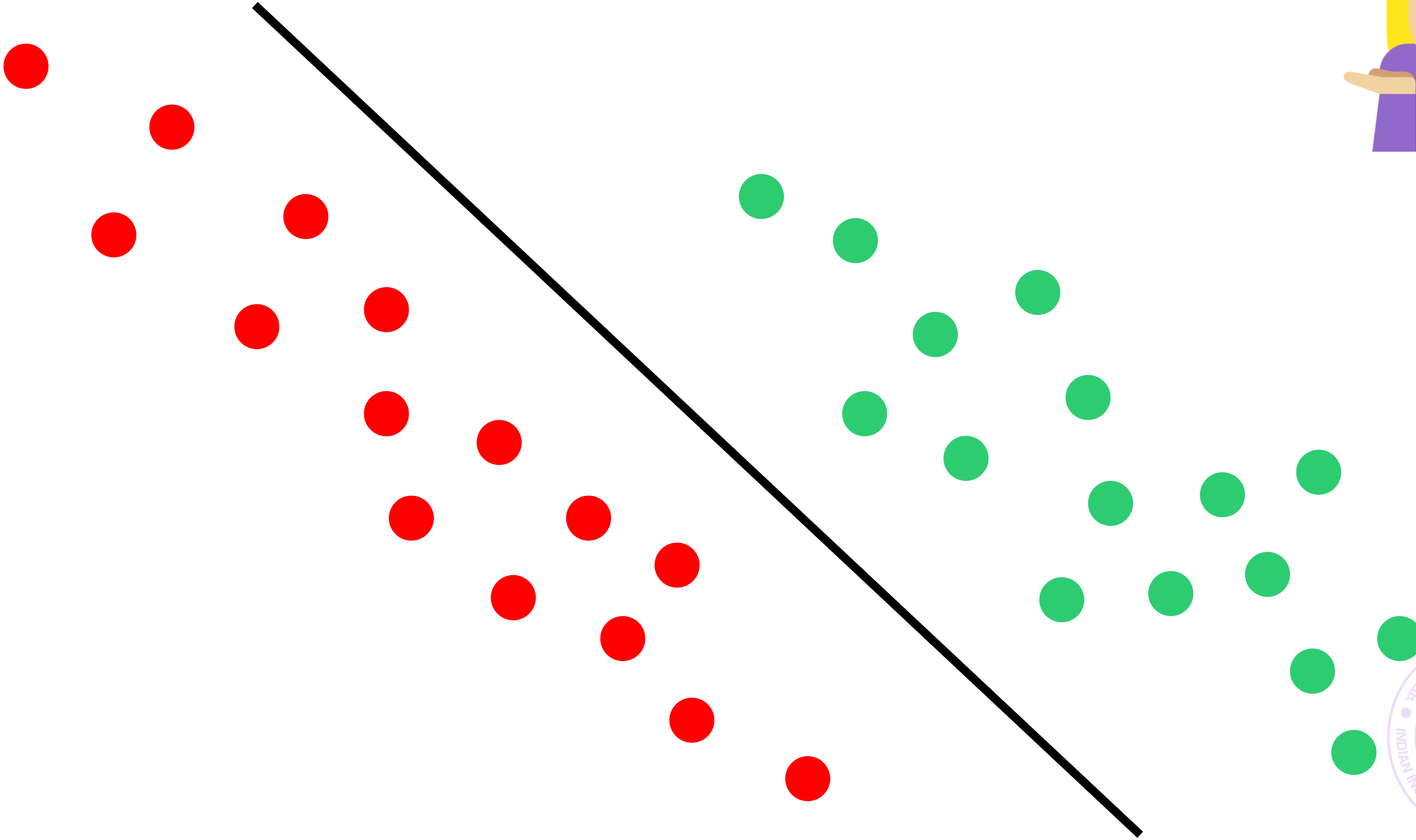
The “best” Linear Classifier



The “best” Linear Classifier

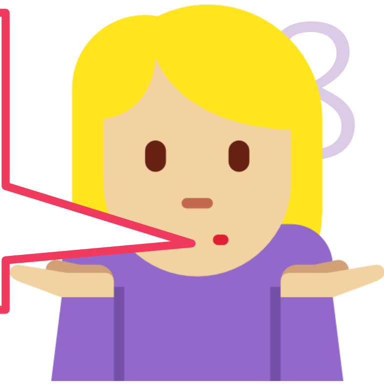


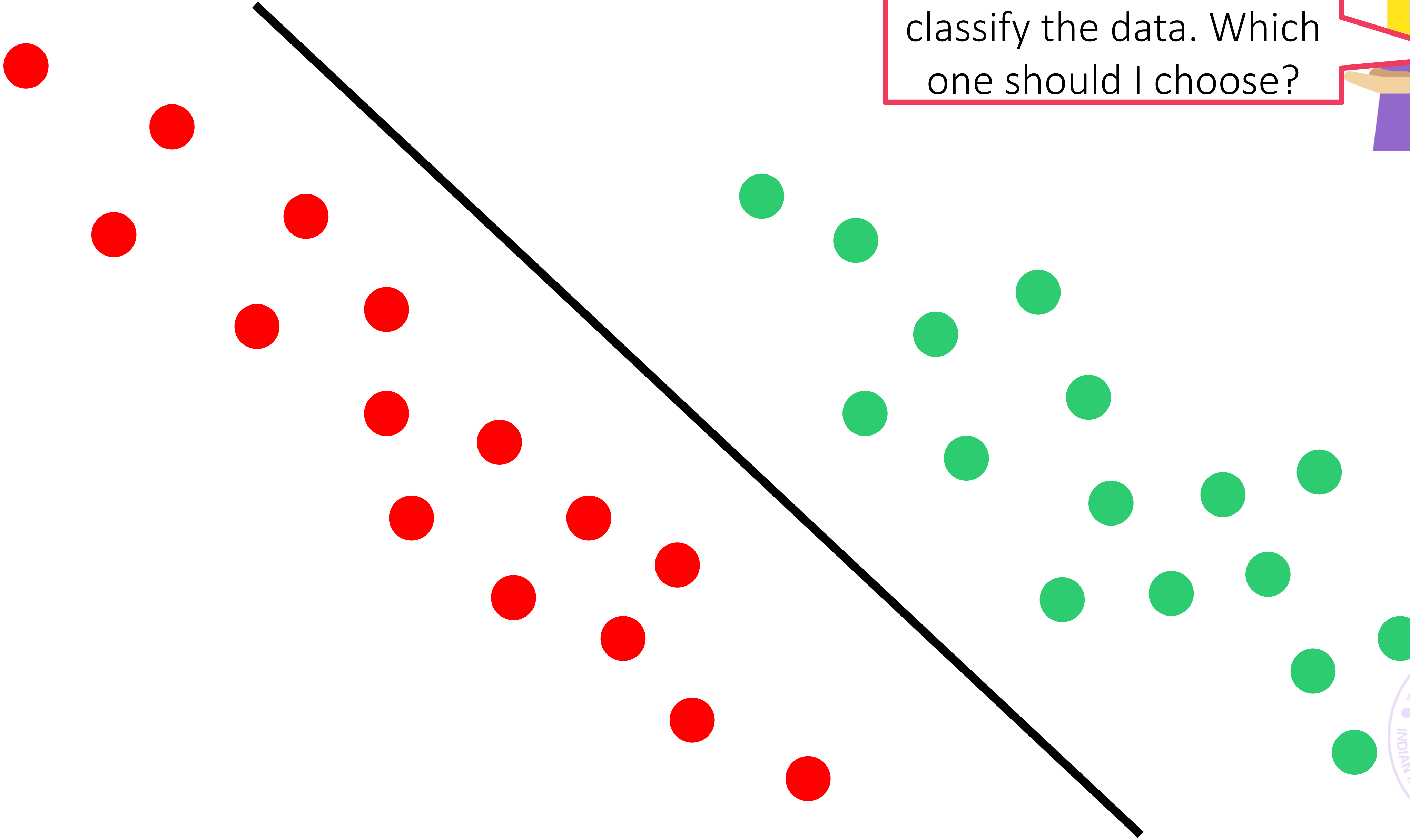
The “best” Linear Classifier



The “best” Linear Classifier

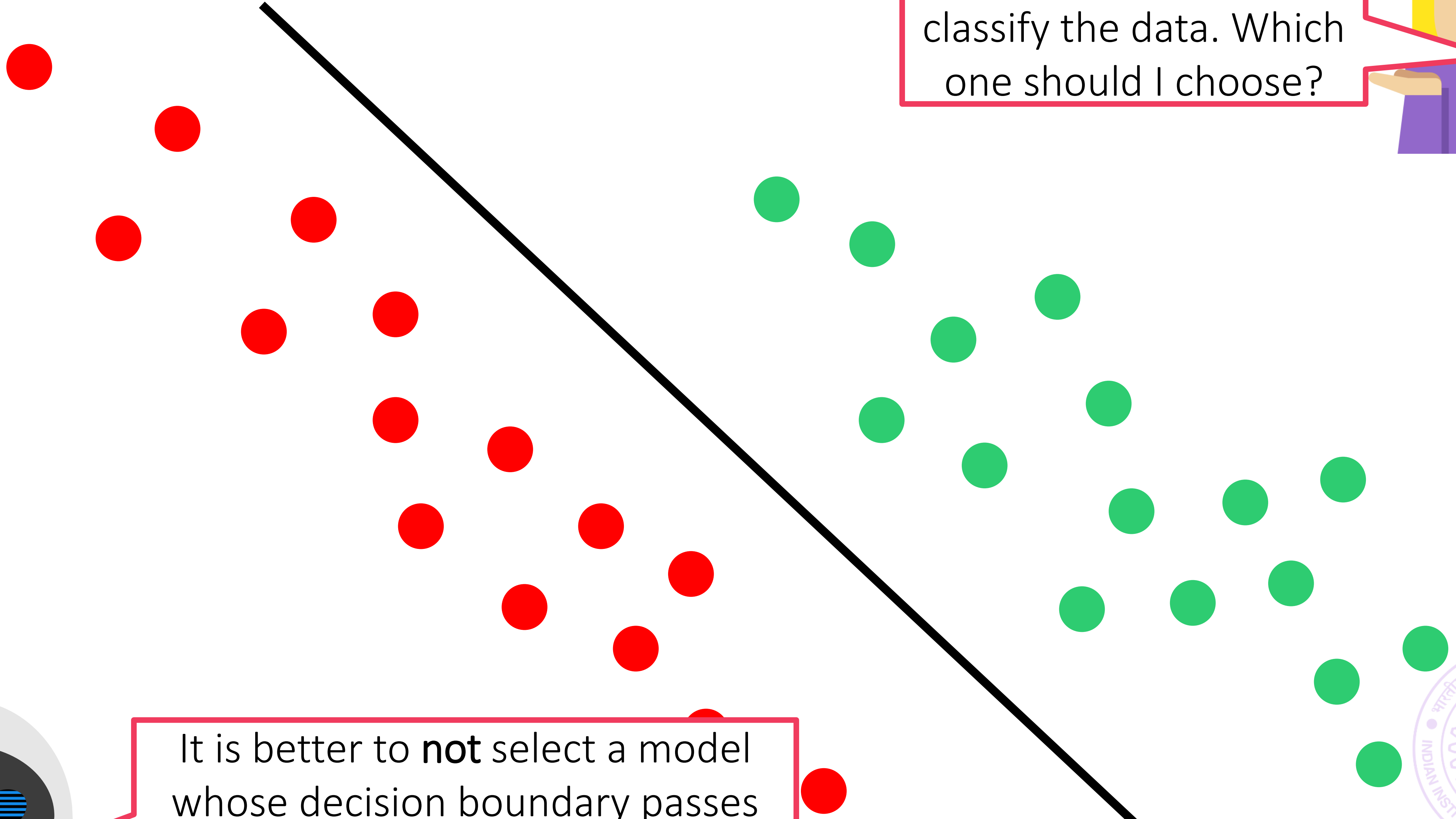
It seems infinitely many classifiers perfectly classify the data. Which one should I choose?





The “best” Linear Classifier

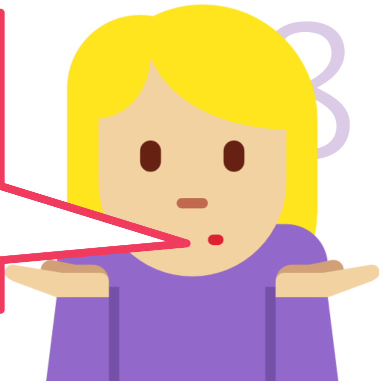
It seems infinitely many classifiers perfectly classify the data. Which one should I choose?

