AdaBoost: Weak Learners

- AdaBoost "sits on" a weak learner: you can plug in any weak learner you like.

- Usually this is a Decision Stump, but it doesn't have to be. It just has to be "good enough" (i.e., $p(correct) > 0.5$)).

- In the project, you use the standard AdaBoost algorithm but replace the Decision Stumps with "Weighted Weak Linear" classifiers.

In a decision stump we look for the best weighted classification performance *along the axes of the co-ordinate system*. So, for a system in 2-d, we search along the x-axis, then the y-axis, and return the stump that performs best (according to the weights).

A Weighted Weak Linear classifier removes the restriction of only looking along the co-ordinate axes: instead, it picks ***the best overall direction*** and searches for the best performance along that.

AdaBoost: Weak Learners

Here we present an algorithm for the Weighted Weak Linear classifier.

The presentation is very similar to the original presentation for the decision stump; first we present an unweighted version, to show the idea of picking the best overall direction (producing an unweighted weak linear classifier), then add weights and show how they modify the system to generate a Weighted Weak Linear classifier.

The final classifier is quite similar to the decision stump; in some ways it is simpler (only a single direction to search along), in others more complex (need to choose the direction along which to do the search).

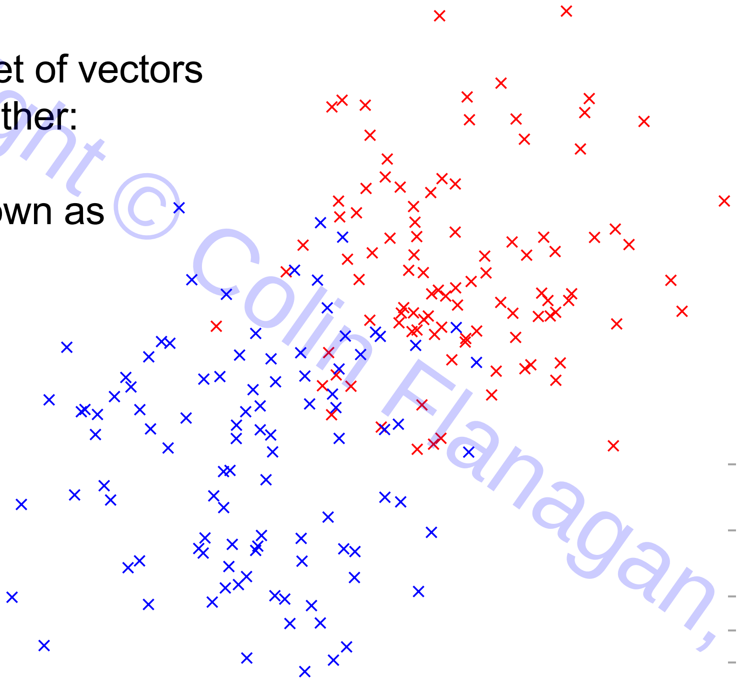
Remember that it is the Weighted Weak Linear classifier you need for the project, the initial presentation of the (unweighted) weak linear classifier is just to "set the scene".

Weak Linear Classifier

Dataset: a set of vectors $\vec{x}_i$ labelled either:

+1 (here shown as red crosses)

or −1 (blue).



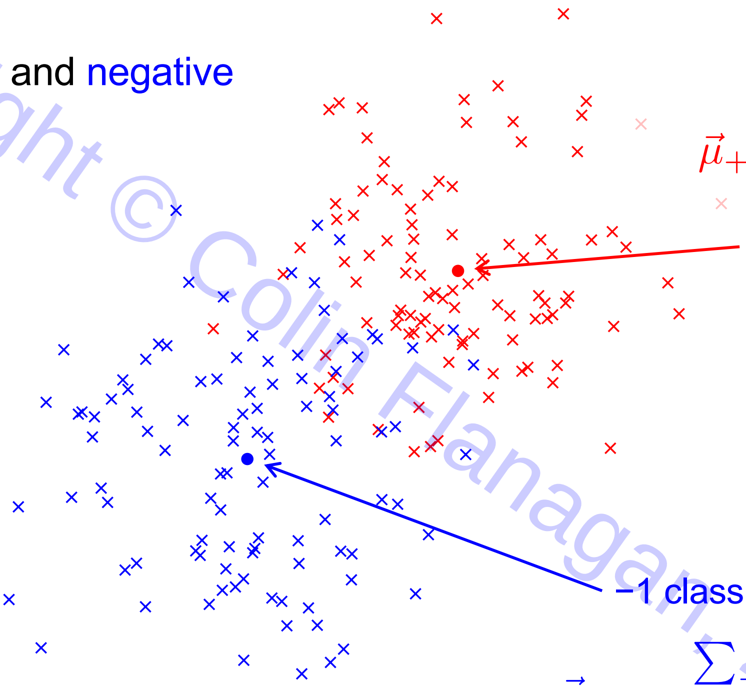
```
-0.834  0.391 -1
 1.663  0.488 +1
-0.287  0.283 -1
 1.195  0.038 +1
-0.159  0.566 -1
-1.335 -0.618 +1
-0.548  0.145 -1
 1.356  0.366 +1
 0.295  0.685 -1
-1.200  0.941 +1
          ⋮
```

Brightspace site: Content / Project Datasets $\left\{ \begin{array}{l} \text{adaboost-train-23.txt} \\ \text{adaboost-test-23.txt} \end{array} \right.$

```
train = np.loadtxt('adaboost-train-23.txt')
```

