Support Vector Machines

From Maximal Margins to Kernel Methods

The Big Picture

Support Vector Machines (SVMs) are powerful classifiers developed in the 1990s. The key idea is elegant: find the decision boundary that maximizes the 'gap' between classes. This chapter builds up SVMs in three stages:

Stage	Method	Handles
1	Maximal Margin Classifier	Perfectly separable data
2	Support Vector Classifier	Overlapping classes (soft margins)
3	Support Vector Machine	Non-linear boundaries (kernels)

1. Maximal Margin Classifier

1.1 What is a Hyperplane?

In p-dimensional space, a **hyperplane** is a flat surface of dimension p-1 that divides the space into two halves.

Examples by dimension:

* In 2D: A hyperplane is a **line** (1-dimensional)
* In 3D: A hyperplane is a **plane** (2-dimensional)
* In pD: A hyperplane has dimension p-1

Mathematical definition:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

If we define f(X) as:

$$f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

Then:

* f(X) = 0 means X lies exactly on the hyperplane
* f(X) > 0 means X lies on one side
* f(X) < 0 means X lies on the other side

1.2 The Separating Hyperplane

If we can find a hyperplane that perfectly separates two classes (all blue points on one side, all red on the other), we call it a **separating hyperplane**.

Classification rule: Simply check which side of the hyperplane a new point falls on:

$$\hat{y} = \text{sign}(f(X))$$

Example: Imagine classifying emails as spam/not-spam. If X1 = number of exclamation marks and X2 = number of 'FREE' occurrences, a separating line might be: 2*X1 + 3*X2 - 10 = 0. Emails above this line are spam; below are legitimate.

1.3 The Problem: Which Hyperplane?

If the data is separable, there are **infinitely many** separating hyperplanes. Which one should we choose?

Intuition: Choose the hyperplane that is 'most confident' - the one that stays as far as possible from both classes. This is the **Maximal Margin Hyperplane**.

1.4 The Margin and Support Vectors

Margin: The perpendicular distance from the hyperplane to the nearest training observation.

Maximal Margin Hyperplane: The separating hyperplane with the largest margin.

Support Vectors: The training observations that lie exactly on the margin boundary (equidistant from the hyperplane).

Critical insight: The maximal margin hyperplane depends ONLY on the support vectors. Moving any other observation (as long as it doesn't cross the margin) has NO effect on the classifier!

1.5 The Optimization Problem

We want to find the hyperplane that maximizes the margin M:

$$\max_{\beta_0, \beta_1, \dots, \beta_p} M \quad \text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1$$

Subject to: every observation is on the correct side with margin at least M:

$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq M \quad \forall i$$

where y_i is +1 or -1 indicating class membership.

1.6 Limitation

The fatal flaw: If the data is NOT perfectly linearly separable (which is almost always the case in practice), the maximal margin hyperplane simply does not exist. We need a more flexible approach.

2. Support Vector Classifier (Soft Margin)

2.1 The Problem with Hard Margins

Two issues with the maximal margin classifier:

* **Existence**: If classes overlap at all, no separating hyperplane exists
* **Sensitivity**: Even if separable, a single outlier can drastically shift the boundary (overfitting)

Example: Imagine 100 blue points and 100 red points that are well-separated, except one blue point that accidentally ended up deep in red territory. The maximal margin classifier would contort itself to accommodate this one point, likely misclassifying many red points.

2.2 The Solution: Allow Some Mistakes

The **Support Vector Classifier** (also called 'soft margin classifier') allows some observations to be:

* On the wrong side of the margin (margin violation)
* Even on the wrong side of the hyperplane (misclassification)

2.3 Slack Variables

We introduce **slack variables** epsilon_i for each observation:

epsilon_i Value	Interpretation
epsilon_i = 0	Observation is on correct side of margin
0 < epsilon_i < 1	Observation violates margin but is correctly classified
epsilon_i > 1	Observation is on wrong side of hyperplane (misclassified)

The modified constraint becomes:

$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq M(1 - \varepsilon_i)$$

2.4 The Tuning Parameter C

We constrain the total amount of slack:

$$\varepsilon_i \geq 0, \quad \sum_{i=1}^{n} \varepsilon_i \leq C$$

C is the 'budget' for violations:

C Value	Effect	Bias-Variance
Small C	Low tolerance for violations. Narrow margins. Few support vectors.	Low bias, High variance
Large C	High tolerance. Wide margins. Many support vectors.	High bias, Low variance

Analogy: Think of C as how 'forgiving' the teacher is. Small C = strict teacher who penalizes every mistake harshly. Large C = lenient teacher who allows many small errors in exchange for a simpler, more generalizable rule.