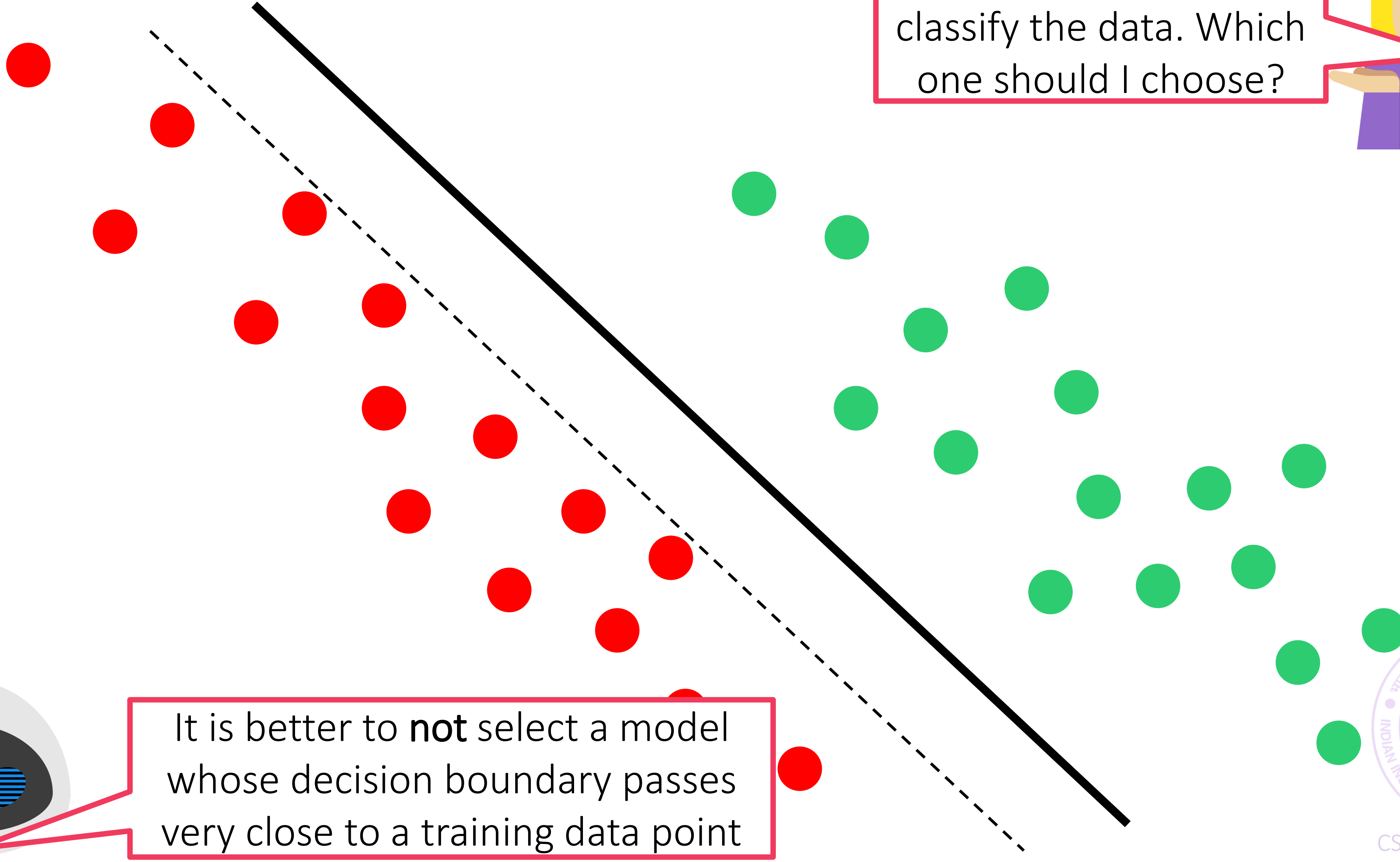


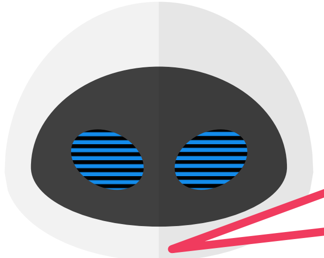
It is better to **not** select a model whose decision boundary passes very close to a training data point

The “best” Linear Classifier

It seems infinitely many classifiers perfectly classify the data. Which one should I choose?



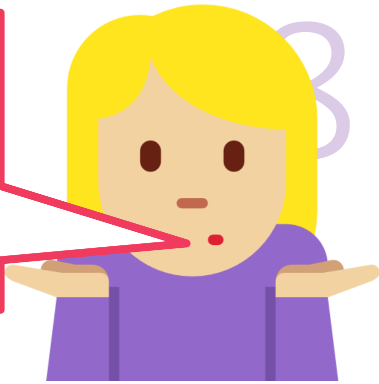


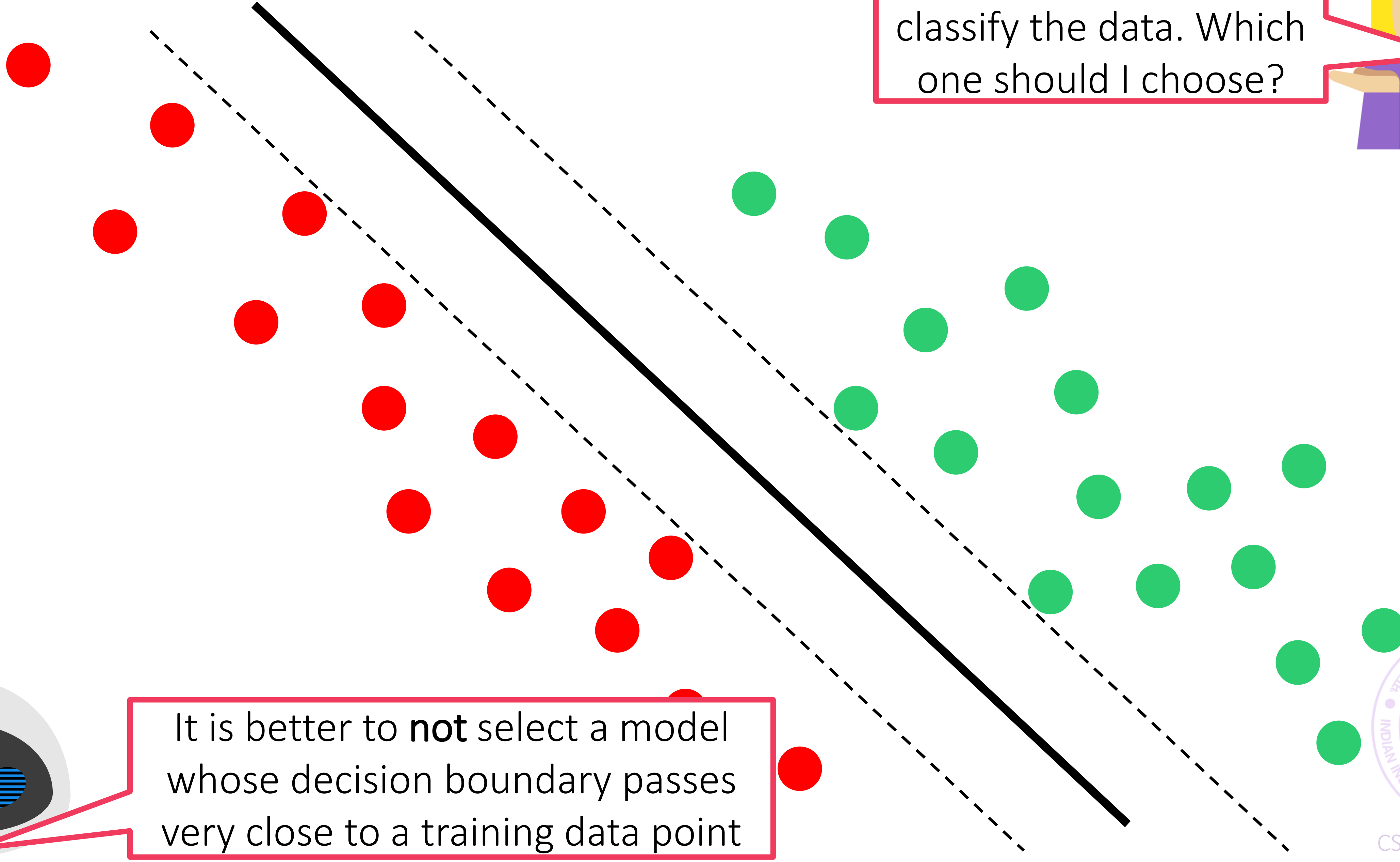


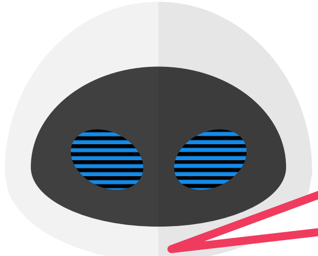
It is better to **not** select a model whose decision boundary passes very close to a training data point

The “best” Linear Classifier

It seems infinitely many classifiers perfectly classify the data. Which one should I choose?



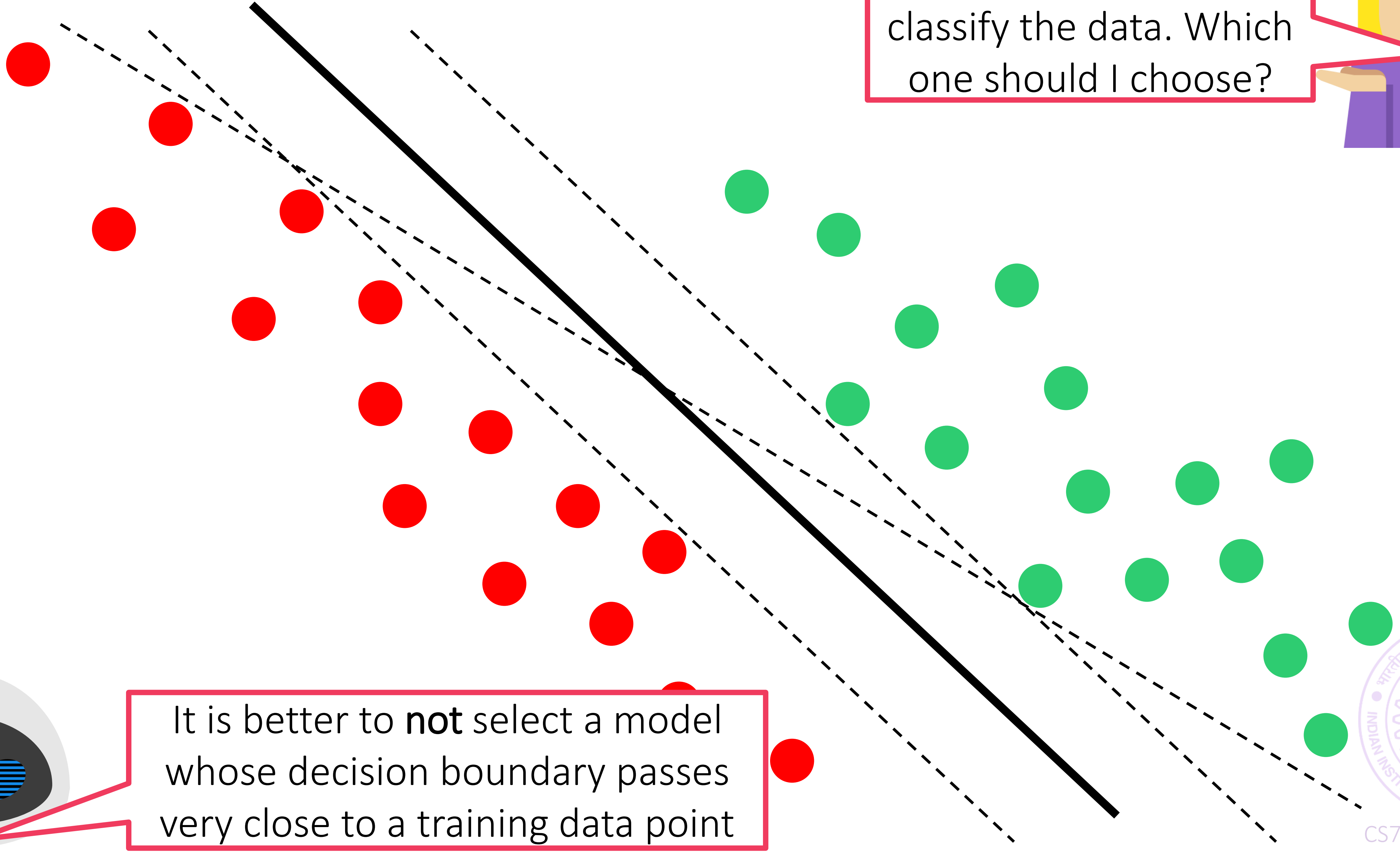




It is better to **not** select a model whose decision boundary passes very close to a training data point

The “best” Linear Classifier

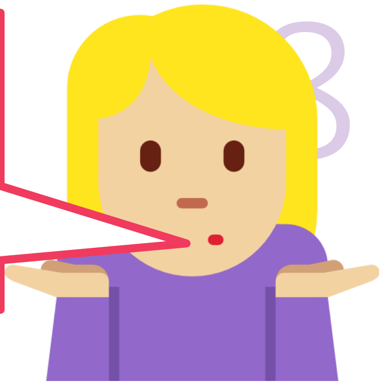
It seems infinitely many classifiers perfectly classify the data. Which one should I choose?