Weak Linear Classifier: Find Best Direction

Find positive and negative *class means*



$$\vec{\mu}_{+1} = \frac{\sum_{+ve} \vec{x}_{+ve}}{m_{+ve}}$$

+1 class mean

−1 class mean

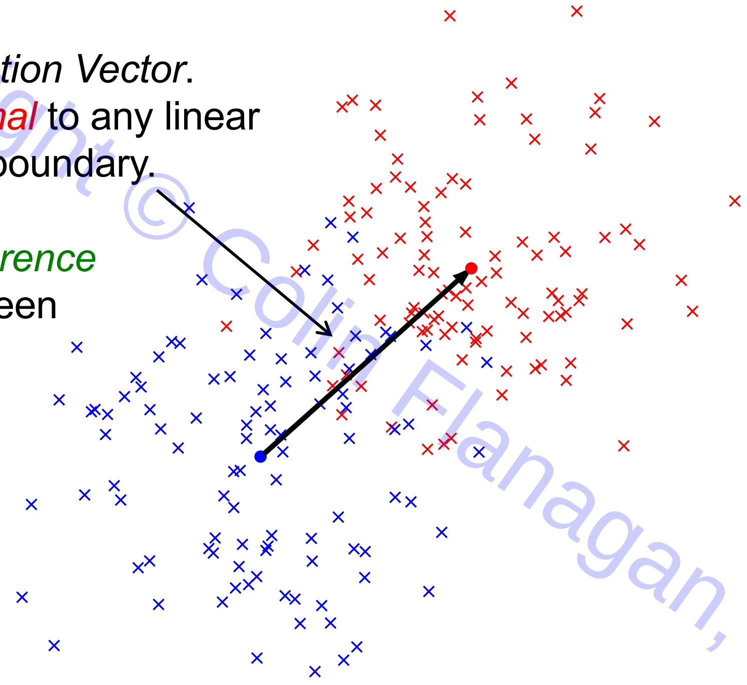
$$\vec{\mu}_{-1} = \frac{\sum_{-ve} \vec{x}_{-ve}}{m_{-ve}}$$

Total number of –ve examples.

Weak Linear Classifier

The *Orientation Vector*.
This is *normal* to any linear separating boundary.

It's the Difference vector between means.



