In what follows, remember:

X: collection of all data (points);
x_i for an individual data point
y: collection of targets (answers, or the label for each class/cluster);
y_i for the target (class) of an individual point.

For example, for handwritten digits:

- x_i is the image or the array of pixel values;
- y_i is the digit ("target", "label"), or the class/cluster the image belongs

  - X: is collection of all x_i's (an array of arrays);
  - y: is collection of all y_i's (an array of values — in this case: (0, 1, 2, …, 9))

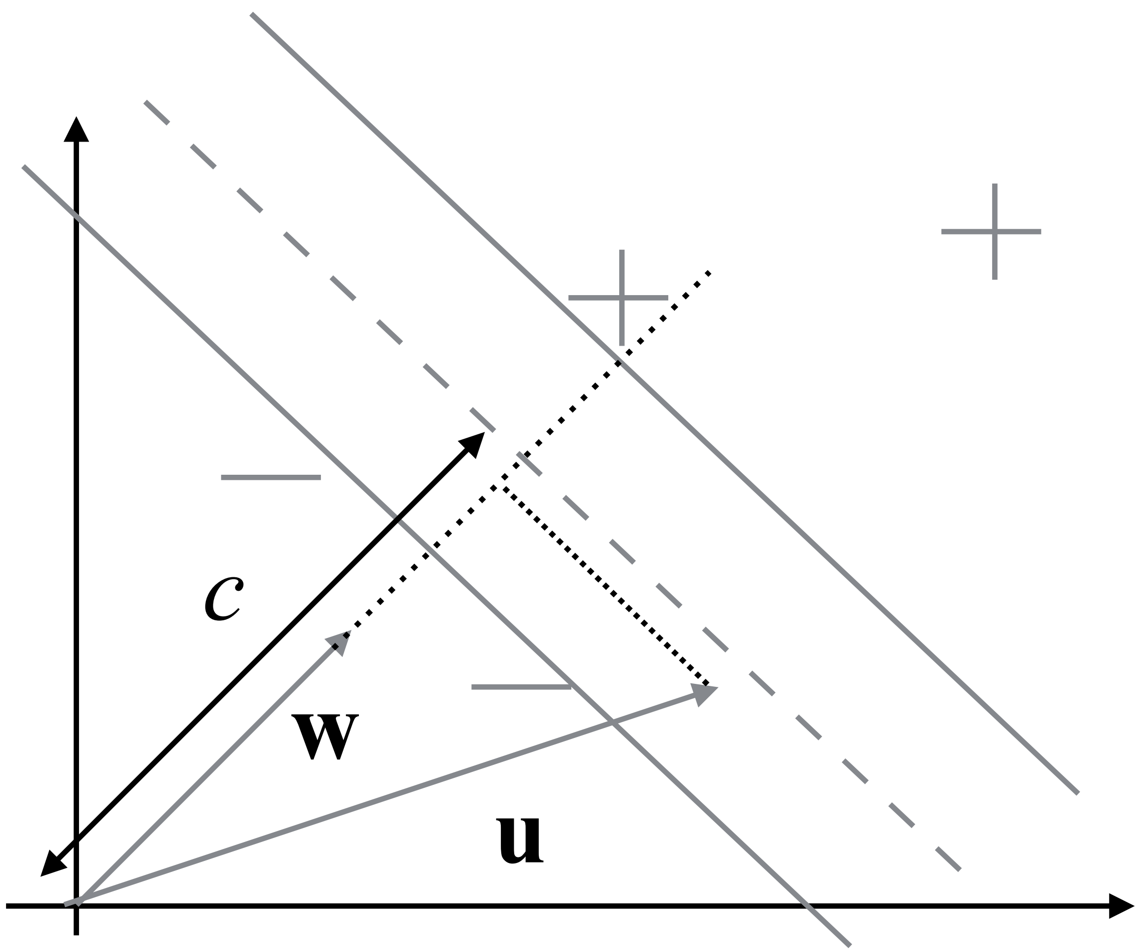
Let's start with the simplest kind of classification: only two classes.
You can call them (0, 1), or, (+, –).