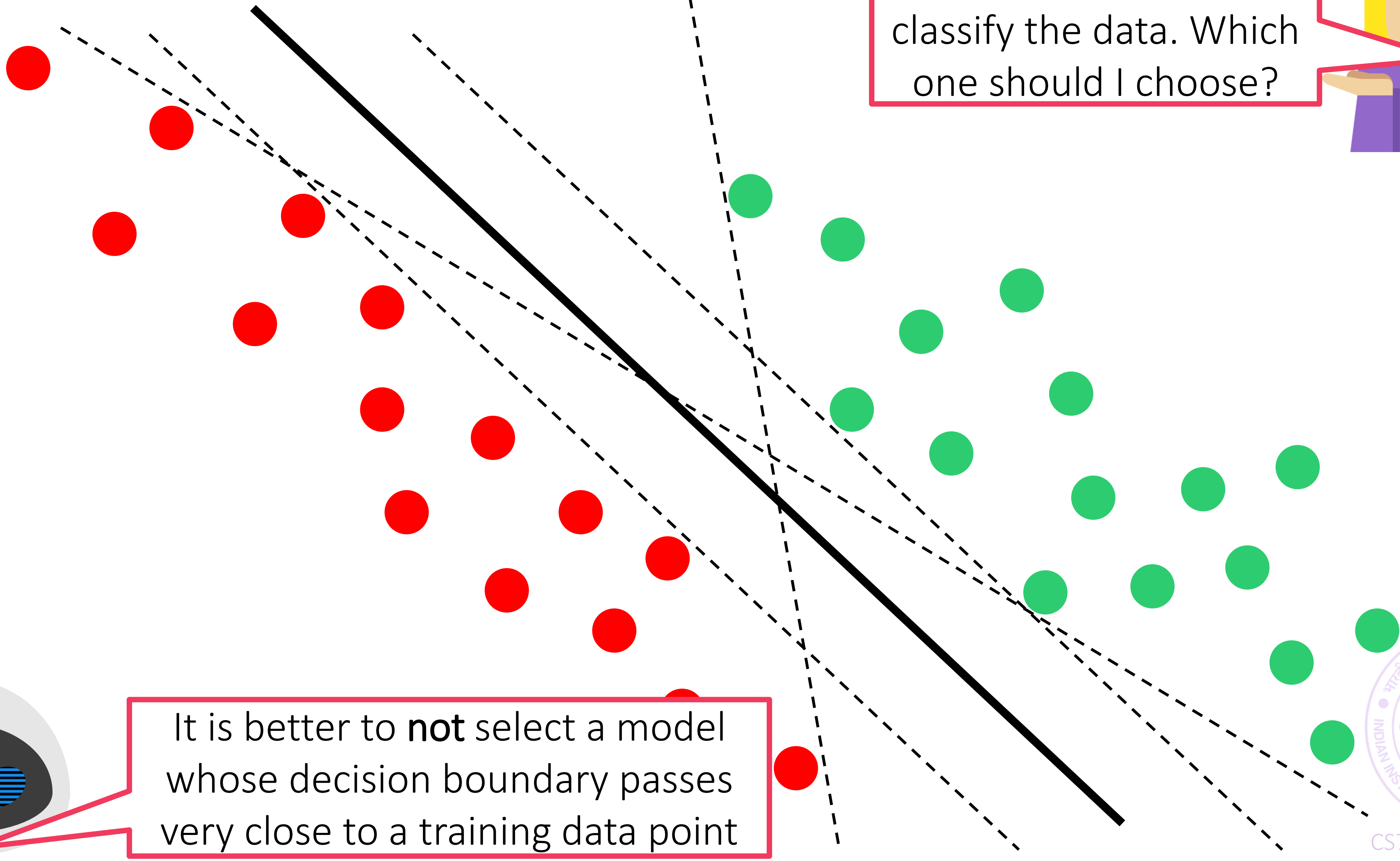


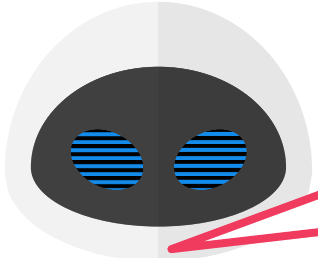
It is better to **not** select a model whose decision boundary passes very close to a training data point

The "best" Linear Classifier

It seems infinitely many classifiers perfectly classify the data. Which one should I choose?

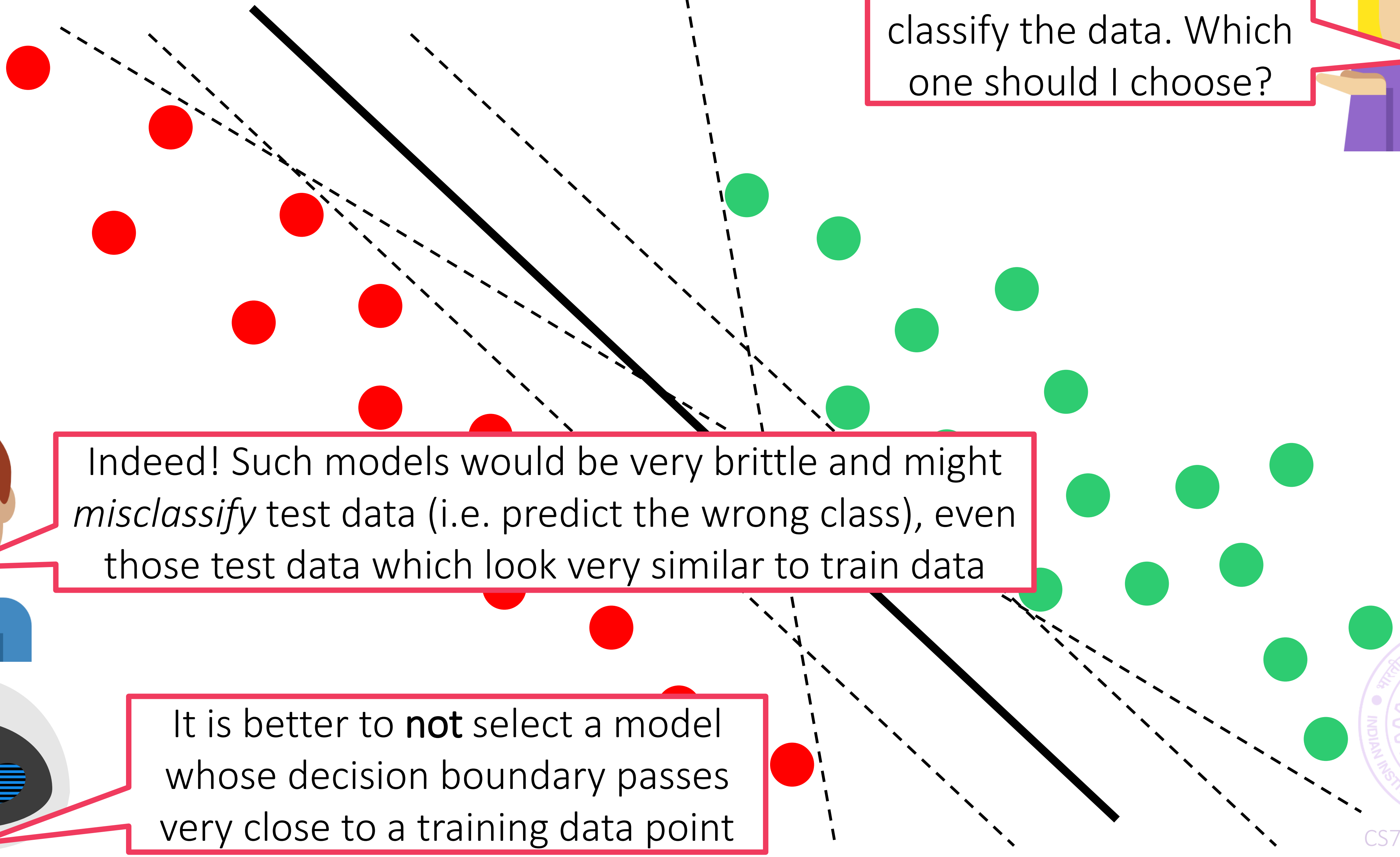


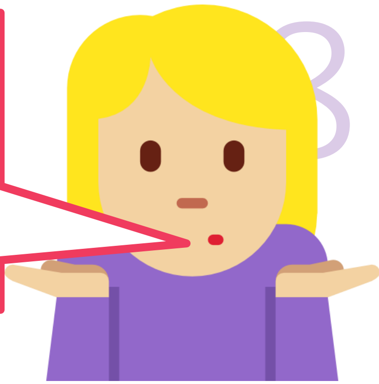




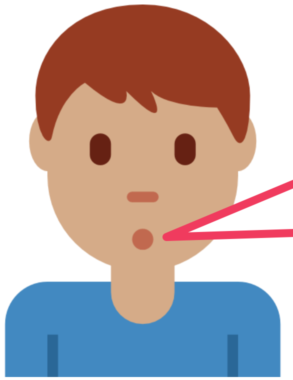
It is better to **not** select a model whose decision boundary passes very close to a training data point

The "best" Linear Classifier

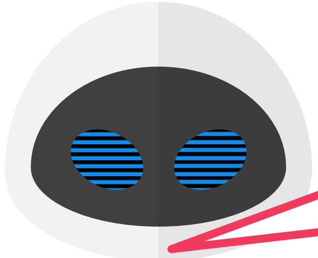




It seems infinitely many classifiers perfectly classify the data. Which one should I choose?



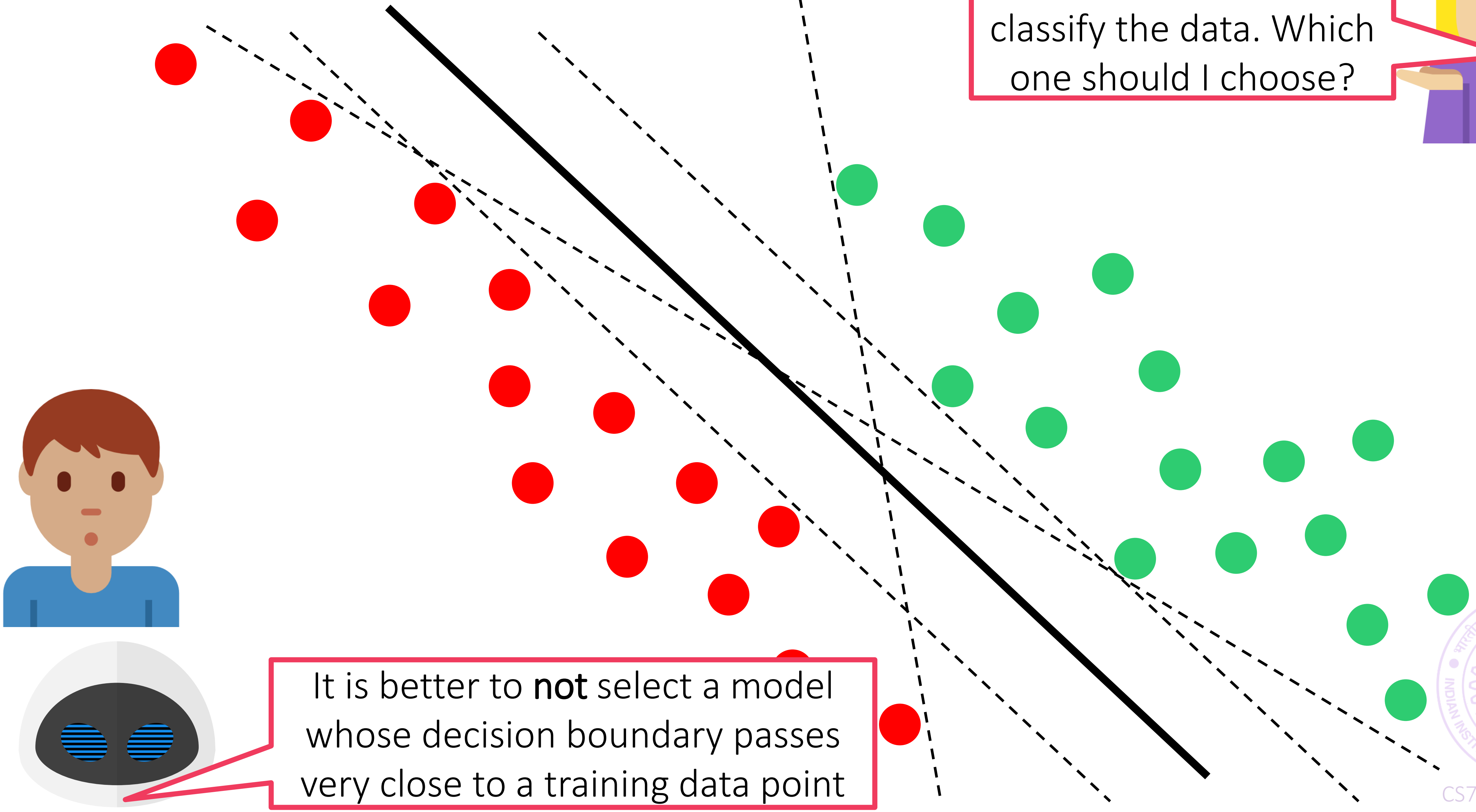
Indeed! Such models would be very brittle and might *misclassify* test data (i.e. predict the wrong class), even those test data which look very similar to train data



It is better to **not** select a model whose decision boundary passes very close to a training data point

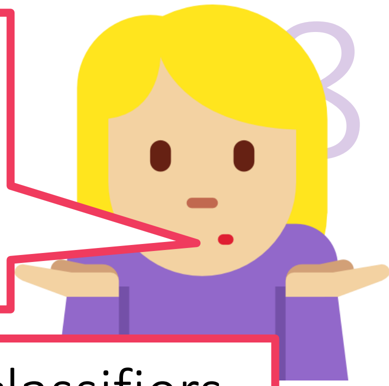
The “best” Linear Classifier

It seems infinitely many classifiers perfectly classify the data. Which one should I choose?