$$\vec{r} = \vec{\mu}_{+1} - \vec{\mu}_{-1}$$

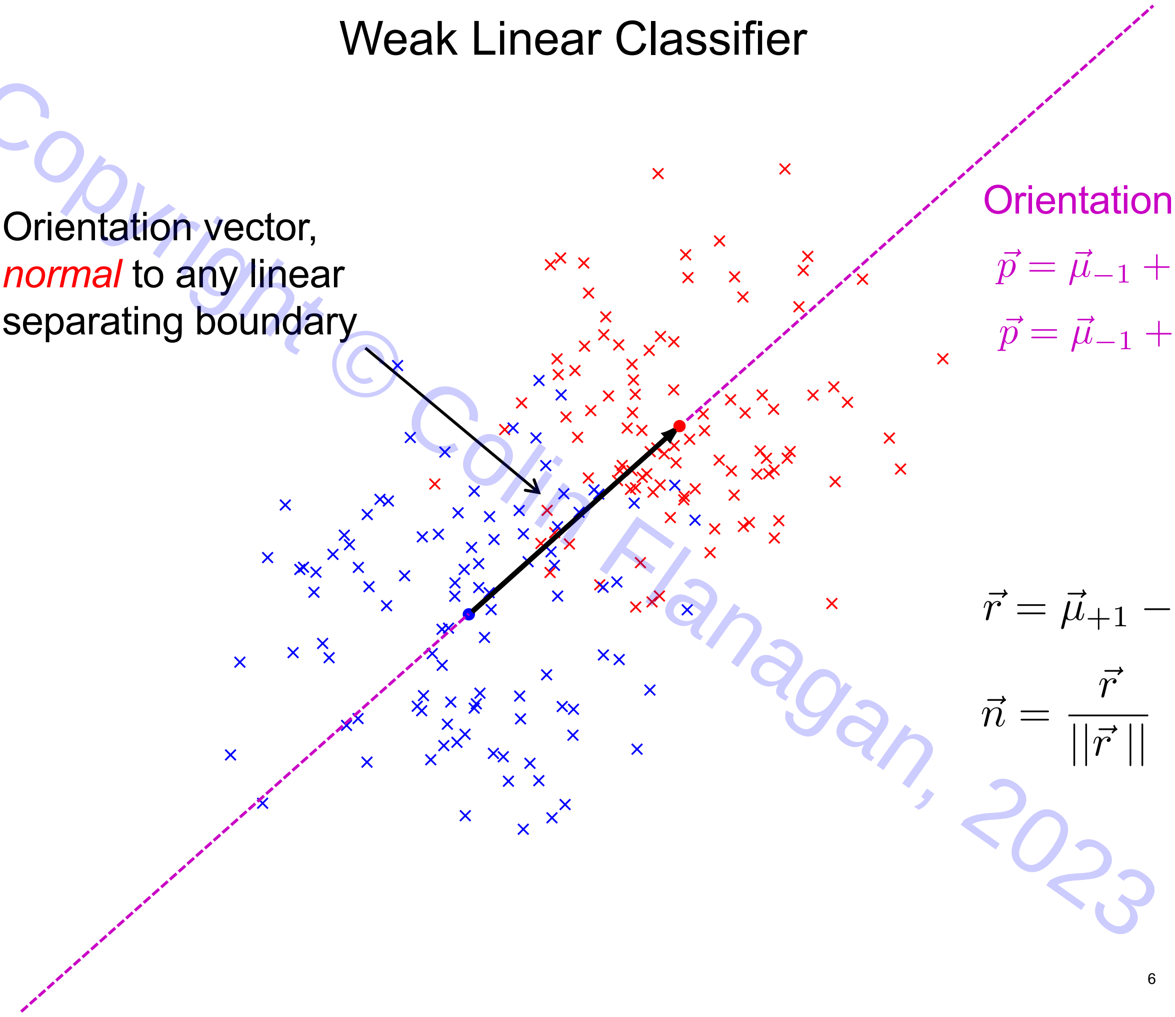
Orientation (difference) vector

$$\vec{n} = \frac{\vec{r}}{||\vec{r}\,||}$$

Normalised orientation vector

Weak Linear Classifier

Orientation vector, *normal* to any linear separating boundary



Orientation line

$$\vec{p} = \vec{\mu}_{-1} + \alpha\,\vec{r}$$

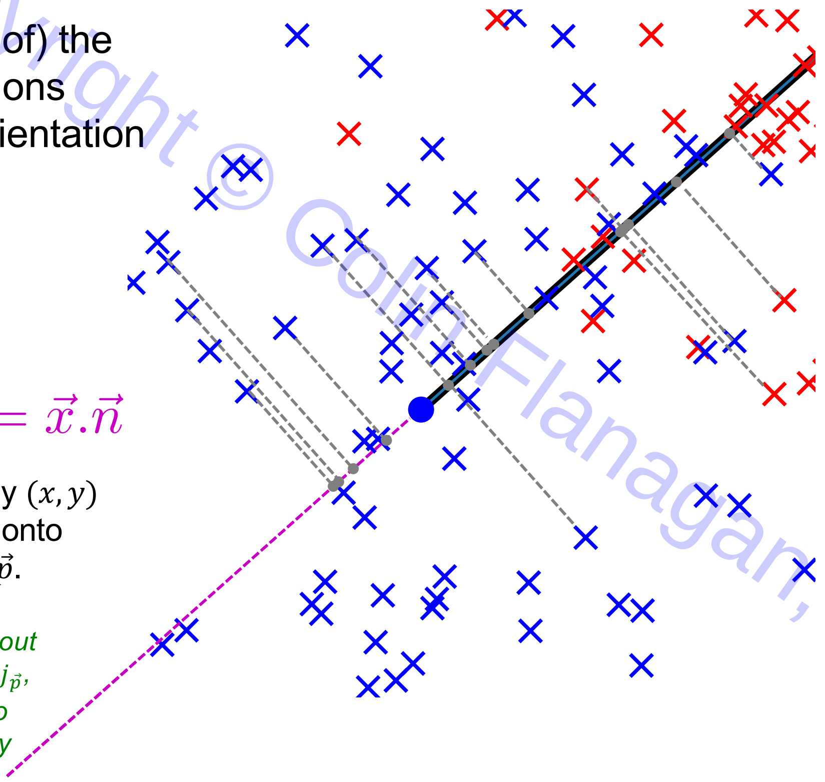
$$\vec{p} = \vec{\mu}_{-1} + \beta\,\vec{n}$$

$$\vec{r} = \vec{\mu}_{+1} - \vec{\mu}_{-1}$$

$$\vec{n} = \frac{\vec{r}}{||\vec{r}\,||}$$

Weak Linear Classifier

(Some of) the projections onto orientation line



Orientation line

$\vec{p} = \vec{\mu}_{-1} + \alpha\,\vec{r}$

$\vec{p} = \vec{\mu}_{-1} + \beta\,\vec{n}$

$$\text{proj}_{\vec{p}}(\vec{x}) = \vec{x}.\vec{n}$$

Projection of any (x, y) point (vector $\vec{x}$) onto orientation line $\vec{p}$.

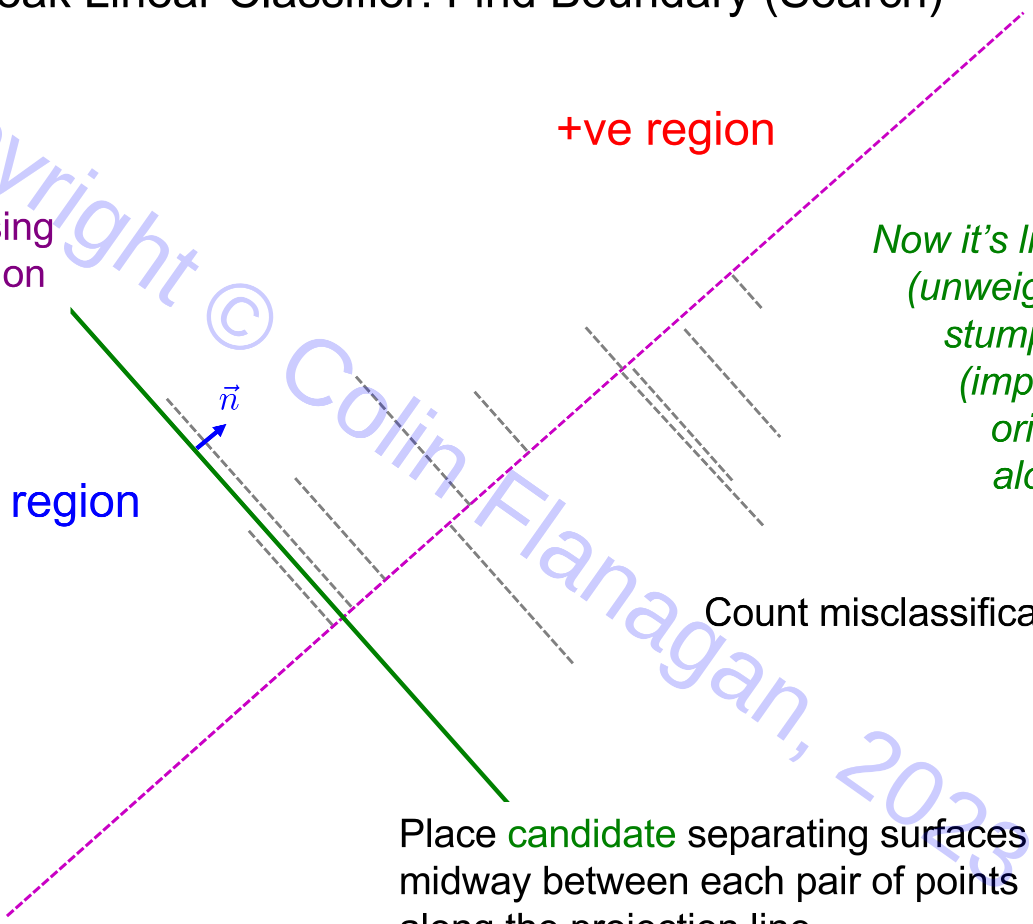
(We don't care about orientation of $proj_{\vec{p}}$, we're doing 2-d to 1-d dimensionality reduction.)

$$\vec{n} = \frac{\vec{r}}{||\vec{r}\,||}$$

Normalised orientation vector

Weak Linear Classifier: Find Boundary (Search)

Sort by increasing projection value.



