Week 4: SVMs & Backpropagation

Study Guide with Full Explanations

Contents

1. Support Vector Machines (SVMs)

Note

Background. A Support Vector Machine (SVM) is a supervised learning algorithm used for classification. The key idea is to find the *best* separating hyperplane between two classes — the one that maximises the gap (margin) between the classes. Before diving into the questions, keep in mind:

- In its simplest form an SVM is a **linear** classifier.
- Using the **kernel trick**, an SVM can learn **non-linear** decision boundaries.
- For binary classification the class labels are $y \in \{-1, +1\}$.

Question 1

Jack says SVMs are famous for learning *nonlinear* functions via the kernel trick. Jill says SVMs are actually linear classifiers trained with the maximum-margin loss. Who is right?

Answer

Both are partially correct.

- **Jill is right about the foundation.** In its core form, an SVM finds a linear hyperplane $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ that separates the two classes with maximum margin.
- **Jack is right about common usage.** Because of the mathematical form of the SVM optimisation problem, it can be reformulated to use the **kernel trick**, which implicitly maps data into a high-dimensional feature space and thereby learns a non-linear boundary in the original space — while solving a linear problem internally.

In practice, when people say "SVM" they often mean the kernelised (non-linear) version. If the linear version is intended, it is explicitly called a *linear SVM*.

The Soft-Margin SVM Optimisation Problem

The (soft-margin) SVM finds a hyperplane $\mathbf{w}^\top \mathbf{x} + b$ by solving:

$$\text{minimise} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0$$

where ξ_i ("slack variables") allow some points to violate the margin, and C controls the trade-off between maximising the margin and penalising violations.

Question 2

What does index i represent? How many terms does the sum have, and how many constraints are there?

Answer

- i **indexes the training data points**. Each i refers to one labelled example $(\mathbf{x}_i, y_i)$.
- The sum $\sum_i \xi_i$ has **one term per training instance** — as many terms as there are data points.
- There are **two constraints per instance**: $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$.

Question 3

What is the value and role of y_i in the SVM formulation?

Answer

y_i is the **class label** of training instance i:

$$y_i = \begin{cases} +1 & \text{if } \mathbf{x}_i \text{ is a positive example} \\ -1 & \text{if } \mathbf{x}_i \text{ is a negative example} \end{cases}$$

Why is this useful? Without y_i, we would need two separate constraints:

$$\mathbf{w}^\top \mathbf{x}_i + b \geq 1 \quad \text{(for positives)} \qquad \mathbf{w}^\top \mathbf{x}_i + b \leq -1 \quad \text{(for negatives)}$$

By multiplying by y_i, both are combined into a *single* constraint:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$$

This is a common and elegant trick in SVM derivations.
After training, the predicted class of a new point $\mathbf{x}$ is given by:

$$\hat{y} = \text{sign}(f(\mathbf{x})) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$$

Question 4

After training, we have $\mathbf{w} = (5, 3)^\top$ and $b = -1$. Is the point $\mathbf{x}_i = (1, -1)^\top$ with label $y_i = 1$ a support vector?

Answer

A point is a **support vector** if it lies exactly on the margin, i.e. $\mathbf{w}^\top \mathbf{x}_i + b = y_i$.
Let us compute:

$$\mathbf{w}^\top \mathbf{x}_i + b = \begin{pmatrix} 5 \\ 3 \end{pmatrix}^\top \begin{pmatrix} 1 \\ -1 \end{pmatrix} - 1 = 5 \cdot 1 + 3 \cdot (-1) - 1 = 5 - 3 - 1 = \mathbf{1}$$

Since $y_i = 1$ and $\mathbf{w}^\top \mathbf{x}_i + b = 1$, the condition $\mathbf{w}^\top \mathbf{x}_i + b = y_i$ is satisfied.
Yes, $\mathbf{x}_i$ is a support vector.

Question 5

Does this mean we expect $f(\mathbf{x}_i) = y_i$ for *every* training instance?

Answer

No. The SVM is *not* trying to fit the target value y_i directly the way a regression model does. Instead, the SVM tries to ensure that each point is on the *correct side* of the margin. Only the special points sitting exactly on the margin (the support vectors) satisfy $f(\mathbf{x}_i) = y_i$. All other correctly classified points satisfy $y_i \cdot f(\mathbf{x}_i) > 1$.

Question 6

Jack thinks support vectors are instances where $f(\mathbf{x}_i)$ exactly equals y_i. Jill thinks they are instances exactly on the margin. Who is right?