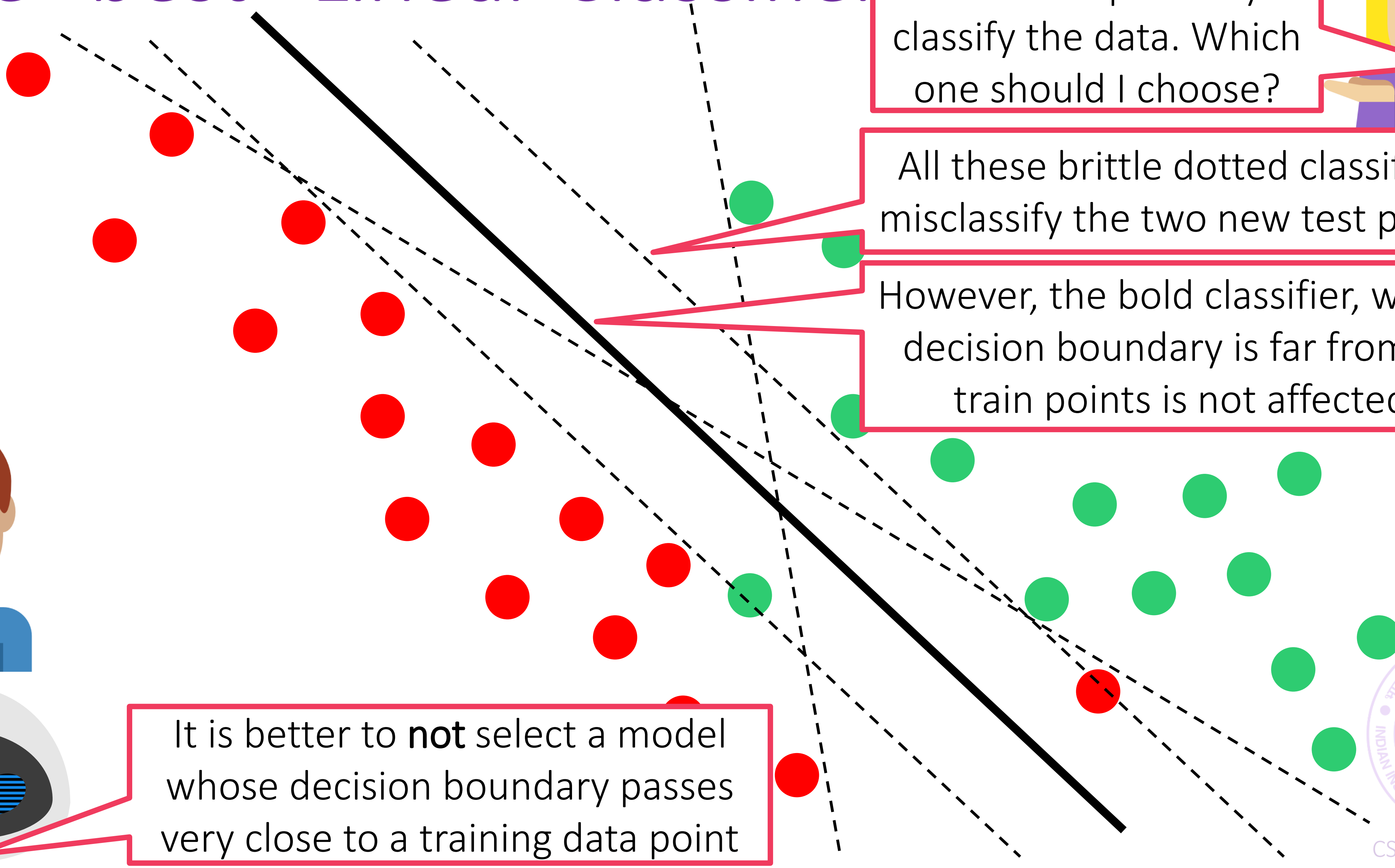


It is better to **not** select a model whose decision boundary passes very close to a training data point

The “best” Linear Classifier

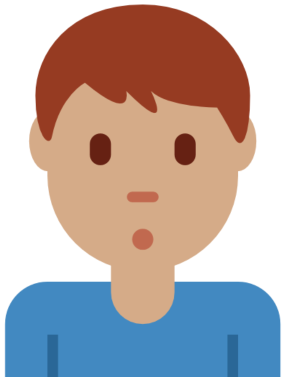


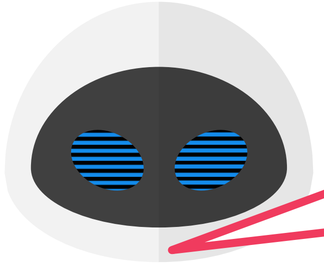
It seems infinitely many classifiers perfectly classify the data. Which one should I choose?



All these brittle dotted classifiers misclassify the two new test points

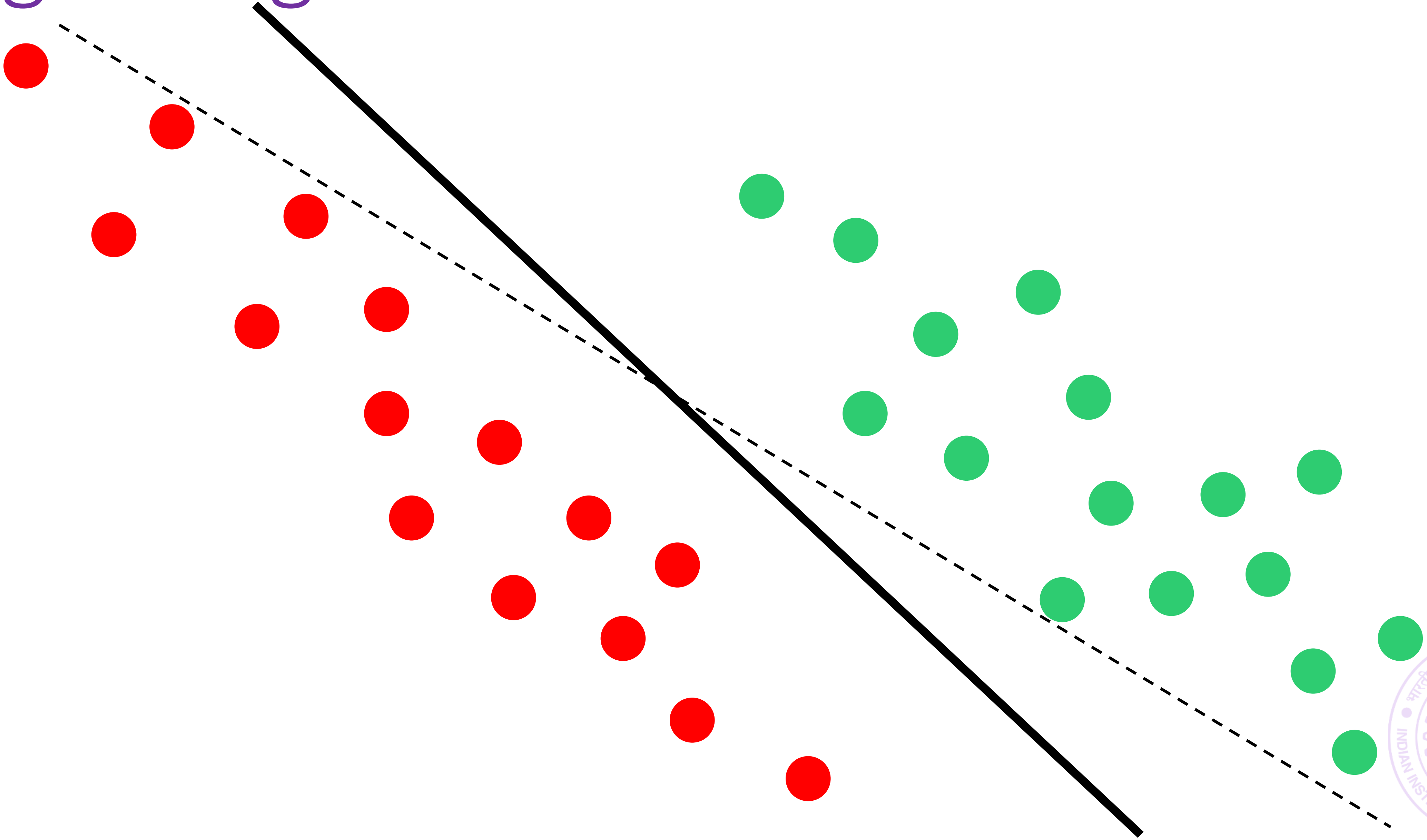
However, the bold classifier, whose decision boundary is far from all train points is not affected



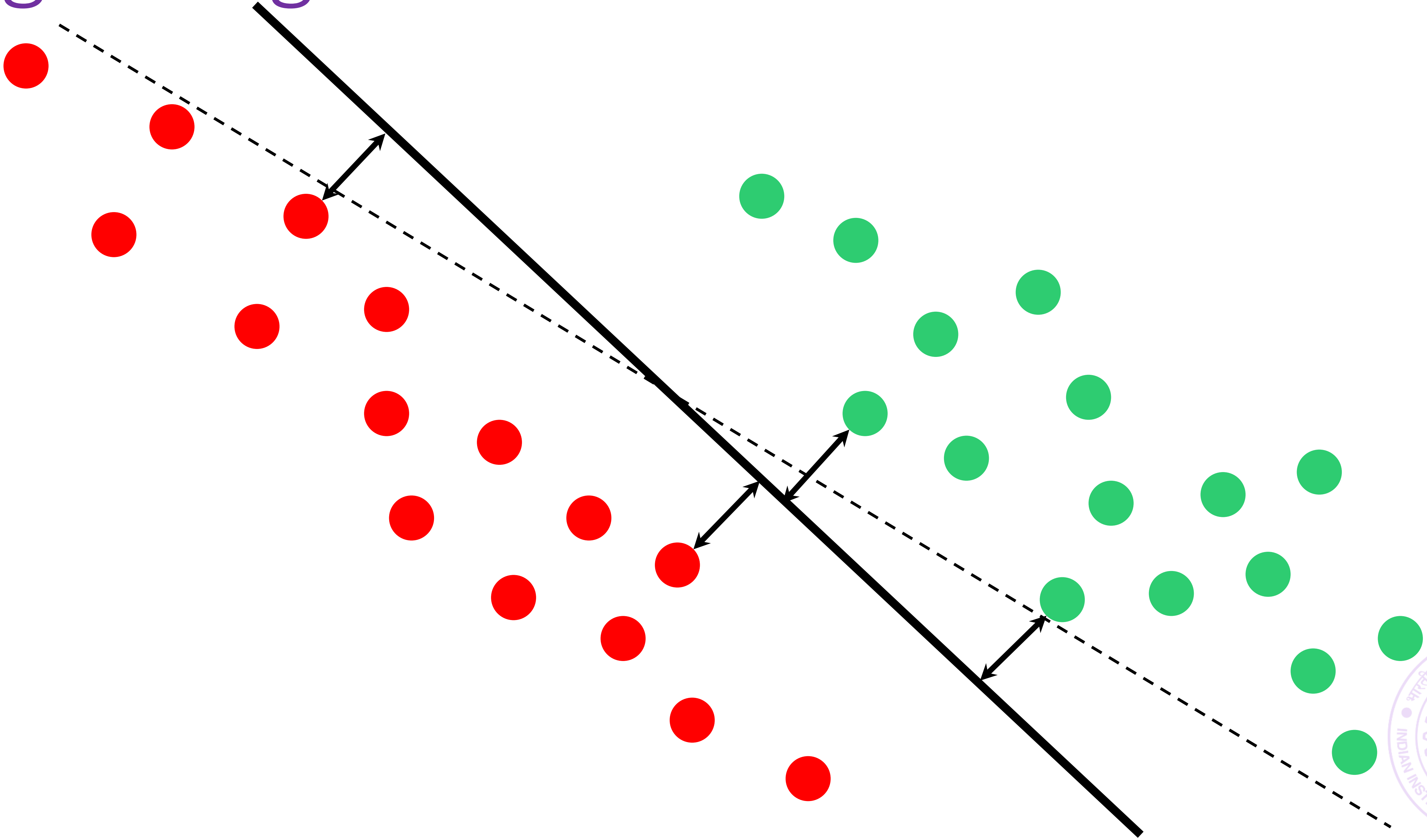


It is better to **not** select a model whose decision boundary passes very close to a training data point