+ve region

−ve region

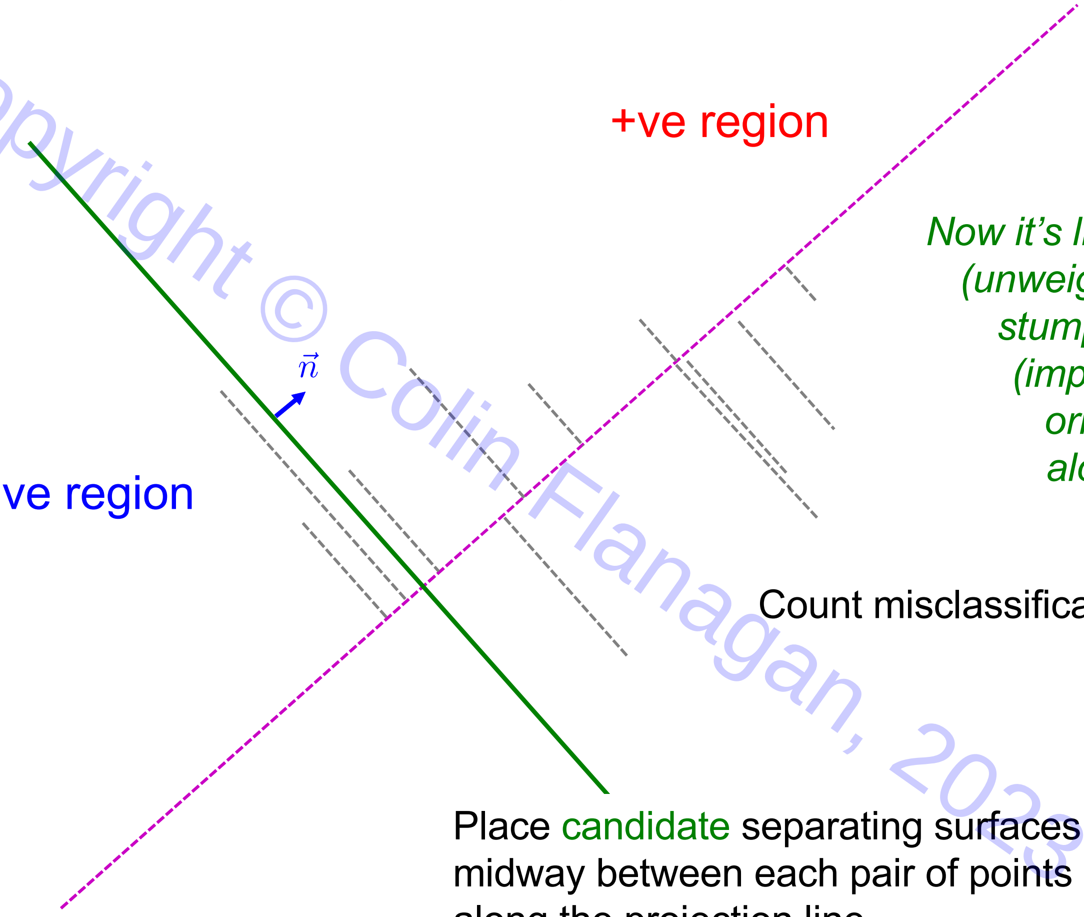
Now it's like an (unweighted) stump, just (implicitly) oriented along $\vec{p}$.

Count misclassifications.

$$\vec{n} = \frac{\vec{r}}{||\vec{r}\,||}$$

Place candidate separating surfaces midway between each pair of points along the projection line.

Weak Linear Classifier



+ve region

−ve region

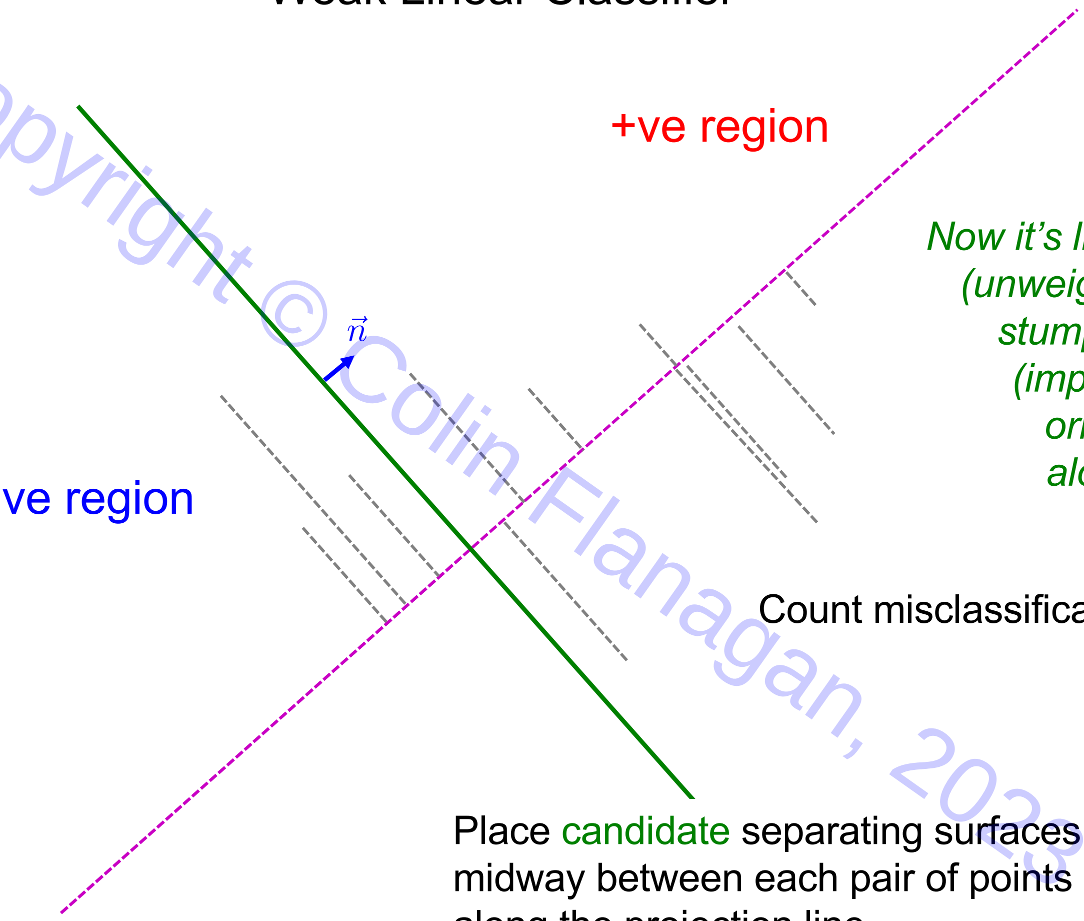
Now it's like an (unweighted) stump, just (implicitly) oriented along $\vec{p}$.

Count misclassifications.

$$\vec{n} = \frac{\vec{r}}{||\vec{r}\,||}$$

Place candidate separating surfaces midway between each pair of points along the projection line.