Answer

Both are right — these are two descriptions of the same thing.
The **margin** of a classifier is the region between the two hyperplanes:

$$f(\mathbf{x}) = -1 \qquad \text{and} \qquad f(\mathbf{x}) = +1$$

A point lying exactly on the margin means:

- For a positive point: $f(\mathbf{x}_i) = +1 = y_i$
- For a negative point: $f(\mathbf{x}_i) = -1 = y_i$

In both cases $f(\mathbf{x}_i) = y_i$, confirming Jack's statement. In summary:

$$\boxed{\text{Support vector} \iff \mathbf{w}^\top \mathbf{x}_i + b = y_i}$$

Worked Examples with $\mathbf{w} = (1, -2)^\top$, $b = 1$

Note

For questions 7–11 we use the classifier:

$$\mathbf{w} = \begin{pmatrix} 1 \\ -2 \end{pmatrix}, \quad b = 1, \quad f(\mathbf{x}) = x_1 - 2x_2 + 1$$

Question 7

What classes are assigned to $(2,1)^\top$, $(1,2)^\top$, and $(2,2)^\top$?

Answer

Recall: if $f(\mathbf{x}) > 0$ the point is **positive**; if $f(\mathbf{x}) < 0$ it is **negative**.

Point	Calculation	$f(\mathbf{x})$	Class
$(2,1)^\top$	$1 \cdot 2 - 2 \cdot 1 + 1 = 2 - 2 + 1$	$+1$	Positive
$(1,2)^\top$	$1 \cdot 1 - 2 \cdot 2 + 1 = 1 - 4 + 1$	-2	Negative
$(2,2)^\top$	$1 \cdot 2 - 2 \cdot 2 + 1 = 2 - 4 + 1$	-1	Negative

Question 8

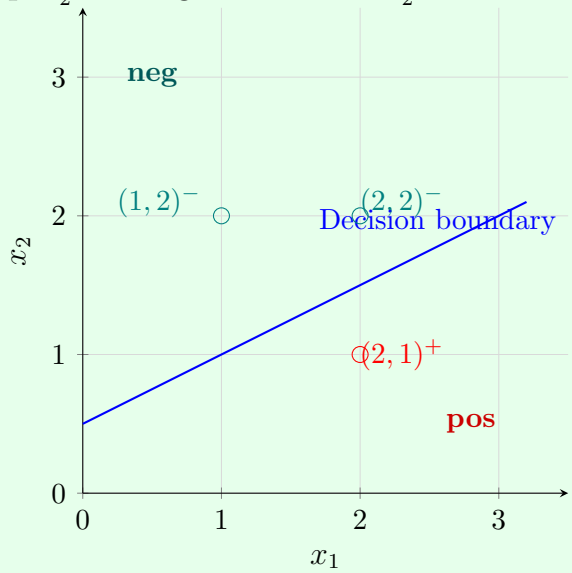
Derive and draw the decision boundary. Label the positive and negative sides.

Answer

The decision boundary is the set of points where $f(\mathbf{x}) = 0$:

$$x_1 - 2x_2 + 1 = 0 \implies x_2 = \frac{1}{2}x_1 + \frac{1}{2}$$

This is a line with **slope** $\frac{1}{2}$ crossing the x_2-axis at $\frac{1}{2}$.



Question 9

Which of the following are support vectors? $\mathbf{x}_1 = (1, 3/2)^\top$, $y_1 = 1$; $\mathbf{x}_2 = (0, 1)^\top$, $y_2 = -1$; $\mathbf{x}_3 = (1, 1)^\top$, $y_3 = 1$; $\mathbf{x}_4 = (1, 1/2)^\top$, $y_4 = 1$.

Answer

Check $\mathbf{w}^\top \mathbf{x}_i + b = y_i$ for each point:

Point	$\mathbf{w}^\top \mathbf{x}_i + b$	y_i	Equal?	Support vector?
$\mathbf{x}_1 = (1, 3/2)^\top$	$1 - 3 + 1 = -1$	$+1$	No	**No**
$\mathbf{x}_2 = (0, 1)^\top$	$0 - 2 + 1 = -1$	-1	Yes	**Yes**
$\mathbf{x}_3 = (1, 1)^\top$	$1 - 2 + 1 = 0$	$+1$	No	**No**
$\mathbf{x}_4 = (1, 1/2)^\top$	$1 - 1 + 1 = 1$	$+1$	Yes	**Yes**

So $\mathbf{x}_2$ and $\mathbf{x}_4$ are support vectors.

Note

Note about $\mathbf{x}_1$: $\mathbf{x}_1$ sits on the *negative* margin ($f(\mathbf{x}_1) = -1$) but has a *positive* label ($y_1 = 1$). This means it is on the *wrong* side of the margin — a violation. A hard-margin SVM would not be valid with this point in the dataset; a soft-margin SVM would incur a penalty $\xi_1 > 0$.

Question 10

Draw the margin edges and plot all four points.