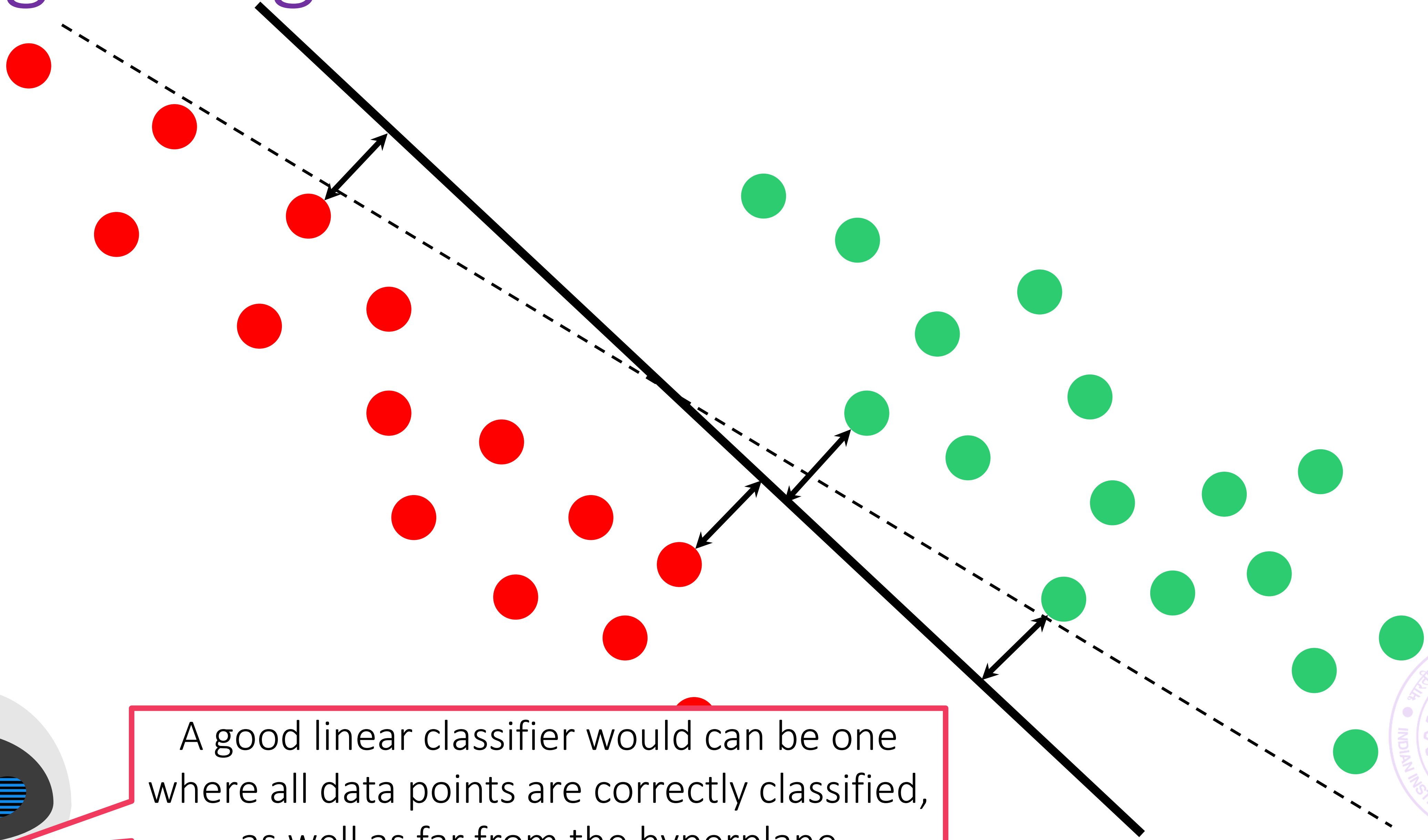
Large Margin Classifiers

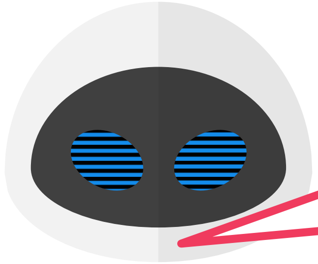


Large Margin Classifiers



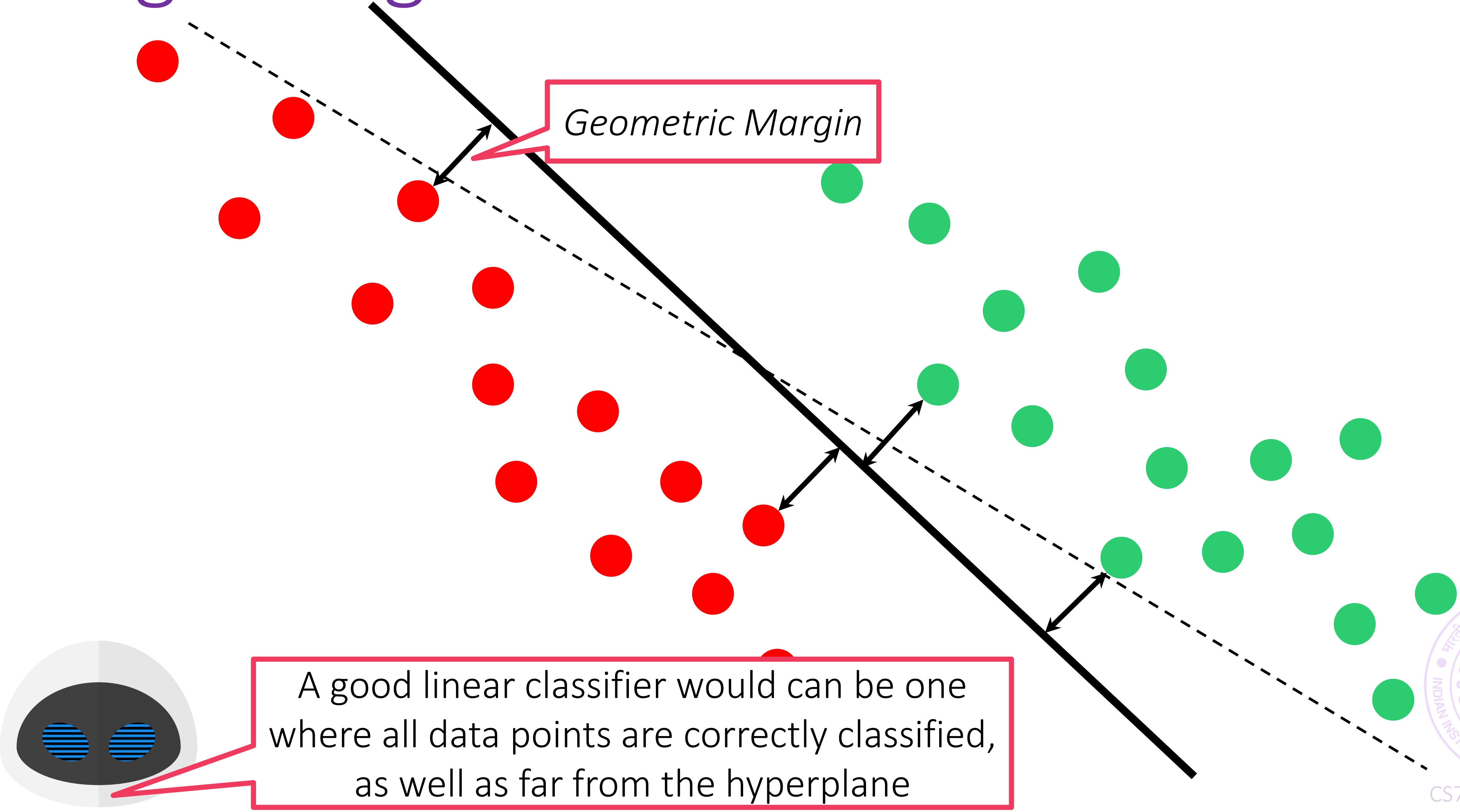
Large Margin Classifiers





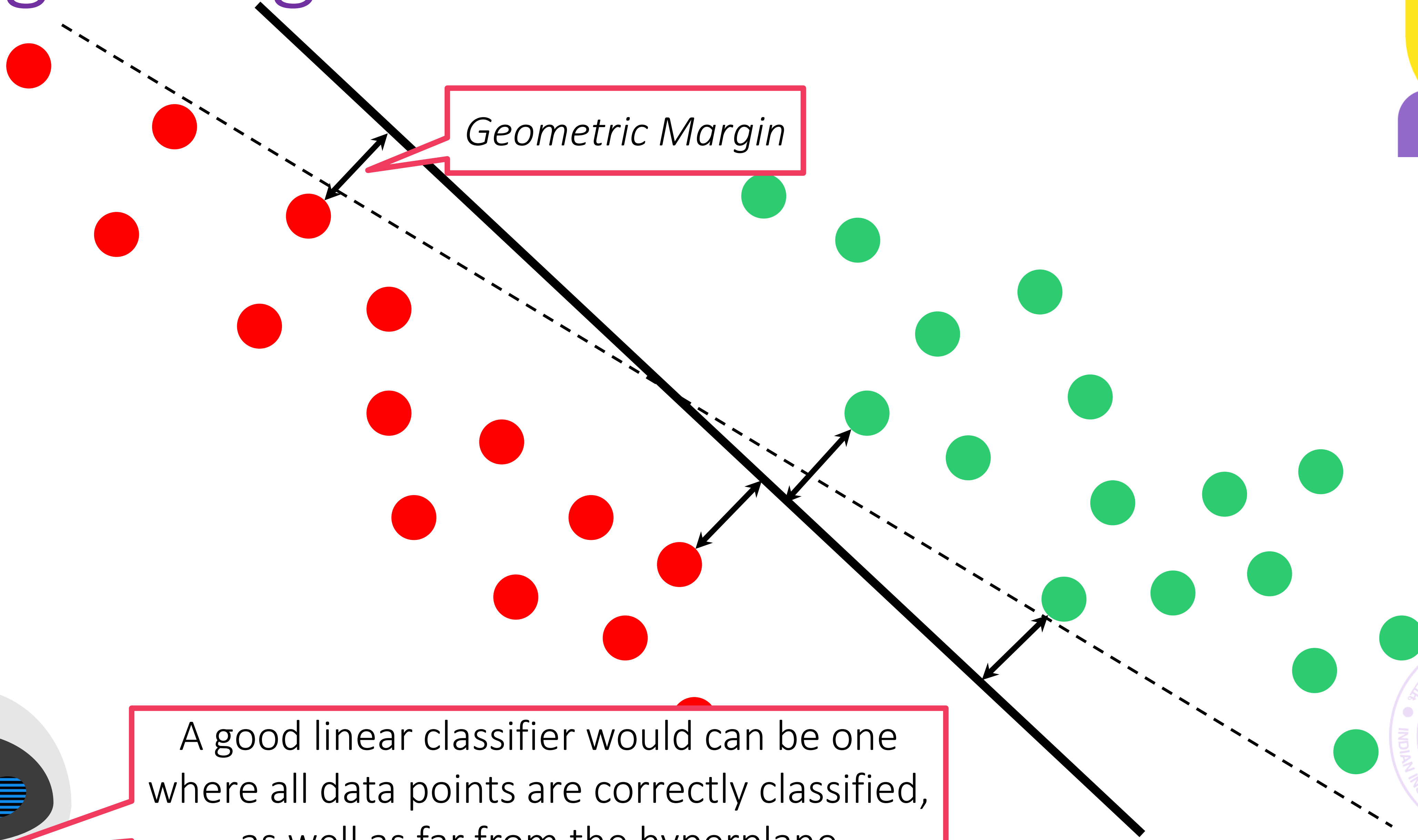
A good linear classifier would can be one where all data points are correctly classified, as well as far from the hyperplane