Support Vector Machine

Support Vector Machine (SVM) and Decision Boundaries

Decision boundaries: The widest street approach

w is the normal to the boundaries (or the meridian of the gap).



w: decision vector

u: the vector associated with a point you would like to classify

$\mathbf{u}\cdot\mathbf{w} \geq c,$ **u** belongs to +

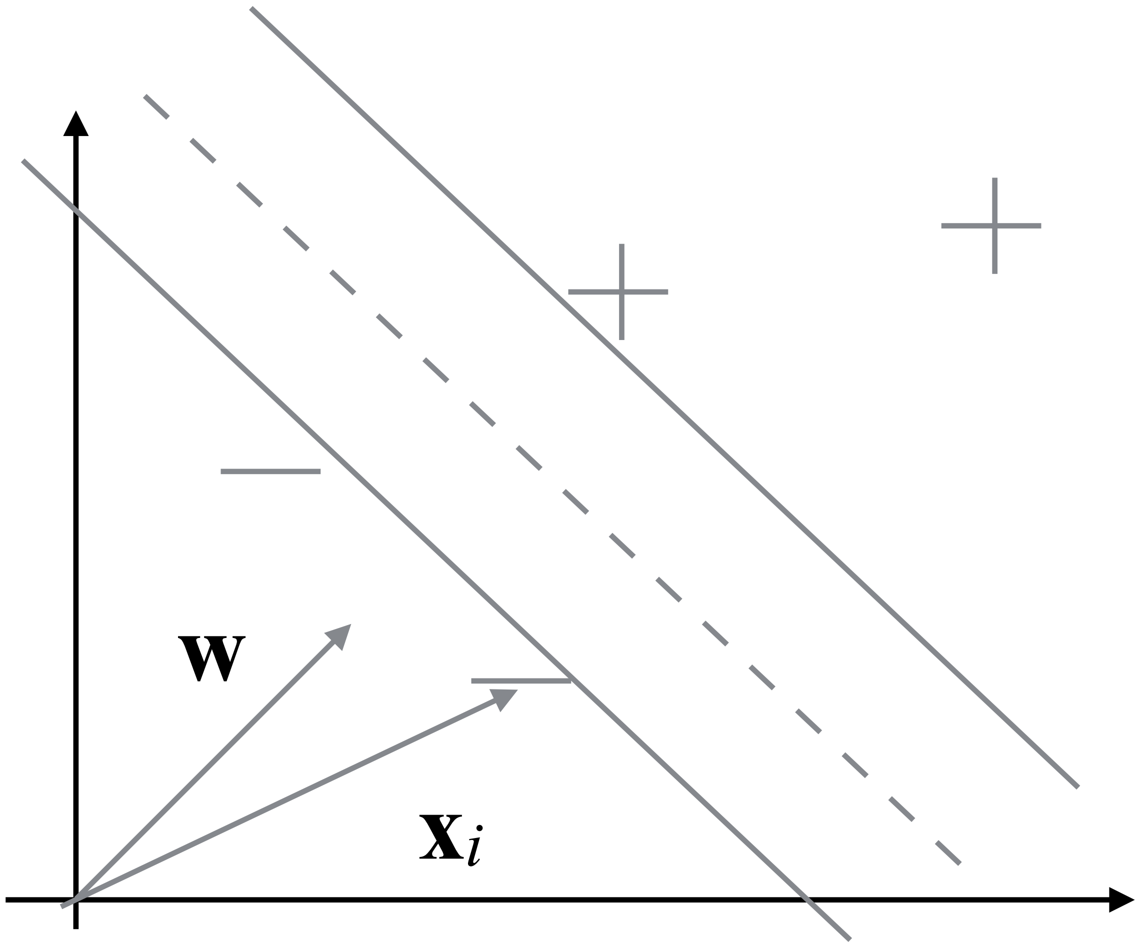
Otherwise,   **u** belongs to –

$\mathbf{u}\cdot\mathbf{w} + b \geq 0$ $b = -c$

"Decision Function"

- What is the **w** that maximizes the width of the street?
- Once we know **w**, we also have to find c that puts the meridian equidistant from the edges

Starting with "training data"



For all the training data, we require:

$$\mathbf{x}_+ \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_- \cdot \mathbf{w} + b \leq -1$$

Define y_i (targets):