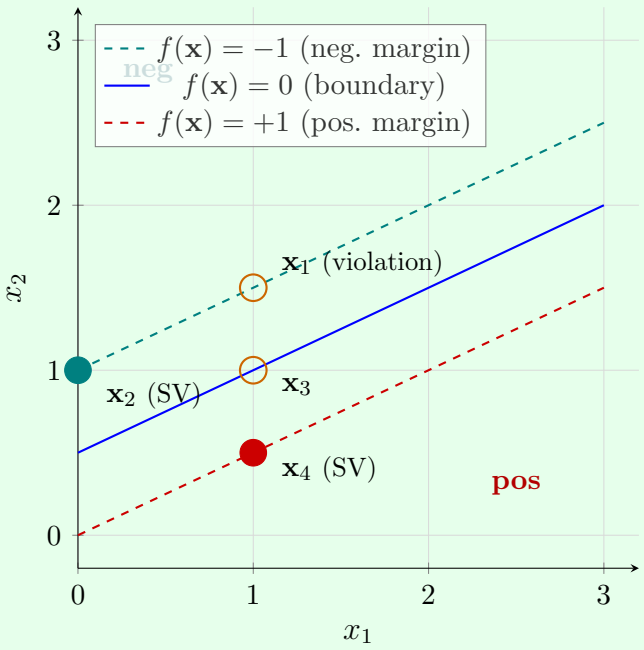
Answer

The margin edges are defined by $f(\mathbf{x}) = +1$ and $f(\mathbf{x}) = -1$.
Positive margin edge ($f(\mathbf{x}) = 1$):

$$x_1 - 2x_2 + 1 = 1 \implies x_2 = \tfrac{1}{2}x_1$$

Negative margin edge ($f(\mathbf{x}) = -1$):

$$x_1 - 2x_2 + 1 = -1 \implies x_2 = \tfrac{1}{2}x_1 + 1$$



The margin width is $\dfrac{2}{\|\mathbf{w}\|} = \dfrac{2}{\sqrt{1^2 + (-2)^2}} = \dfrac{2}{\sqrt{5}}$.

Question 11 (Puzzle)

Find a classifier with the *same* decision boundary but a margin *twice* as wide.

Answer

Key insight: The margin width equals $\frac{2}{\|\mathbf{w}\|}$. To double it, we need $\|\mathbf{w}\|$ to be *half as large* while keeping the direction of $\mathbf{w}$ unchanged (which keeps the same decision boundary orientation).

Solution: Halve every element of $\mathbf{w}$:

$$\mathbf{w}_{\text{new}} = \frac{1}{2}\mathbf{w} = \begin{pmatrix} 1/2 \\ -1 \end{pmatrix}$$

Now find b_{new}. The decision boundary must still cross $x_2 = \frac{1}{2}$ when $x_1 = 0$:

$$\tfrac{1}{2} \cdot 0 + (-1) \cdot \tfrac{1}{2} + b_{\text{new}} = 0 \implies b_{\text{new}} = \tfrac{1}{2}$$

Verification:

- New decision boundary: $\frac{1}{2}x_1 - x_2 + \frac{1}{2} = 0 \implies x_2 = \frac{1}{2}x_1 + \frac{1}{2}$ (same as before)
- New margin width: $\dfrac{2}{\|(1/2, -1)^\top\|} = \dfrac{2}{\sqrt{5}/2} = \dfrac{4}{\sqrt{5}} = \text{twice } \dfrac{2}{\sqrt{5}}$

Final answer: $\mathbf{w} = \begin{pmatrix} 1/2 \\ -1 \end{pmatrix}, \quad b = \dfrac{1}{2}.$

2. Backpropagation

Note

Background. Backpropagation is the algorithm used to compute gradients in a neural network efficiently. It applies the **chain rule** of calculus from the loss backward through each layer. We compute *local derivatives* at each module and multiply them together to get the full gradient.

The Network Architecture

The network is defined by:

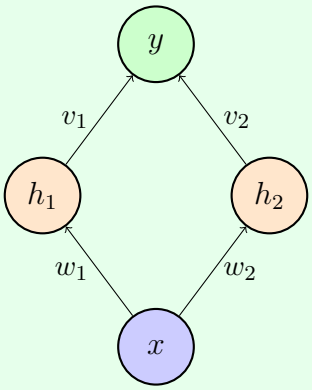
$$\begin{aligned} y &= v_1 h_1 + v_2 h_2 && \text{(output layer)} \\ h_1 &= \sigma(k_1) && \text{(hidden neuron 1)} \\ h_2 &= \sigma(k_2) && \text{(hidden neuron 2)} \\ k_1 &= w_1 x && \text{(pre-activation 1)} \\ k_2 &= w_2 x && \text{(pre-activation 2)} \end{aligned}$$

where σ is the logistic sigmoid: $\sigma(z) = \dfrac{1}{1+e^{-z}}$.

Question 12

Draw the network.