Weak Linear Classifier



+ve region

−ve region

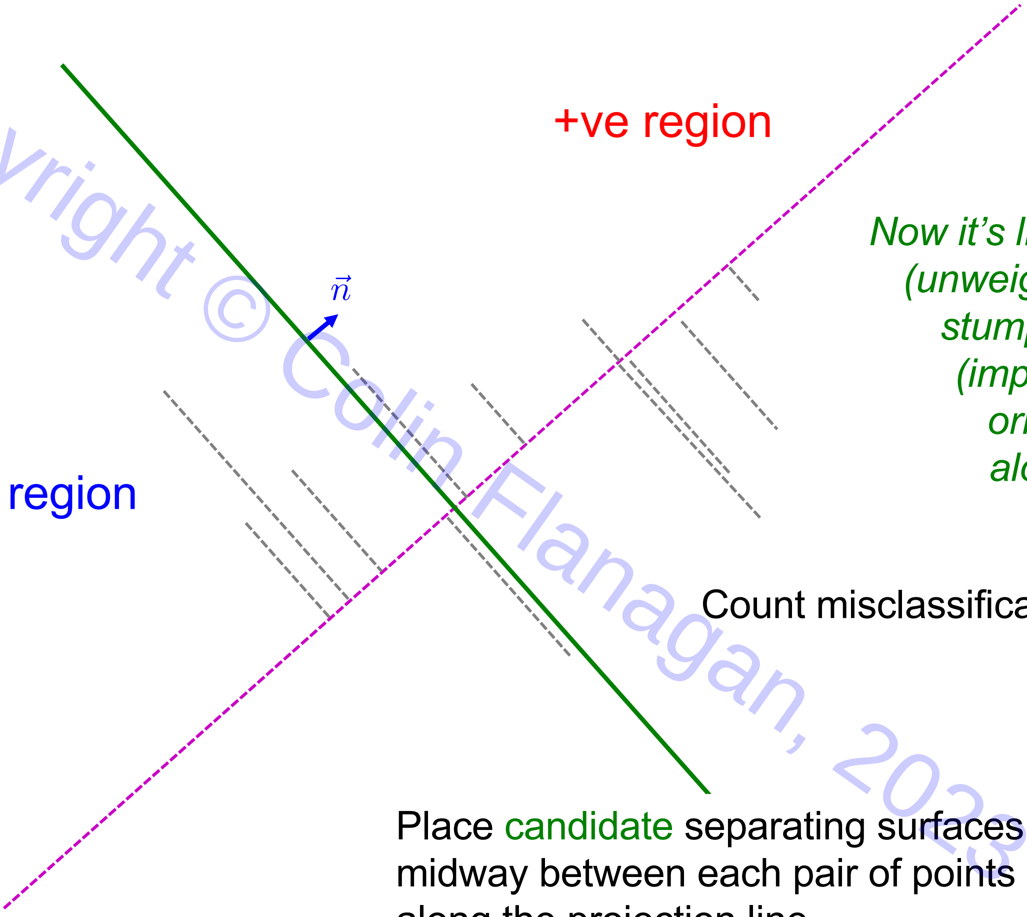
Now it's like an (unweighted) stump, just (implicitly) oriented along $\vec{p}$.

Count misclassifications.

$$\vec{n} = \frac{\vec{r}}{||\vec{r}\,||}$$

Place candidate separating surfaces midway between each pair of points along the projection line.

Weak Linear Classifier



Now it's like an (unweighted) stump, just (implicitly) oriented along $\vec{p}$.

Count misclassifications.

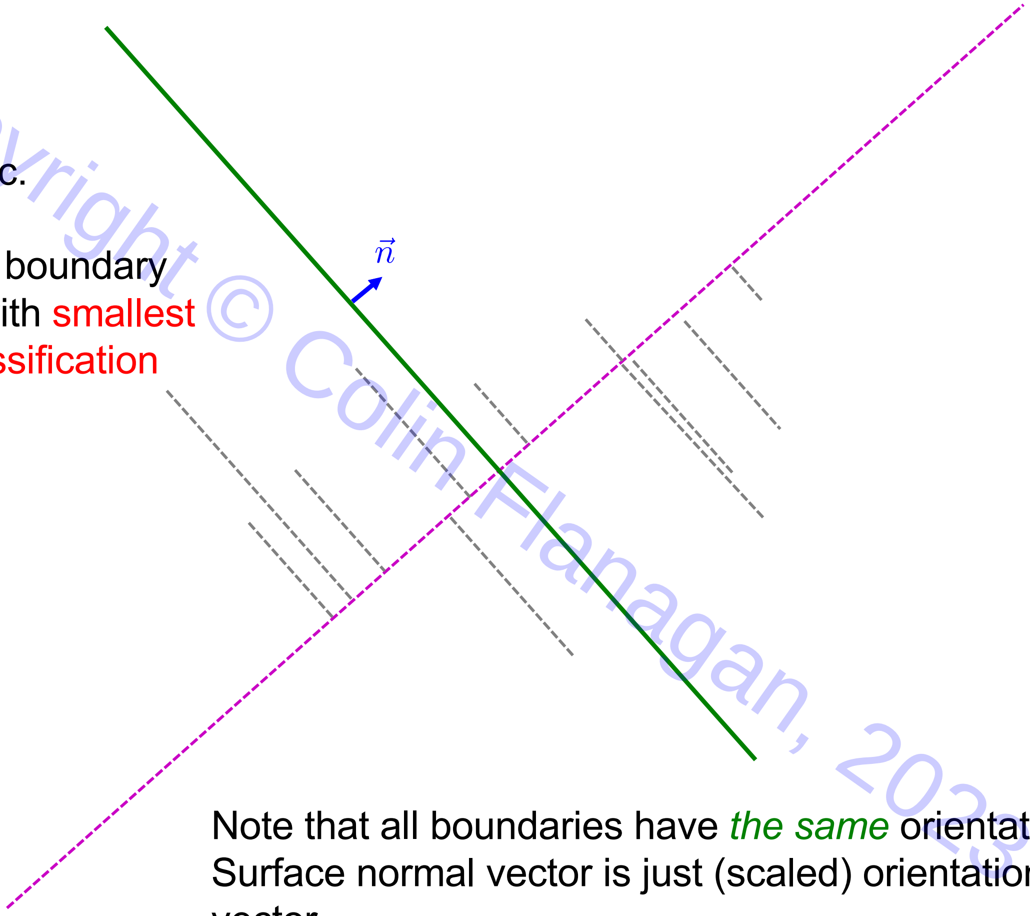
$$\vec{n} = \frac{\vec{r}}{||\vec{r}\,||}$$

Place candidate separating surfaces midway between each pair of points along the projection line.