$y_i = +1$ for +ve points

$y_i = -1$ for –ve points

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1 \qquad (1)$$

for all points

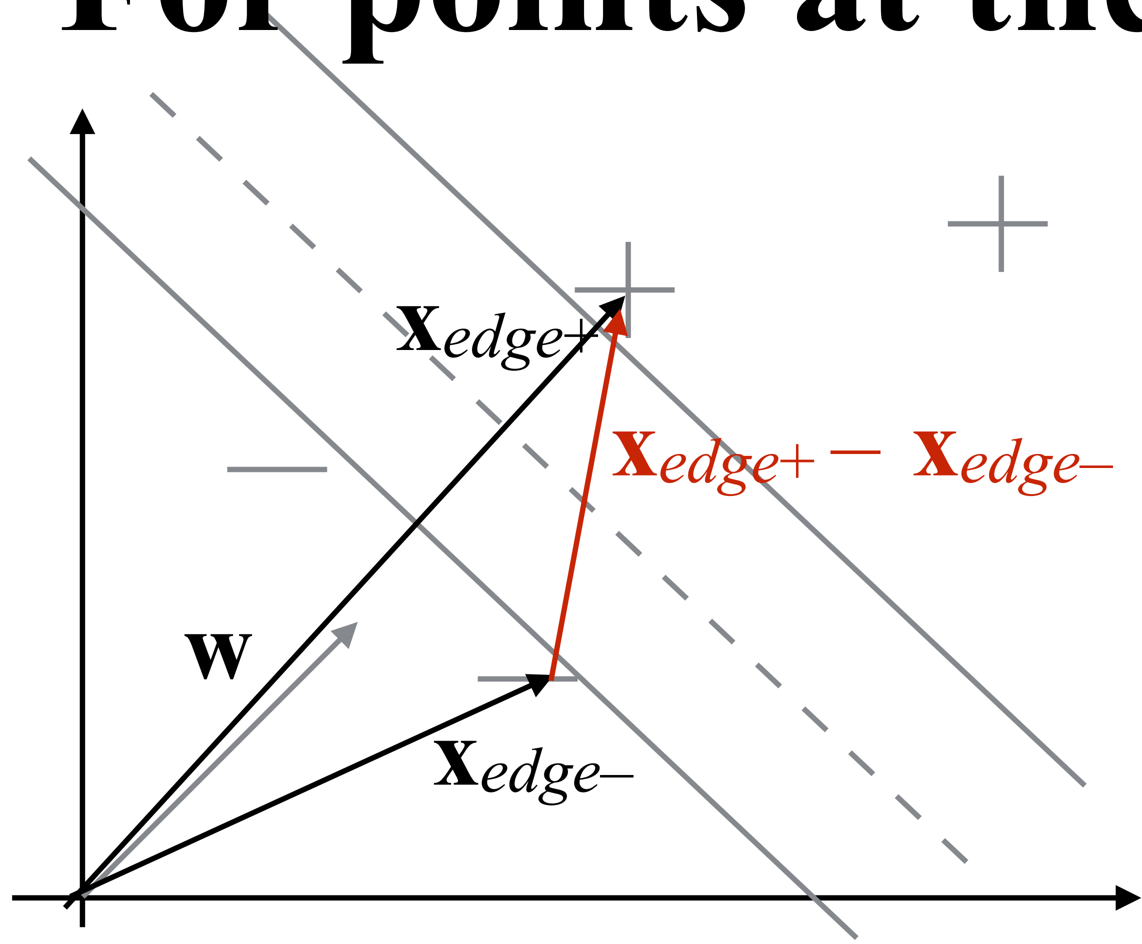
For training data points at the edge, demand

$$y_i(\mathbf{x}_{edge,\,i} \cdot \mathbf{w} + b) = 1$$

Or

$$\boxed{y_i(\mathbf{x}_{edge,\,i} \cdot \mathbf{w} + b) - 1 = 0}$$

For points at the edges of the street



Using points at the edge, we can figure out the width of the street:

$$\text{width} = (\mathbf{x}_{edge+} - \mathbf{x}_{edge-}) \cdot \frac{\mathbf{w}}{|\mathbf{w}|}$$