Large Margin Classifiers



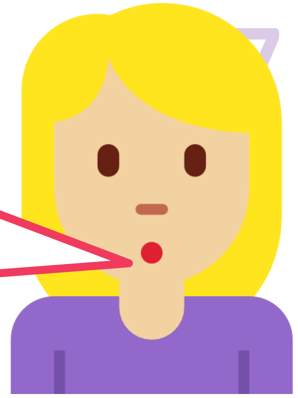
A good linear classifier would can be one where all data points are correctly classified, as well as far from the hyperplane

Large Margin Classifiers

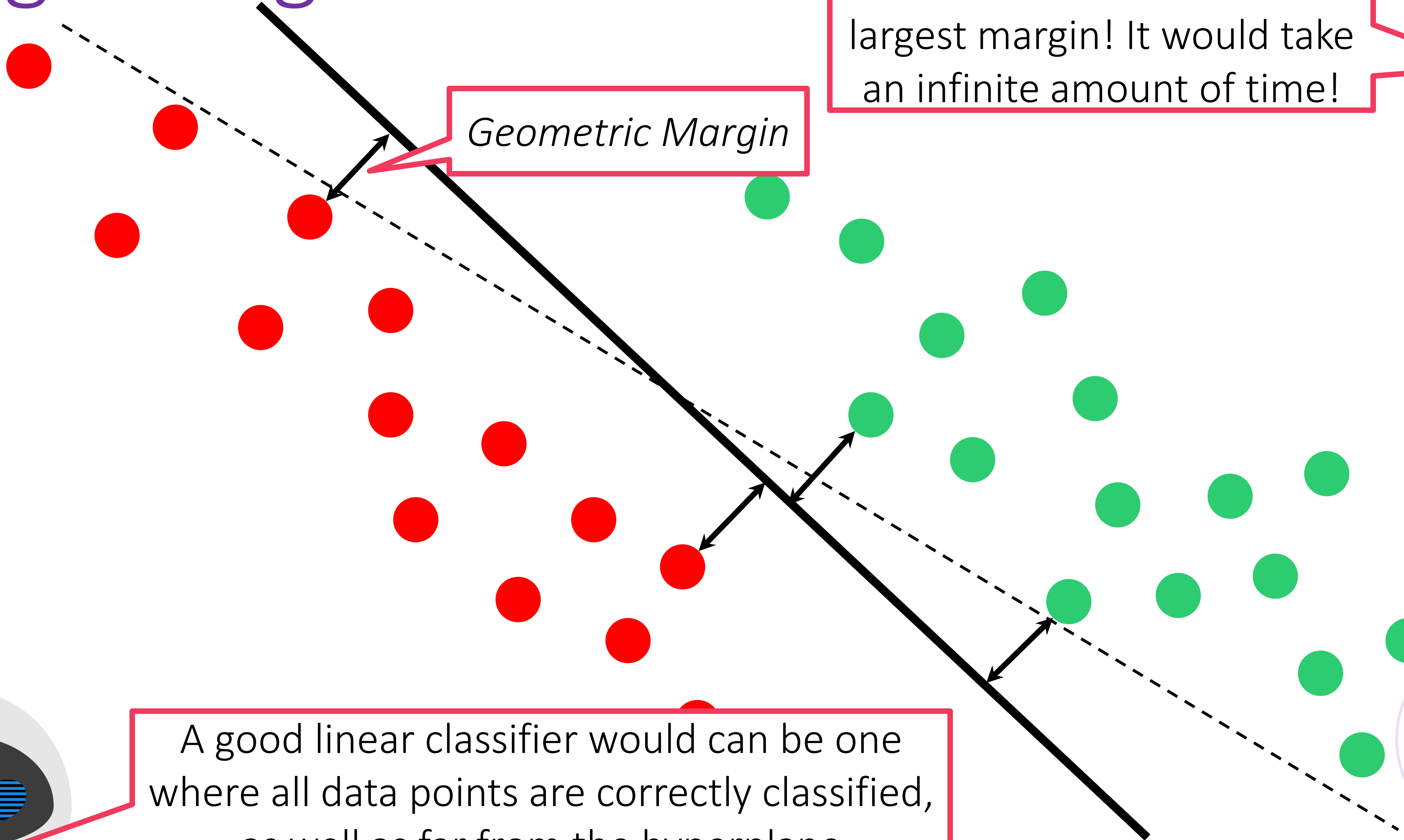


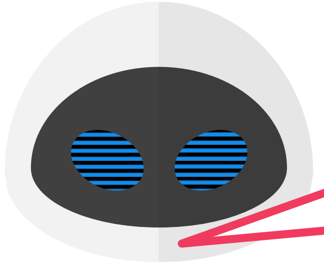
A good linear classifier would can be one where all data points are correctly classified, as well as far from the hyperplane

Large Margin Classifiers



I cannot search all classifiers to find the one with the largest margin! It would take an infinite amount of time!

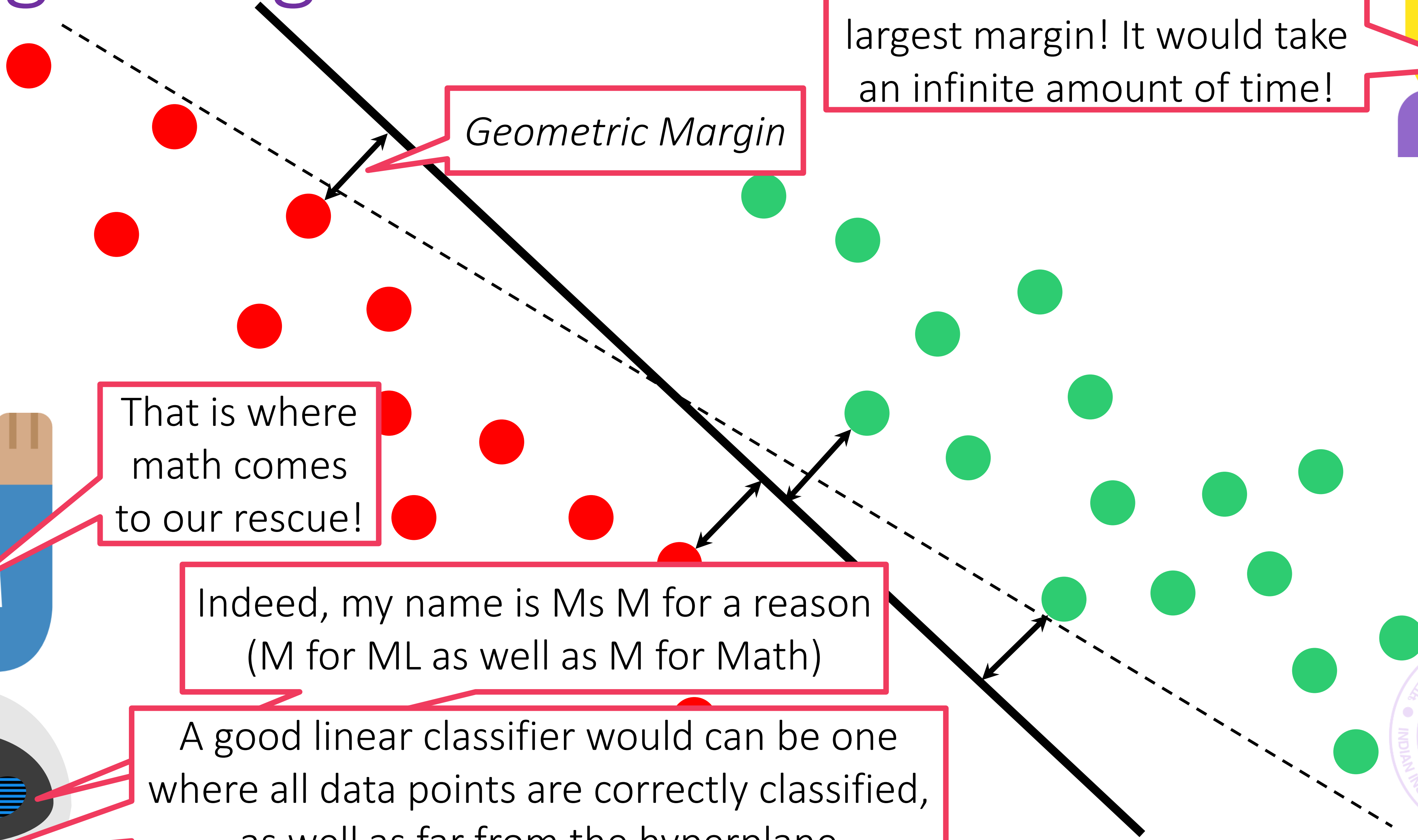




A good linear classifier would can be one where all data points are correctly classified, as well as far from the hyperplane

Large Margin Classifiers

I cannot search all classifiers to find the one with the largest margin! It would take an infinite amount of time!



That is where math comes to our rescue!

Indeed, my name is Ms M for a reason (M for ML as well as M for Math)

A good linear classifier would can be one where all data points are correctly classified, as well as far from the hyperplane

Large Margin Classifiers

The distance of origin from hyperplane $\mathbf{w}^\top\mathbf{x} + b = 0$ is $|b|/\|\mathbf{w}\|_2$

The distance of a point $\mathbf{p}$ from this hyperplane is $|\mathbf{w}^\top\mathbf{p} + b|/\|\mathbf{w}\|_2$

Given train data for a binary classification problem $\left\{\left(\mathbf{x}^i, y^i\right)\right\}_{i=1}^{n}$ where $\mathbf{x}^i \in \mathbb{R}^d$ and $y^i \in \{-1,1\}$, we want two things from a classifier

It should classify every point correctly – how to ask this politely?

One way: demand that for all $i = 1 \ldots n$, $\mathbf{sign}\left(\mathbf{w}^\top\mathbf{x}^i + b\right) = y^i$

Easier way: demand that for all $i = 1 \ldots n$, $y^i \cdot (\mathbf{w}^\top\mathbf{x}^i + \mathrm{b}) \geq 0$

It should not let any data point come close to the boundary

Demand that $\min_{i=1\ldots n} \left|\mathbf{w}^\top\mathbf{x}^i + b\right|/\|\mathbf{w}\|_2$ be as large as possible

Geometric Margin

Support Vector Machines

Just a fancy way of saying

> *Please find me a linear classifier that perfectly classifies the train data while keeping data points as far away from the hyperplane as possible*

The mathematical way of writing this request is the following

Constraints

$$\max_{\mathbf{w},b} \left\{ \min_{i=1\ldots n} \left|\mathbf{w}^\top \mathbf{x}^i + b\right| / \|\mathbf{w}\|_2 \right\}$$

Objective

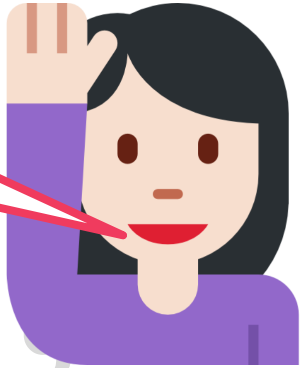
such that $y^i \cdot (\mathbf{w}^\top \mathbf{x}^i + \mathrm{b}) \geq 0$ for all $i = 1 \ldots n$

Support Vector Machines

Just a fancy way of saying

> *Please find me a linear classifier that perfectly classifies the train data while keeping data points as far away from the hyperplane as possible*

Let us simplify this optimization problem



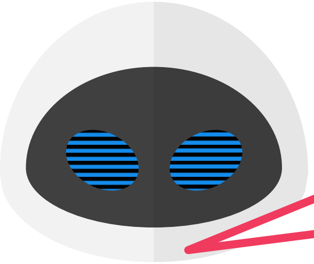
The mathematical way of writing this request is the following

Constraints

$$\max_{\mathbf{w},b}\left\{\min_{i=1\ldots n}\left|\mathbf{w}^\top\mathbf{x}^i+b\right|/\|\mathbf{w}\|_2\right\}$$

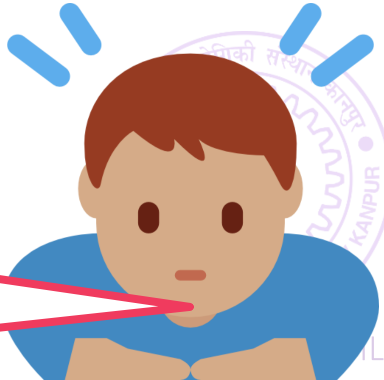
Objective

such that $y^i\cdot(\mathbf{w}^\top\mathbf{x}^i+\mathrm{b})\geq 0$ for all $i=1\ldots n$



This is known as an *optimization problem* with an *objective* and lots of *constraints*

This looks so complicated, how will I ever find a solution to this optimization problem?