Weak Linear Classifier

Etc., etc.

Return boundary point with smallest misclassification rate.



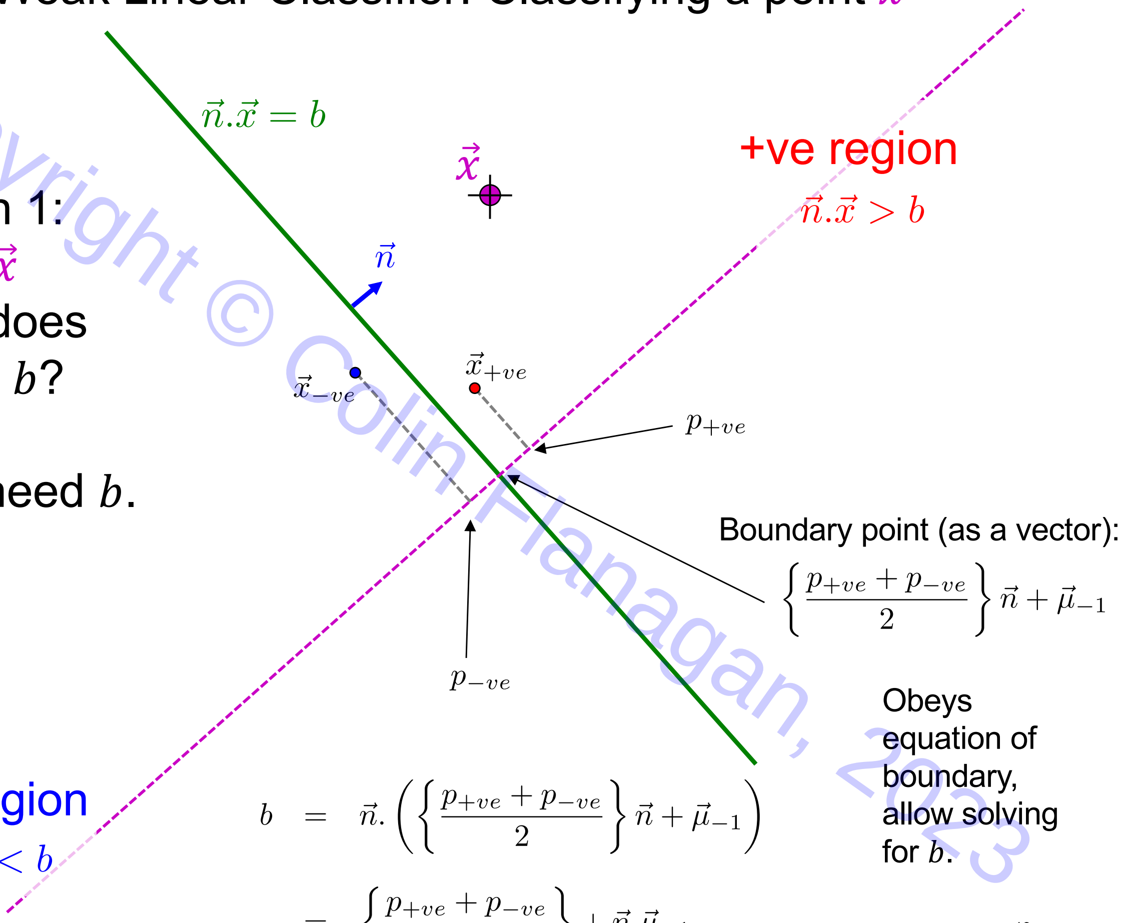
$$\vec{n} = \frac{\vec{r}}{||\vec{r}\,||}$$

Note that all boundaries have *the same* orientation. Surface normal vector is just (scaled) orientation vector.

Weak Linear Classifier: Classifying a point $\vec{x}$

Approach 1: check $\vec{n}.\vec{x}$ *directly*, does it exceed b?

For this need b.



Boundary point (as a vector):

$$\left\{\frac{p_{+ve} + p_{-ve}}{2}\right\}\vec{n} + \vec{\mu}_{-1}$$

Obeys equation of boundary, allow solving for b.