But $\qquad \mathbf{x}_{edge,i} \cdot \mathbf{w} = y_i - b$

So $\qquad \mathbf{x}_{edge+} \cdot \mathbf{w} = 1 - b$

$$\mathbf{x}_{edge-} \cdot \mathbf{w} = -1 - b$$

$$\boxed{y_i(\mathbf{x}_{edge,i} \cdot \mathbf{w} + b) - 1 = 0}$$

$$\rightarrow \mathbf{x}_{edge,i} \cdot \mathbf{w} + b = y_i$$

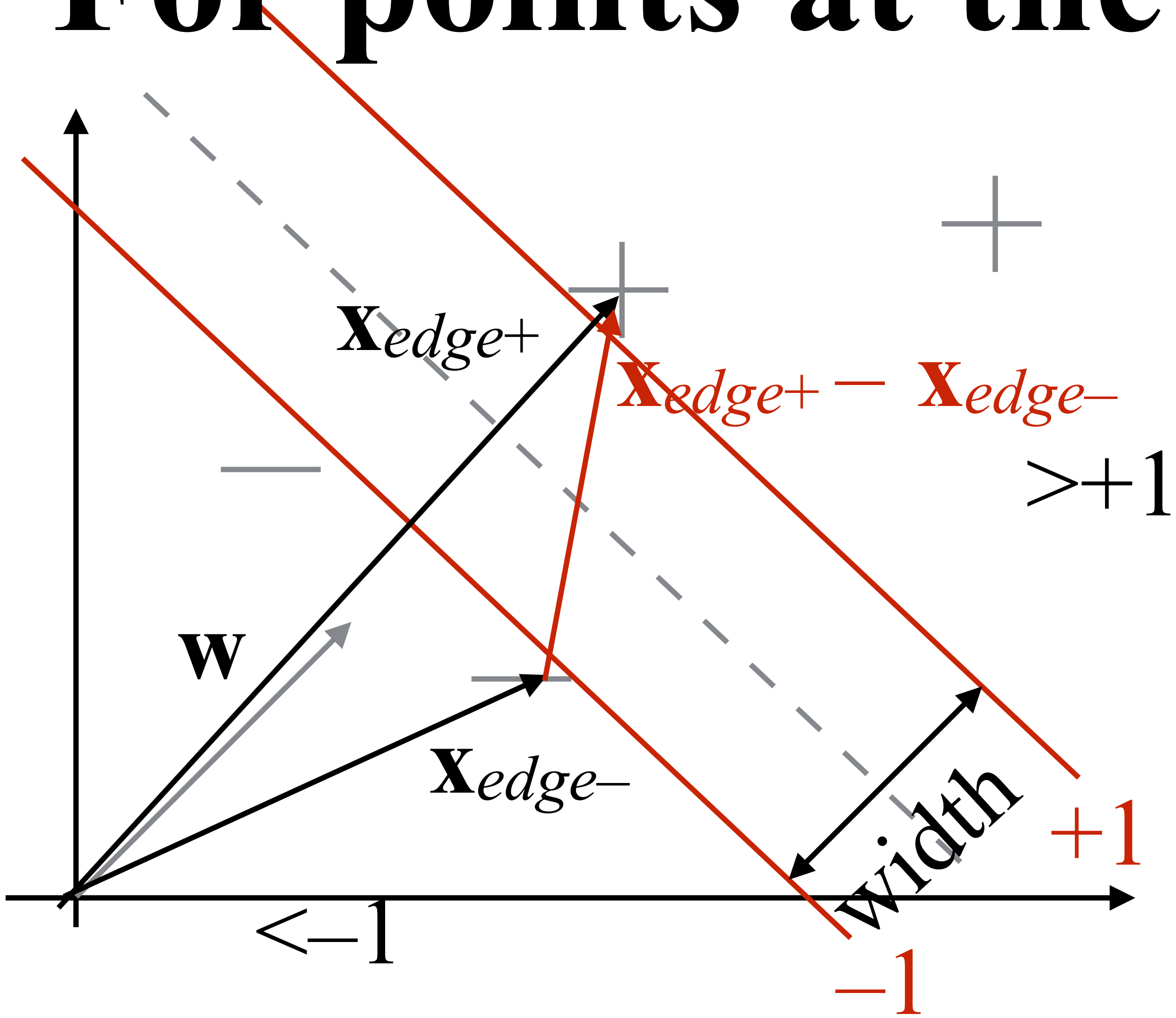
$$\mathbf{x}_{edge,i} \cdot \mathbf{w} = y_i - b$$

$$\text{width} = (1 - b + 1 + b) \cdot \frac{1}{|\mathbf{w}|}$$

$$= \frac{2}{|\mathbf{w}|}$$

Training data at the edges: they constitute the **Support Vectors.**

For points at the edges of the street



$$\text{width} = \frac{2}{|\mathbf{w}|}$$

Max width
$\rightarrow$min $|\mathbf{w}|$
$\rightarrow$ $\boxed{\text{min } \frac{1}{2} |\mathbf{w}|^2}$ The creativity of a mathematician