Constrained Optimization 101

HOW WE MUST SPEAK TO MS M

Objective

Constraints

$$\min_x f(x)$$

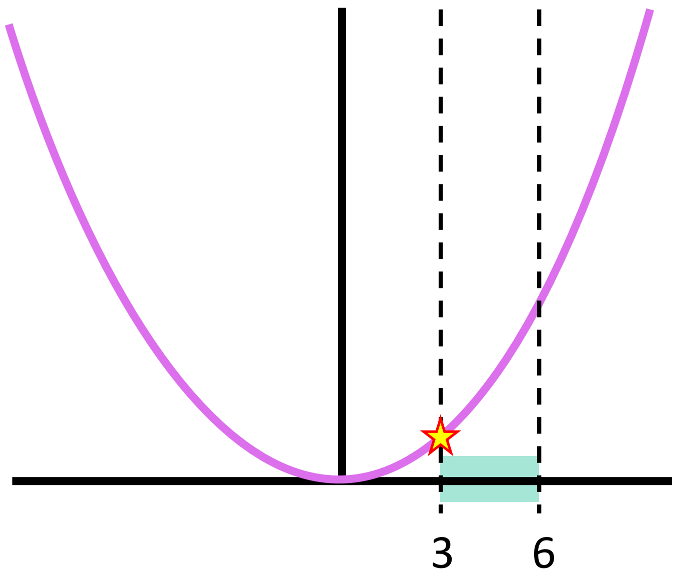
such that $p(x) < 0$

and $q(x) > 0$ etc. etc.

$$\min_x x^2$$

s.t. $x \leq 6$

and $x \geq 3$



HOW WE SPEAK TO A HUMAN

I want to find an unknown x that gives me the *best* value according to this function f

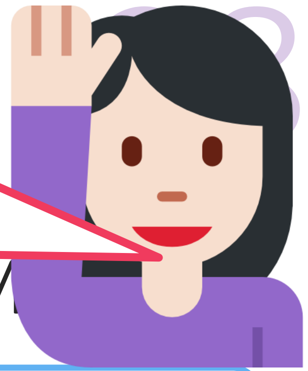
Oh! btw, not any x would do! x must satisfy these conditions

All I am saying is, of the values of x that satisfy my conditions, find me the one that gives the best value according to f

Constrained Optimi

HOW WE MUST SPEAK TO MS M ... HOW WE SPEAK TO A HUM

Constraints are usually specified using math equations. The set of points that satisfy *all* the constraints is called the *feasible set* of the optimization problem



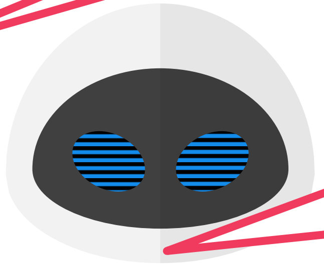
Objective

Constraints

$$\min_x f(x)$$

such that $p(x) < 0$

and $q(x) > 0$ etc. etc.



For your specified constraints, the optimal (least) value of f is 9 and it is achieved at $x = 3$

I want to find an unknown x that gives me the best value ... on f

... would do! x must satisfy these conditions

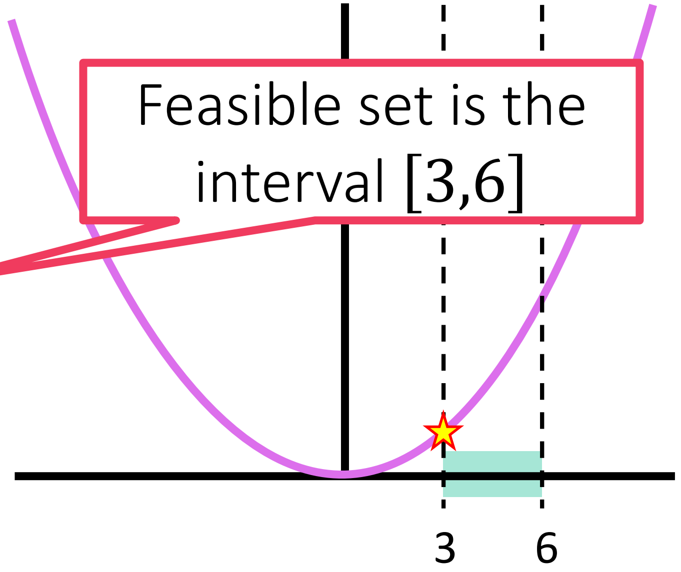
All I am saying is, of the values of x that satisfy my conditions, find me the one that gives the best value according to f

$$\min_x x^2$$

s.t. $x \leq 6$

and $x \geq 3$

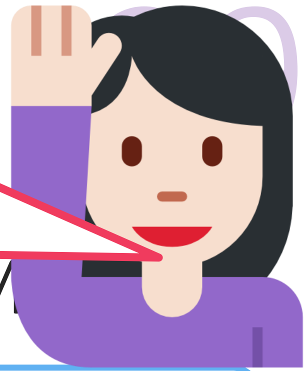
Feasible set is the interval [3,6]



Constrained Optimi

HOW WE MUST SPEAK TO MS M

Constraints are usually specified using math equations. The set of points that satisfy *all* the constraints is called the *feasible set* of the optimization problem



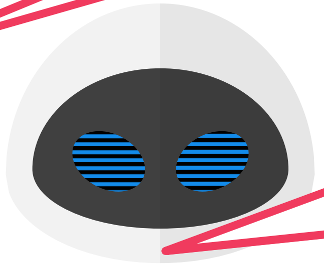
Objective

Constraints

$$\min_x f(x)$$

such that $p(x) < 0$

and $q(x) > 0$ etc. etc.



You optimization problem has no solution since no point satisfies all your constraints ☹

I want to find an unknown x

value

on f

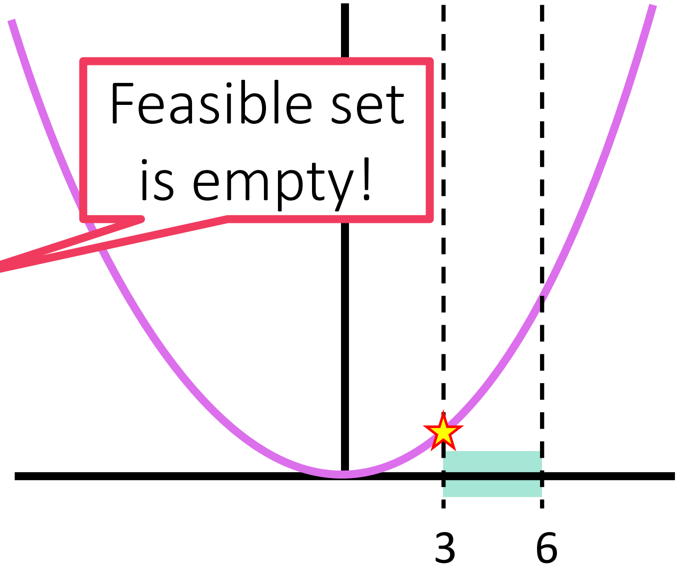
ld do! x must satisfy these conditions

All I am saying is, of the values of x that satisfy my conditions, find me the one that gives the best value according to f

$$\min_x x^2$$

s.t. $x \geq 6$

and $x \leq 3$



Back to SVMs

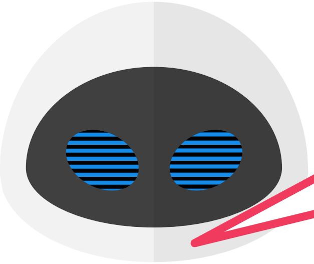
Assume there do exist models that perfectly classify all train data

Consider one such model $(\mathbf{w}, b)$ which classifies train data perfectly

Now, $\left|\mathbf{w}^\top \mathbf{x}^i + b\right| / \|\mathbf{w}\|_2 = \left|y^i \cdot \left(\mathbf{w}^\top \mathbf{x}^i + b\right)\right| / \|\mathbf{w}\|_2$ as $y^i \in \{-1,1\}$

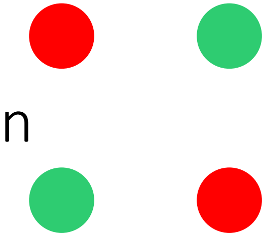
Thus, geometric margin is same as $\min_{i=1\ldots n} \left|y^i \cdot \left(\mathbf{w}^\top \mathbf{x}^i + b\right)\right| / \|\mathbf{w}\|_2 = \min_{i=1\ldots n} y^i \cdot \left(\mathbf{w}^\top \mathbf{x}^i + b\right) / \|\mathbf{w}\|_2$ since model has perfect classification!

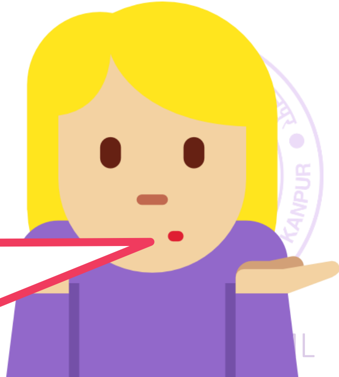
We will use this useful fact to greatly simplify the optimization problem



We will remove this assumption later

What if train data is *non-linearly separable* i.e no linear classifier can perfectly classify it? For example





Support Vector Mach

Called the *functional margin*. Note that geometric margin = functional margin/$\|\mathbf{w}\|_2$

Let i_0 be the data point that comes closest to the hyperplane i.e.

$$\min_{i=1\ldots n} y^i \cdot \left(\mathbf{w}^\top \mathbf{x}^i + b\right) = y^{i_0} \cdot \left(\mathbf{w}^\top \mathbf{x}^{i_0} + b\right)$$

Recall that all this discussion holds only for a perfect classifier $(\mathbf{w}, b)$

Let $\epsilon = y^{i_0} \cdot \left(\mathbf{w}^\top \mathbf{x}^{i_0} + b\right)$ and consider $\tilde{\mathbf{w}} = \mathbf{w}/\epsilon\,, \tilde{b} = b/\epsilon$

Note this gives us $y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right) \geq 1$ for all $i = 1 \ldots n$ as well as $\min_{i=1\ldots n} y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right)/\|\tilde{\mathbf{w}}\|_2 = 1/\|\tilde{\mathbf{w}}\|_2$ (as $y^{i_0} \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^{i_0} + \tilde{b}\right) = 1$)

Thus, instead of searching for $(\mathbf{w}, b)$, easier to search for $\left(\tilde{\mathbf{w}}, \tilde{b}\right)$

$$\max_{\tilde{\mathbf{w}}, \tilde{b}} \left\{1/\|\tilde{\mathbf{w}}\|_2\right\}$$

such that $y^i \cdot (\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}) \geq 1$ for all $i = 1 \ldots n$

Support Vector Mach

Called the *functional margin*. Note that geometric margin = functional margin/$\|\mathbf{w}\|_2$

Let i_0 be the data point that comes closest to the hyperplane i.e.
$\min_{i=1\ldots n} y^i \cdot \left(\mathbf{w}^\top \mathbf{x}^i + b\right) = y^{i_0} \cdot \left(\mathbf{w}^\top \mathbf{x}^{i_0} + b\right)$

Recall that all this discussion holds only for a perfect classifier $(\mathbf{w}, b)$

Let $\epsilon = y^{i_0} \cdot \left(\mathbf{w}^\top \mathbf{x}^{i_0} + b\right)$ and consider $\tilde{\mathbf{w}} = \mathbf{w}/\epsilon, \tilde{b} = b/\epsilon$

Note this gives us $y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right) \geq 1$ for all $i = 1 \ldots n$ as well as
$\min_{i=1\ldots n} y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right)/\|\tilde{\mathbf{w}}\|_2 = 1/\|\tilde{\mathbf{w}}\|_2$ (as $y^{i_0} \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^{i_0} + \tilde{b}\right) = 1$)

Thus, instead of searching for $(\mathbf{w}, b)$, easier to search for $\left(\tilde{\mathbf{w}}, \tilde{b}\right)$

$$\min_{\tilde{\mathbf{w}}, \tilde{b}} \{\|\tilde{\mathbf{w}}\|_2\}$$

such that $y^i \cdot (\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}) \geq 1$ for all $i = 1 \ldots n$

Support Vector Mach

Called the *functional margin*. Note that geometric margin = functional margin/$\|\mathbf{w}\|_2$

Let i_0 be the data point that comes closest to the hyperplane i.e.

$$\min_{i=1\ldots n} y^i \cdot \left(\mathbf{w}^\top \mathbf{x}^i + b\right) = y^{i_0} \cdot \left(\mathbf{w}^\top \mathbf{x}^{i_0} + b\right)$$