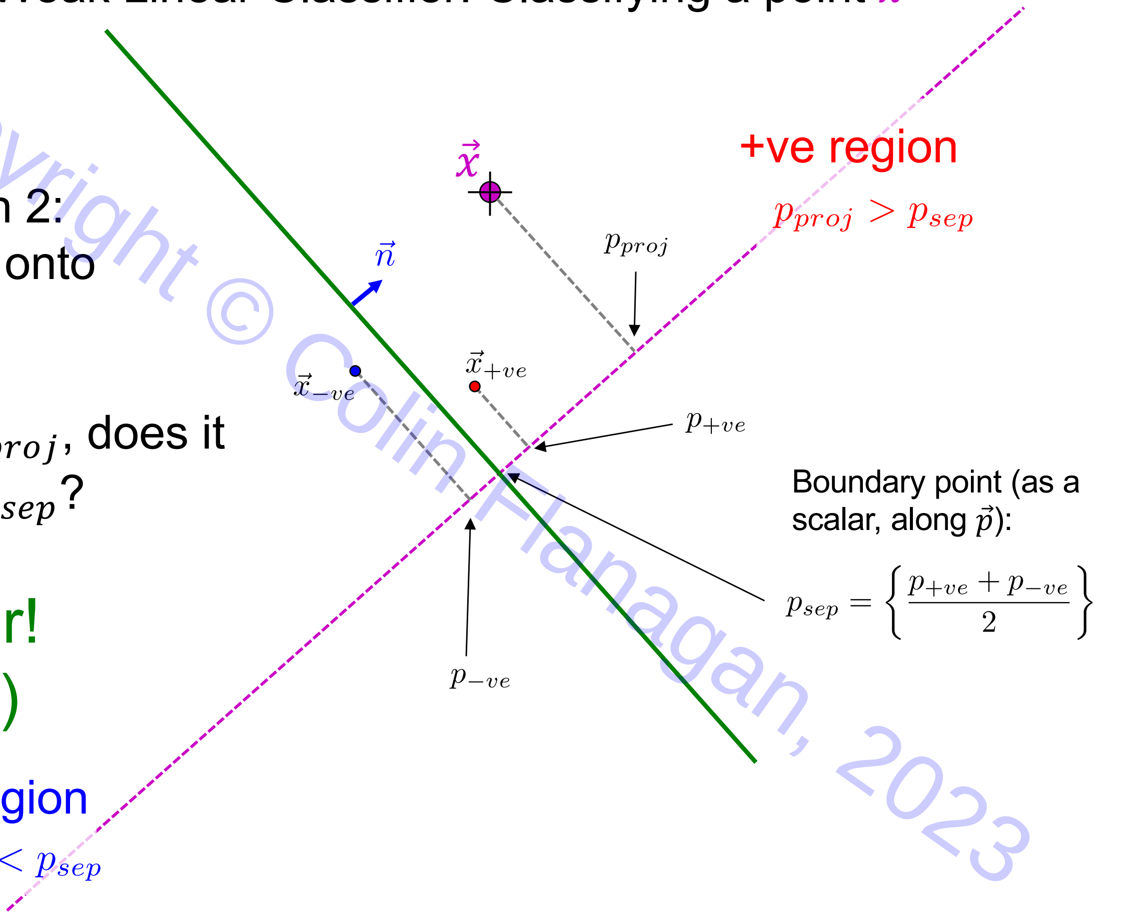
$$\begin{aligned} b &= \vec{n}.\left(\left\{\frac{p_{+ve} + p_{-ve}}{2}\right\}\vec{n} + \vec{\mu}_{-1}\right) \\ &= \left\{\frac{p_{+ve} + p_{-ve}}{2}\right\} + \vec{n}.\vec{\mu}_{-1} \end{aligned}$$

Weak Linear Classifier: Classifying a point $\vec{x}$

Approach 2: project $\vec{x}$ onto line $\vec{p}$.

Check p_{proj}, does it exceed p_{sep}?

Simpler! (Better)



Boundary point (as a scalar, along $\vec{p}$):

$$p_{sep} = \left\{\frac{p_{+ve} + p_{-ve}}{2}\right\}$$

Weighted Weak Linear Classifier

2 changes relative to basic weak linear classifier:

1. Use (AdaBoost supplied) *weights* for this round to determine the *weighted* class means.

2. Use *weights* of misclassified instances when calculating the (weighted) misclassification error.

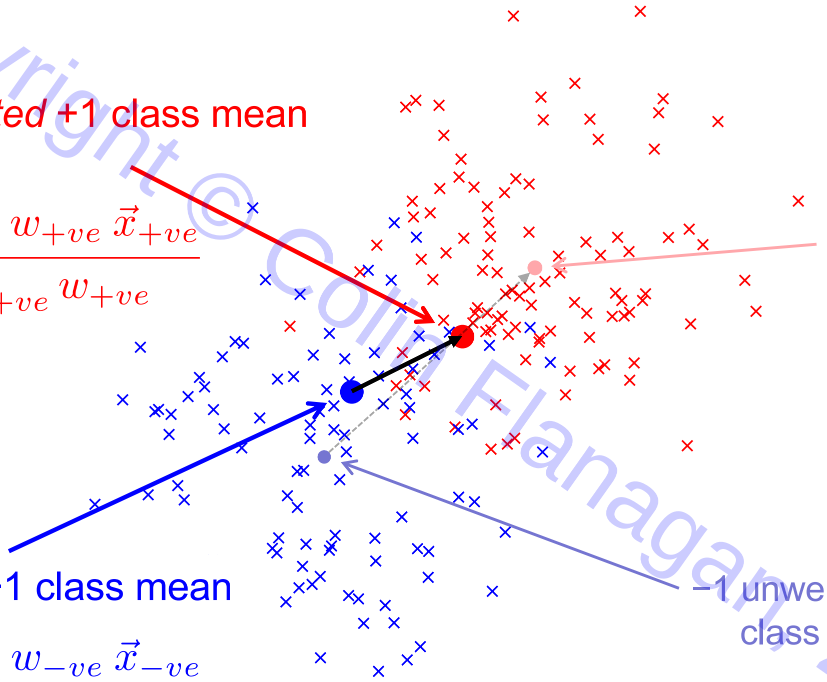
Point 1 is vital to allow the changing weights distributions generated by AdaBoost to change the *orientations* of the weak linear classifiers.

Point 2 is clearly needed for AdaBoost to work at all.

Weighted Weak Linear Classifier

Weighted +1 class mean

$$\vec{\mu}_{+1_w} = \frac{\sum_{+ve} w_{+ve}\, \vec{x}_{+ve}}{\sum_{+ve} w_{+ve}}$$