$$\boxed{y_i(\mathbf{x}_{edge,\,i} \cdot \mathbf{w} + b) - 1 = 0}$$

λ_i's: Lagrange's multipliers

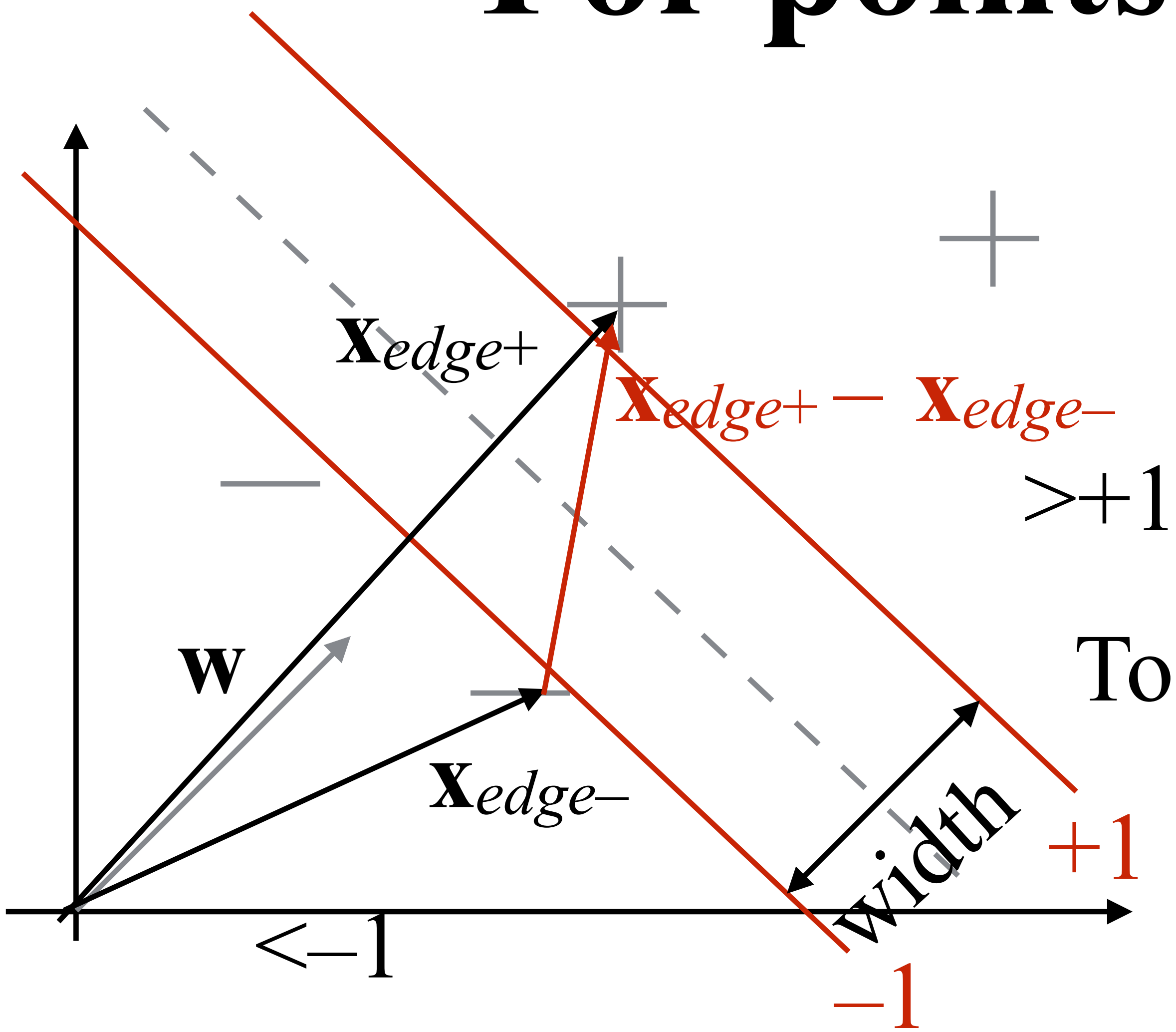
$$\boxed{L = \frac{1}{2}|\mathbf{w}|^2 - \sum \lambda_i \Big[\underbrace{y_i(\mathbf{x}_{edge,\,i} \cdot \mathbf{w} + b) - 1}_{=0}\Big]}$$

$$= \frac{1}{2}\mathbf{w} \cdot \mathbf{w} - \sum \lambda_i \left[y_i(\mathbf{x}_{edge,\,i} \cdot \mathbf{w} + b) - 1 \right]$$

So the λ_i's can be anything!

Then what good are they? You'll see!