Recall that all this discussion holds only for a perfect classifier $(\mathbf{w}, b)$

Let $\epsilon = y^{i_0} \cdot \left(\mathbf{w}^\top \mathbf{x}^{i_0} + b\right)$ and consider $\tilde{\mathbf{w}} = \mathbf{w}/\epsilon, \tilde{b} = b/\epsilon$

Note this gives us $y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right) \geq 1$ for all $i = 1 \ldots n$ as well as $\min_{i=1\ldots n} y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right)/\|\tilde{\mathbf{w}}\|_2 = 1/\|\tilde{\mathbf{w}}\|_2$ (as $y^{i_0} \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^{i_0} + \tilde{b}\right) = 1$)

Thus, instead of searching for $(\mathbf{w}, b)$, easier to search for $\left(\tilde{\mathbf{w}}, \tilde{b}\right)$

$$\min_{\tilde{\mathbf{w}}, \tilde{b}} \left\{\|\tilde{\mathbf{w}}\|_2^2\right\}$$

such that $y^i \cdot (\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}) \geq 1$ for all $i = 1 \ldots n$

The C-SVM Technique

For *linearly separable* cases where we suspect a perfect classifier exists

$$\min_{\tilde{\mathbf{w}},\tilde{b}} \frac{1}{2}\|\tilde{\mathbf{w}}\|_2^2$$

$$\text{s.t. } y^i \cdot (\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}) \geq 1 \text{ for all } i \in [n]$$

If a linear classifier cannot perfectly classify data, then find model using

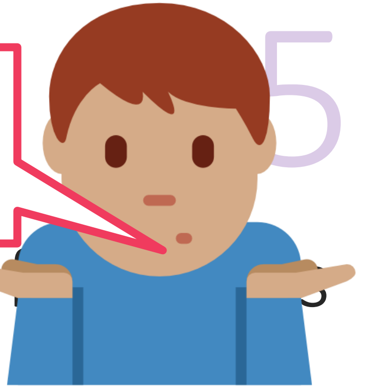
$$\min_{\tilde{\mathbf{w}},\tilde{b},\{\xi_i\}} \frac{1}{2}\|\tilde{\mathbf{w}}\|_2^2 + C\sum_{i=1}^n \xi_i$$

$$\text{s.t. } y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right) \geq 1 - \xi_i \text{ for all } i \in [n]$$

$$\text{as well as } \xi_i \geq 0 \text{ for all } i \in [n]$$

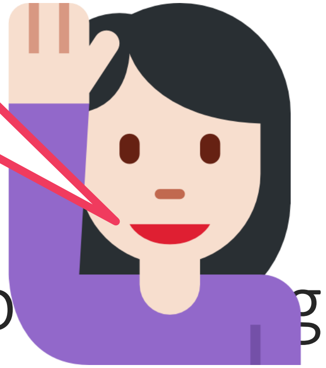
The C-SVM Te

What prevents me from misusing the slack variables to learn a model that misclassifies every data point?



For *linearly separable* cases w

The C term prevents you from doing so. If we set C to be a large value (it is a hyper-parameter), then it will penalize solutions that misuse slack too much



s.t. $y^i \cdot (\tilde{\mathbf{w}}^\top \mathbf{x}$ ... ≥ 1 for all $i \in [n]$

If a linear classifier cannot perfectly classify data, then find mo

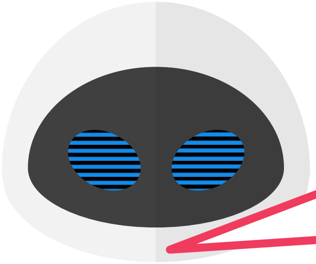
$$\min_{\tilde{\mathbf{w}}, \tilde{b}, \{\xi_i\}} \frac{1}{2}\|\tilde{\mathbf{w}}\|_2^2 + C \sum_{i=1}^{n}$$

Having the constraint $\xi_i \geq 0$ prevents us from misusing slack to artificially inflate the margin

$$\text{s.t. } y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right) \geq 1 - \xi_i \text{ for all } i \in [n]$$

as well as $\xi_i \geq 0$ for all $i \in [n]$

Recall English phrase *"cut me some slack"*



The ξ_i terms are called *slack variables*. They allow some data points to come close to the hyperplane or be misclassified altogether

From C-SVM to Loss Functions

We can further simplify the previous optimization problem

Note ξ_i basically allows us to have $y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right) < 1$ (even < 0)

Thus, the amount of slack we want is just $\xi_i = 1 - y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right)$

However, recall that we must also satisfy $\xi_i \geq 0$

Another way of saying that if you already have $y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right) \geq 1$, then you don't need any slack i.e. you should have $\xi_i = 0$ in this case

Thus, we need only set $\xi_i = \left[1 - y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right)\right]_+$

The above is nothing but the popular hinge loss function!

From C-SVM to Loss Functions

We can further simplify the previous optimization problem

Note ξ_i basically allows us to have $y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right) < 1$ (even < 0)

Thus, the amount of slack we want is just $\xi_i = 1 - y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right)$

However, recall that we must also satisfy $\xi_i \geq 0$

Another way of saying that if you already have $y^i \cdot \left(\tilde{\mathbf{w}}\right.$

$[x]_+ = \max\{x, 0\}$

then you don't need any slack i.e. you should have $\xi_i = 0$ in this case

Thus, we need only set $\xi_i = \left[1 - y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right)\right]_+$

The above is nothing but the popular hinge loss function!

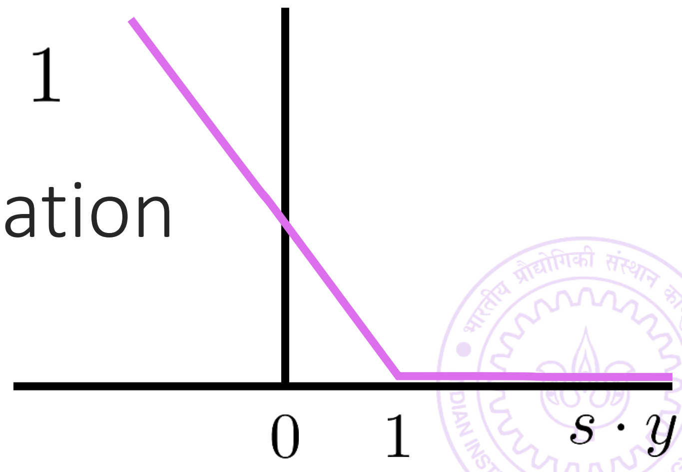
Hinge Loss

Captures how well as a classifier classified a data point

Suppose on a data point $(\mathbf{x}, y), y \in \{-1,1\}$, a model gives prediction score of s (for a linear model $(\mathbf{w}, b)$, we have $s = \mathbf{w}^\top \mathbf{x} + b$)

We obviously want $s \cdot y \geq 0$ for correct classification but we also want $s \cdot y \gg 0$ for large margin – hinge loss function captures both

$$\ell_{\text{hinge}}(s, y) = [1 - s \cdot y]_+ = \begin{cases} 0 & \text{if } s \cdot y \geq 1 \\ 1 - s \cdot y & \text{if } s \cdot y < 1 \end{cases}$$



Note that hinge loss not only penalizes misclassification but also correct classification if the data point gets too close to the hyperplane!

Final Form of C-SVM

Recall that the C-SVM optimization finds a model by solving

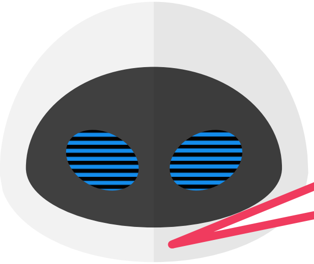
$$\min_{\tilde{\mathbf{w}},\tilde{b},\{\xi_i\}} \frac{1}{2}\|\tilde{\mathbf{w}}\|_2^2 + C\sum_{i=1}^{n}\xi_i$$

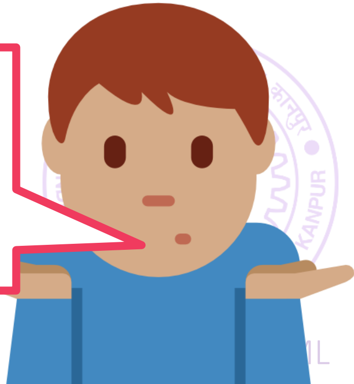
$$\text{s.t. } y^i \cdot \left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}\right) \geq 1 - \xi_i \text{ for all } i \in [n]$$

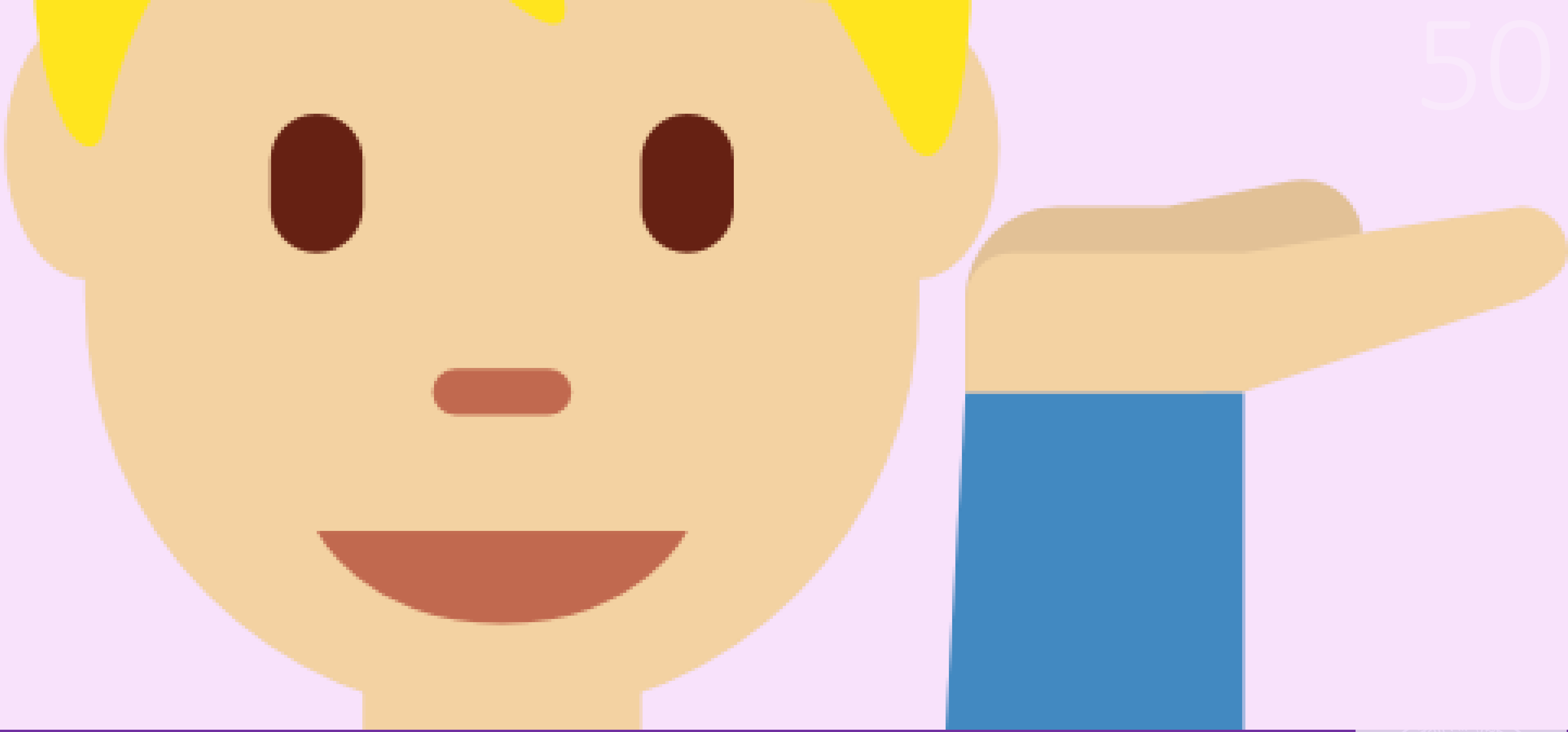
$$\text{as well as } \xi_i \geq 0 \text{ for all } i \in [n]$$

Using the previous discussion, we can rewrite the above very simply

$$\min_{\tilde{\mathbf{w}},\tilde{b}} \frac{1}{2}\|\tilde{\mathbf{w}}\|_2^2 + C\sum_{i=1}^{n}\ell_{\text{hinge}}\left(\tilde{\mathbf{w}}^\top \mathbf{x}^i + \tilde{b}, y^i\right)$$







Emoticons from Flaticon, designed by Twitter
https://www.flaticon.com/packs/smileys-and-people-9