+1 unweighted class mean

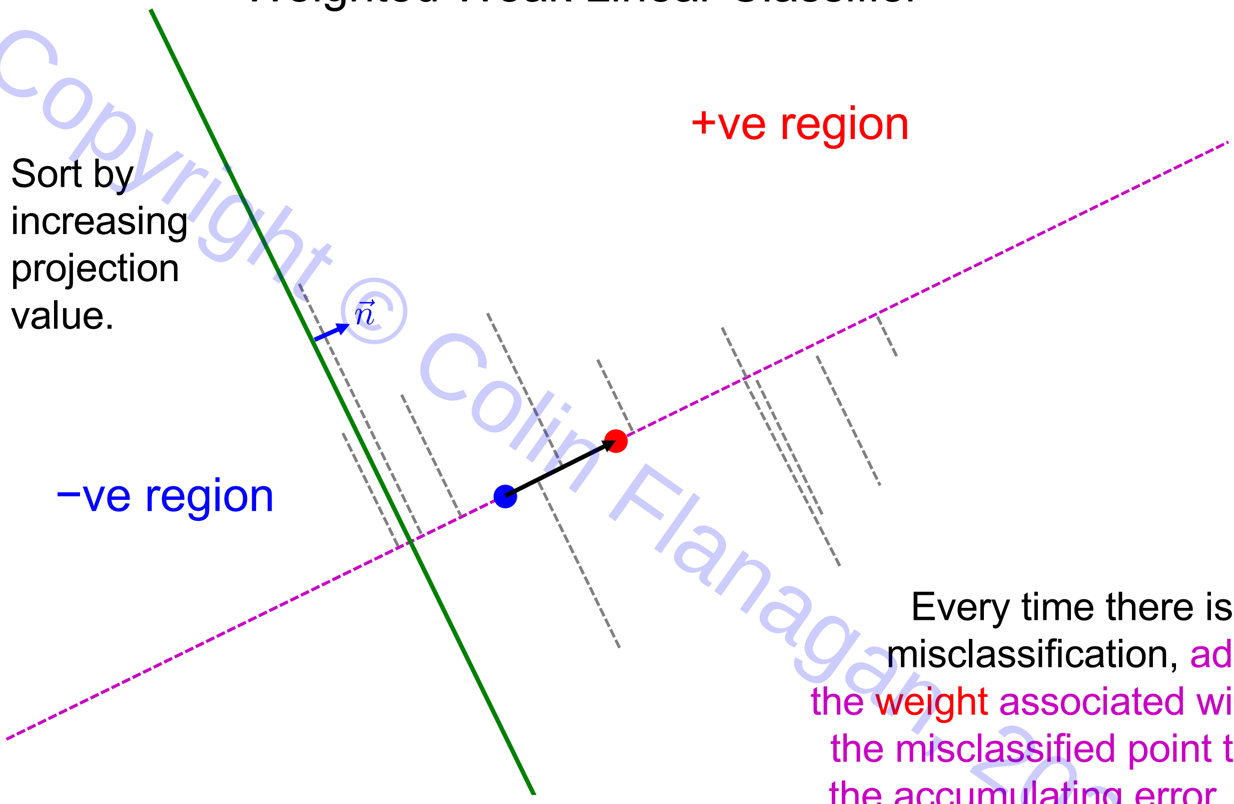
Weighted −1 class mean

$$\vec{\mu}_{-1_w} = \frac{\sum_{-ve} w_{-ve}\, \vec{x}_{-ve}}{\sum_{-ve} w_{-ve}}$$

−1 unweighted class mean

Note how the *orientation vector changes* when using weighted class means as opposed to unweighted.

Weighted Weak Linear Classifier

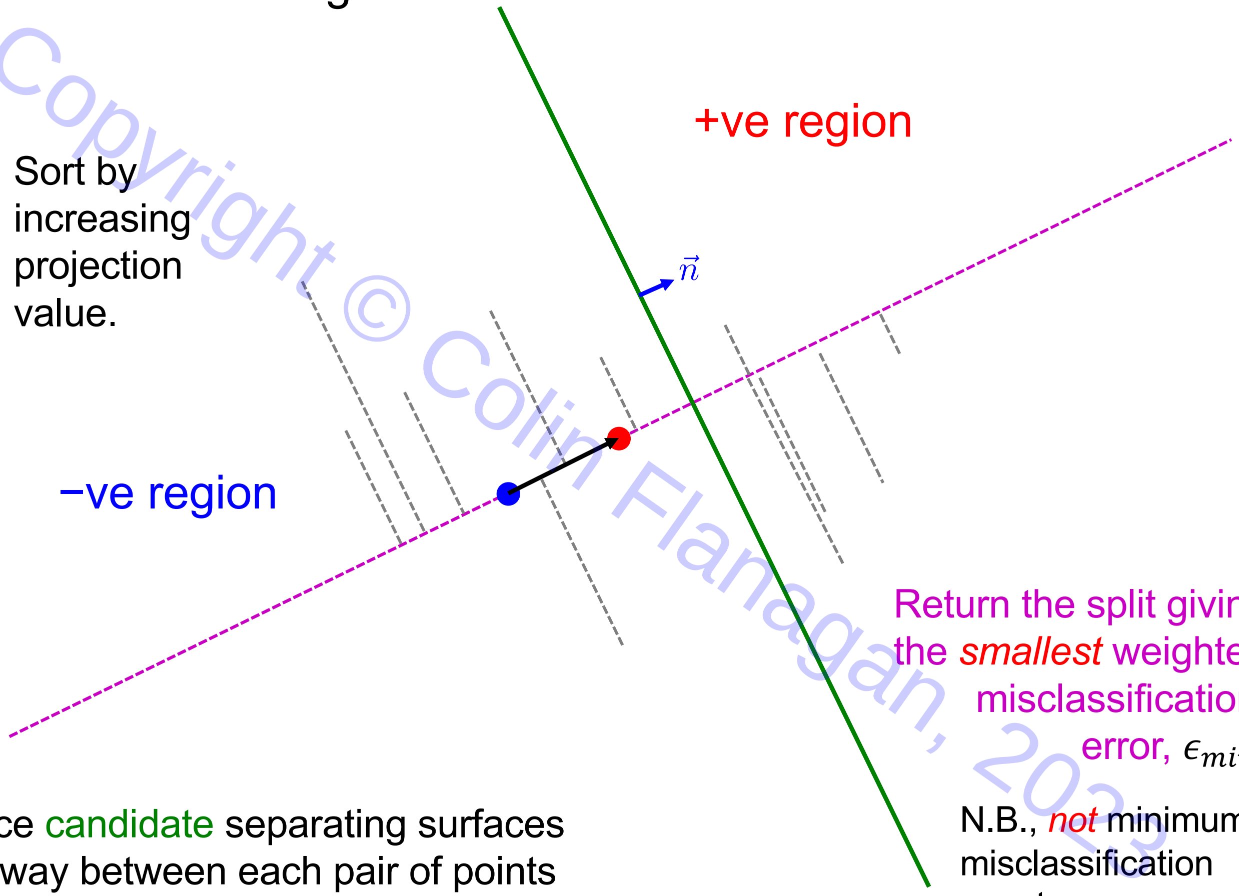


Sort by increasing projection value.

Place candidate separating surfaces midway between each pair of points along the projection line.

Every time there is a misclassification, add the weight associated with the misclassified point to the accumulating error, ϵ.

Weighted Weak Linear Classifier



Sort by increasing projection value.

Place candidate separating surfaces midway between each pair of points along the projection line.

Return the split giving the *smallest* weighted misclassification error, ϵ_{min}.

N.B., *not* minimum misclassification count.