For points at the edges



$$\boxed{L = \frac{1}{2}|\mathbf{w}|^2 - \sum \lambda_i \left[y_i(\mathbf{x}_{edge,\,i} \cdot \mathbf{w} + b) - 1 \right]}$$

$$|\mathbf{w}|^2 = \mathbf{w} \cdot \mathbf{w}$$

To minimize L, take derives w.r.t. $\mathbf{w}$ and b

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum \lambda_i y_i \mathbf{x}_{g,\,i} = 0$$

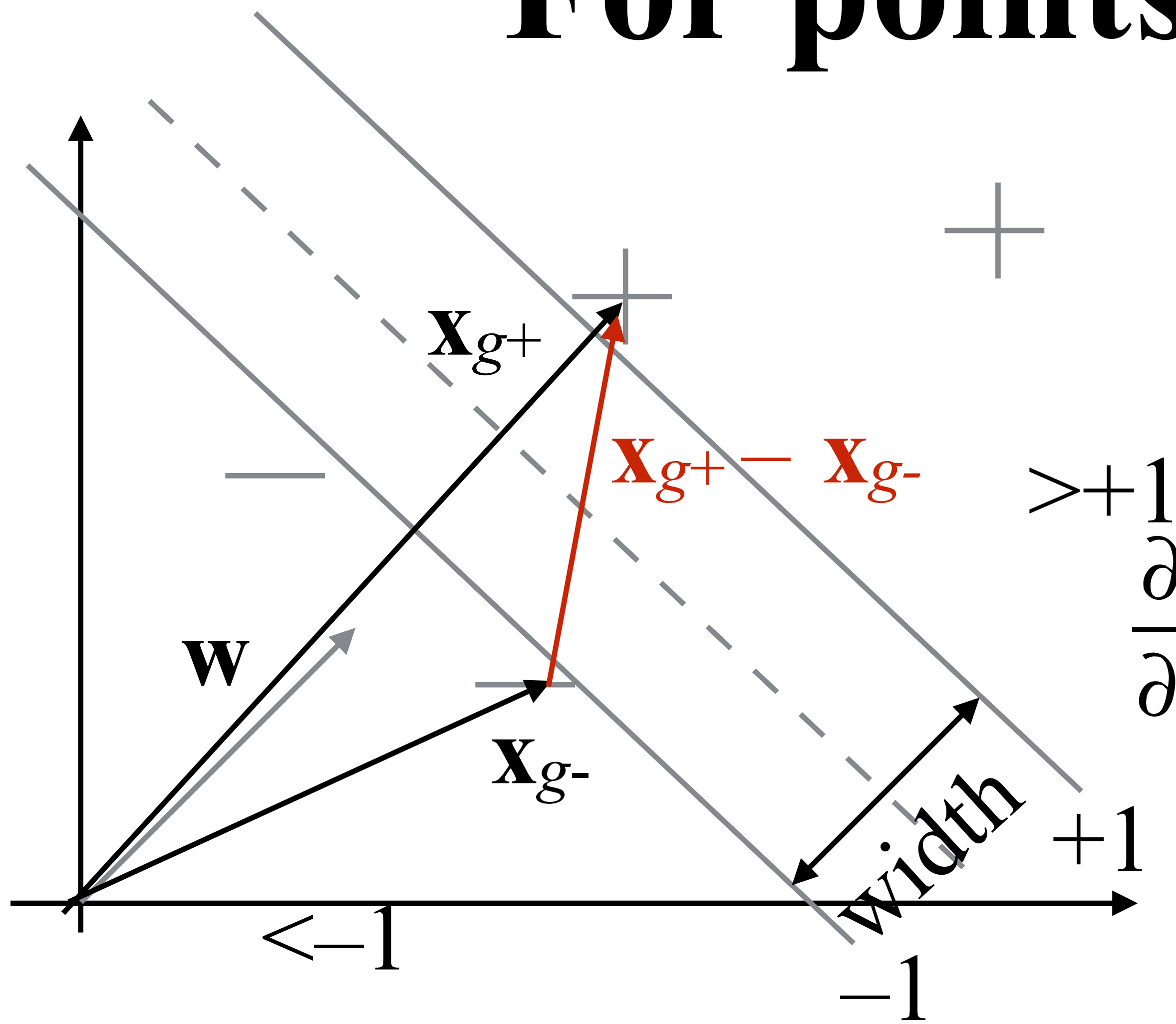
$$\boxed{y_i(\mathbf{x}_{edge,\,i} \cdot \mathbf{w} + b) - 1 = 0}$$

$$\mathbf{w} = \sum \lambda_i y_i \mathbf{x}_{g,\,i}$$

Decision vector is a linear sum of the sample points (for non-edge points, $\lambda_i = 0$)

$$\frac{\partial L}{\partial b} = -\sum \lambda_i y_i = 0 \qquad \text{or} \sum \lambda_i y_i = 0$$

For points at the edges



$$L = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \sum_i \lambda_i \left[y_i(\mathbf{x}_{edge,i}\cdot\mathbf{w}+b)-1 \right]$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum \lambda_i y_i \mathbf{x}_{edge,i} = 0 \quad \rightarrow \mathbf{w} = \sum \lambda_i y_i \mathbf{x}_{edge,i}$$

$$\frac{\partial L}{\partial b} = -\sum \lambda_i y_i = 0 \quad \rightarrow \sum \lambda_i y_i = 0$$

L is an extremum if $\mathbf{w} = \sum \lambda_i y_i \mathbf{x}_{edge,i}$ and $\sum \lambda_i y_i = 0$

[For edge point, the λ_i are to be determined.]