Answer



The network has one input x, two hidden neurons (h_1, h_2) with sigmoid activations, and one output y.

Question 13

Using the chain rule, derive $\dfrac{\partial \text{loss}}{\partial w_1}$.

Answer

The loss is $\text{loss} = \frac{1}{2}(y-t)^2$. We apply the chain rule step by step:

$$\frac{\partial \text{loss}}{\partial w_1} = \underbrace{\frac{\partial \text{loss}}{\partial y}}_{\text{loss}\to y} \cdot \underbrace{\frac{\partial y}{\partial h_1}}_{y\to h_1} \cdot \underbrace{\frac{\partial h_1}{\partial k_1}}_{h_1\to k_1} \cdot \underbrace{\frac{\partial k_1}{\partial w_1}}_{k_1\to w_1}$$

Computing each local derivative:

$$\begin{aligned}
\frac{\partial \text{loss}}{\partial y} &= (y-t) \\
\frac{\partial y}{\partial h_1} &= v_1 \quad \text{(coefficient of } h_1 \text{ in } y = v_1h_1 + v_2h_2\text{)} \\
\frac{\partial h_1}{\partial k_1} &= \sigma(k_1)\big(1-\sigma(k_1)\big) \quad \text{(derivative of sigmoid)} \\
\frac{\partial k_1}{\partial w_1} &= x \quad \text{(since } k_1 = w_1x\text{)}
\end{aligned}$$

Putting it together:

$$\boxed{\frac{\partial \text{loss}}{\partial w_1} = (y-t)\cdot v_1 \cdot \sigma(k_1)\big(1-\sigma(k_1)\big)\cdot x}$$

Why the sigmoid derivative? Recall $\frac{d\sigma}{dz} = \sigma(z)(1-\sigma(z))$. This is one of the most important derivatives in neural networks.

Question 14

The network is diamond-shaped, which usually suggests the multivariate chain rule is needed. Why is it *not* needed here? When would it be needed?

Answer

The **multivariate chain rule** is required when the *variable you are differentiating with respect to* appears along **multiple paths** from that variable to the loss.

In this case: The diamond occurs because y depends on x via *two* paths ($x \to k_1 \to h_1 \to y$ and $x \to k_2 \to h_2 \to y$). However, we are computing $\frac{\partial \text{loss}}{\partial w_1}$, *not* $\frac{\partial \text{loss}}{\partial x}$. Weight w_1 only appears on **one** path ($w_1 \to k_1 \to h_1 \to y$). The other path through h_2 does not involve w_1 at all, so its contribution is zero.

When would the multivariate chain rule be needed? For example:

- Computing $\frac{\partial \text{loss}}{\partial x}$, since x appears in *both* branches.
- Computing gradients when the loss is averaged over a *batch* of data points (as each data point contributes independently but sums together).
- In networks with *skip connections* or *shared weights* where one weight feeds into multiple paths that both lead to the output.

Question 15

With $x = 1$, $t = 1/2$, and all weights $= 1$, do the forward pass, then compute $\frac{\partial \text{loss}}{\partial w_1}$. Use $\sigma(1) \approx 3/4$.

Answer

Forward pass:

Variable	Formula	Value
k_1	$w_1 \cdot x = 1 \cdot 1$	1
k_2	$w_2 \cdot x = 1 \cdot 1$	1
h_1	$\sigma(k_1) = \sigma(1) \approx 3/4$	3/4
h_2	$\sigma(k_2) = \sigma(1) \approx 3/4$	3/4
y	$v_1 h_1 + v_2 h_2 = 1 \cdot \frac{3}{4} + 1 \cdot \frac{3}{4}$	3/2
loss	$\frac{1}{2}(y-t)^2 = \frac{1}{2}(\frac{3}{2} - \frac{1}{2})^2 = \frac{1}{2} \cdot 1$	1/2

Backward pass: Substituting into equation (2):

$$\frac{\partial \text{loss}}{\partial w_1} = (y-t) \cdot v_1 \cdot \sigma(k_1)(1-\sigma(k_1)) \cdot x$$

$$= \underbrace{\left(\tfrac{3}{2} - \tfrac{1}{2}\right)}_{=1} \cdot \underbrace{1}_{v_1} \cdot \underbrace{\tfrac{3}{4} \cdot \tfrac{1}{4}}_{\sigma(1-\sigma(1))} \cdot \underbrace{1}_{x} = 1 \cdot 1 \cdot \tfrac{3}{16} \cdot 1 = \boxed{\frac{3}{16}}$$

Note

Interpretation: The gradient $\frac{3}{16} > 0$ means that increasing w_1 slightly would increase the loss. In gradient descent we update $w_1 \leftarrow w_1 - \eta \cdot \frac{3}{16}$, i.e. we decrease w_1 to reduce the loss.

End of Week 4