If $\lambda_i = 0$ for non-edge points, then can drop "*edge*" in subscript:

$$L = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + \sum_i \lambda_i \left[y_i(\mathbf{x}_i\cdot\mathbf{w}+b)-1 \right]$$

$\mathbf{w} = \sum \lambda_i y_i \mathbf{x}_i$ and $\sum \lambda_i y_i = 0$

(Minimizing conditions)

Skip, 2018

$$L = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \sum_i \lambda_i \left[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \right]$$

L is an extremum if $\quad \mathbf{w} = \sum \lambda_i y_i \mathbf{x}_i \quad (1) \qquad$ and $\sum \lambda_i y_i = 0 \quad (2)$

Let's use (1) to substitute for **w** in L

$$L = \frac{1}{2}\sum_i \lambda_i y_i \mathbf{x}_i \cdot \sum_j \lambda_j y_j \mathbf{x}_j - \sum_i \lambda_i \left[y_i(\mathbf{x}_i \cdot \sum_j \lambda_j y_j \mathbf{x}_j + b) - 1 \right]$$

$$\sum_i \lambda_i y_i \mathbf{x}_i \cdot \sum_j \lambda_j y_j \mathbf{x}_j + b\sum \lambda_i y_i - \sum \lambda_i$$

0, by eqn (2)

At this point, both conditions (1) and (2) are satisfied, thus the value of L is an extremum

$$L_{extrm} = \frac{1}{2}\sum_i \lambda_i y_i \mathbf{x}_i \cdot \sum_j \lambda_j y_j \mathbf{x}_j - \sum_i \lambda_i y_i \mathbf{x}_i \cdot \sum_j \lambda_j y_j \mathbf{x}_j + \sum \lambda_i$$

Skip, 2018

$$L_{extrm} = \frac{1}{2}\sum_i \lambda_i y_i \mathbf{x}_i \cdot \sum_j \lambda_j y_j \mathbf{x}_j - \sum_i \lambda_i y_i \mathbf{x}_i \cdot \sum_j \lambda_j y_j \mathbf{x}_j + \sum_i \lambda_i$$

$$L_{extrm} = -\frac{1}{2}\sum_i \lambda_i y_i \mathbf{x}_i \cdot \sum_j \lambda_j y_j \mathbf{x}_j + \sum_i \lambda_i$$

$$= \sum_i \lambda_i - \frac{1}{2}\sum_i \sum_j \lambda_i y_i \mathbf{x}_i \cdot \lambda_j y_j \mathbf{x}_j$$

$$L_{extrm} = \sum_i \lambda_i - \frac{1}{2}\sum_i \sum_j \lambda_i y_i \lambda_j y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

We know y_i (+1 or –1). We can do $\mathbf{x}_i \cdot \mathbf{x}_j$.

(targets) (data)

Since $\mathbf{x}_i \cdot \mathbf{x}_j$ and y_i are fixed, you can consider L to be a function of λ_i, i.e., if you choose the right λ_i's, L will be minimized.

You can think of L as a hypersurface in a space defined by λ_i. The surface is convex thus is guaranteed to have *one, global minimum.*

Skip, 2018

$$L_{extrm} = \sum_i \lambda_i - \frac{1}{2}\sum_i \sum_j \lambda_i y_i \lambda_j y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

The task is to find the right set of λ_i's that minimizes L, which then minimizes $\frac{1}{2}|\mathbf{w}|^2$.

Remember, for non-boundary points, λ_i should be 0 — therefore in the process of determining the values of λ_i, you will also find out which ones are, and which ones are NOT, at the boundary, which in turns tells you how to draw the boundaries.

Once the λ_i's are found, one can determine **w**, and b:

$$\mathbf{w} = \sum \lambda_i y_i \mathbf{x}_i \qquad \boxed{y_i(\mathbf{x}_{edge,i} \cdot \mathbf{w} + b) - 1 = 0}$$ (from slide 5)

Finally, we can determine which group **u** belongs to (think of **u** as unknown):

$\mathbf{u} \cdot \mathbf{w} + b \geq 0$ **u** belongs to +; otherwise, –."Decision Function" (slide 1)

$$\mathbf{u} \cdot \sum_i \lambda_i y_i \mathbf{x}_i + b \geq 0 \qquad \sum_i \lambda_i y_i \mathbf{u} \cdot \mathbf{x}_i + b \geq 0$$

Significance of the Lagrange's Multipliers

Once the λ_i's are found, one can determine **w**, and b:

$$\boxed{\mathbf{w} = \sum \lambda_i y_i \mathbf{x}_i} \qquad \boxed{y_i(\mathbf{x}_{edge,i} \cdot \mathbf{w} + b) - 1 = 0}$$

The λ_i's are the weights: They tell how important each data point ($\mathbf{x}_i$) is in determining the boundaries:

- Interior points have zero weight — as they should
- Boundary points all have weights of 1
- Points that are close to the boundary get non-zero weights but less than 1 (will show example in notebook)

$$\sum_i \lambda_i y_i \mathbf{u}\cdot\mathbf{x}_i + b \geq 0$$ "Decision Function"

Remember, again, λ_i and y_i are known. All one has to do is to perform $\mathbf{u}\cdot\mathbf{x}_i$.

Summary

1. **Training:** Given data $\mathbf{x}_i$, and their classification y_i (in the simplest case, + or –), find λ_i such that

$$L_{extrm} = \sum_i \lambda_i - \frac{1}{2}\sum_i\sum_j \lambda_i y_i \lambda_j y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

is minimized. Or the gaps between the different groups are maximized (Recall λ_i's define $\mathbf{w}$, which in term defines the boundaries. This kind of problem is standard in numerical computation — the foolproof way would be to scan the λ_i space using a grid with reasonable resolution.

2. **Testing:** Classify new observation $\mathbf{u}$, by calculating

$$\sum_i \lambda_i y_i \mathbf{u}\cdot\mathbf{x}_i + b$$

In the simplest case, it belongs to either the group + or –.

Both the training and the testing steps depend only on the dot product:

Training
(To find the λ_i's):

$$L_{extrm} = \sum_i \lambda_i - \frac{1}{2}\sum_i \sum_j \lambda_i y_i \lambda_j y_j \boxed{\mathbf{x}_i \cdot \mathbf{x}_j}$$

Testing
(To find where the unknown point belongs, or the "decision function"):

$$\sum_i \lambda_i y_i \boxed{\mathbf{u} \cdot \mathbf{x}_i} + b$$

Objects are of the class `sklearn.svm.SVC` has a method called `decision_function()`

The Kernel and Nonlinear Decision Boundaries

$$L_{extrm} = \sum_i \lambda_i - \frac{1}{2}\sum_i \sum_j \lambda_i y_i \lambda_j y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (3) \qquad \sum_i \lambda_i y_i \mathbf{u} \cdot \mathbf{x}_i + b \quad (4)$$

Transformation (may or may not be linear):

$$\mathbf{x}'_i = H(\mathbf{x}_i) \qquad \mathbf{x}'_i \cdot \mathbf{x}'_j = H(\mathbf{x}_i) \cdot H(\mathbf{x}_j)$$

$H(\mathbf{x}_i) \cdot H(\mathbf{x}_j)$ is a function of $\mathbf{x}_i$ and $\mathbf{x}_j$. That is,

$$\mathbf{x}'_i \cdot \mathbf{x}'_j = H(\mathbf{x}_i) \cdot H(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$$

One can then calculate and minimize L' in the transformed space.

To evaluate the decision function, eqn (4), under the transformation, all one has to do is

$$\mathbf{u}' \cdot \mathbf{x}'_i = K(\mathbf{u}, \mathbf{x}_i)$$

K: "The Kernel"

Why Transformation?

So that you can transform the data (nonlinearly if you have to, i.e., warp the space the data points are in), so that you can find a linear boundary between the different classes in the transformed space.

When you transform the data and the boundaries back to the original space, you get non-linear boundaries, as we will see in the notebook.

Example of Kernels

- The linear kernel (no transformation): $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$

- The polynomial kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)^d$$

$d = 1$: same as linear, just a scaling + a shift
$d \geq 2$: Nonlinear

- (Gaussian) Radial basis function kernel (RBF):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left[-\frac{\left(|\mathbf{x}_i| - |\mathbf{x}_j| \right)^2}{2\sigma^2} \right] \equiv \exp\left[-\gamma \left(|\mathbf{x}_i| - |\mathbf{x}_j| \right)^2 \right]$$

Effective for handwritten digits and facial recognition.

In `sklearn.svm`, the default is RBF.