2.5 Support Vectors in Soft Margin

In the soft margin case, **support vectors** are observations that either:

* Lie exactly on the margin (epsilon_i = 0)
* Violate the margin (epsilon_i > 0)

Observations strictly inside the correct margin (with room to spare) do NOT affect the classifier. This makes SVMs robust to observations far from the boundary.

3. Support Vector Machines (Non-Linear)

3.1 The Limitation of Linear Boundaries

Sometimes no linear boundary works well, even with soft margins. The true decision boundary might be curved, circular, or have complex shape.

Example: Classifying points inside vs. outside a circle. No straight line can separate them, but a circular boundary works perfectly.

3.2 The Feature Expansion Approach

One solution: Transform the features to a higher-dimensional space where a linear separator exists.

Example: For circular boundary:

* Original features: X1, X2
* Expanded features: X1, X2, X1^2, X2^2, X1*X2
* A linear boundary in 5D maps to a non-linear (quadratic) boundary in 2D

Problem: The expanded feature space can be enormous. With polynomials of degree d in p dimensions, the number of features explodes combinatorially.

3.3 The Kernel Trick

Here's the beautiful insight: The SVM solution only depends on **inner products** between observations, not the observations themselves!

The standard inner product:

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

The **kernel trick**: Replace the inner product with a **kernel function** K(x_i, x_i') that computes the inner product in some (possibly infinite-dimensional) feature space, **without ever computing the transformation explicitly**.

The classifier becomes:

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i)$$

where S is the set of support vector indices and alpha_i are learned coefficients.

3.4 Common Kernels

1. Linear Kernel (standard SVC):

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

Just the regular inner product. Gives a linear decision boundary.

2. Polynomial Kernel (degree d):

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$

Effectively fits the classifier in a space of all polynomials up to degree d. Higher d = more flexible boundary.

3. Radial Basis Function (RBF) Kernel:

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right)$$

Also called Gaussian kernel. Creates highly flexible, localized decision boundaries.

Kernel	Flexibility	Key Parameter	Use When
Linear	Low	None	Classes are linearly separable
Polynomial	Medium	d (degree)	Polynomial relationship expected
RBF	High	gamma	Complex, unknown boundary shape

3.5 Understanding the RBF Kernel

The RBF kernel deserves special attention because it's so widely used:

How it works:

* K(x_i, x_i') measures **similarity** between observations
* If x_i and x_i' are close: K is near 1 (high similarity)
* If x_i and x_i' are far: K is near 0 (low similarity)
* Only **nearby support vectors** influence a test point's classification

The gamma parameter:

* **Small gamma**: Large 'reach' - points far away still have influence. Smoother boundary.
* **Large gamma**: Small 'reach' - only very close points matter. Wiggly boundary (risk of overfitting).

Intuition: gamma controls how 'local' the classifier is. Large gamma = each support vector creates a small 'bubble' of influence. Small gamma = each support vector affects a large region.

4. SVMs for Multiple Classes

SVMs are inherently **binary** classifiers. For K > 2 classes, we combine multiple binary SVMs:

4.1 One-Versus-One (OvO)

1. Build K(K-1)/2 binary classifiers (all possible pairs)
2. Each classifier votes for one of its two classes
3. Final prediction: the class with the most votes (majority vote)

Example: For 4 classes (A, B, C, D), we build 6 classifiers: A-vs-B, A-vs-C, A-vs-D, B-vs-C, B-vs-D, C-vs-D. A test point might get votes: A=3, B=2, C=1, D=0. Prediction: Class A.

4.2 One-Versus-All (OvA)

1. Build K binary classifiers (each class vs. all others)
2. Each classifier outputs a confidence score (distance from hyperplane)
3. Final prediction: the class whose classifier gives highest confidence

Method	Number of Classifiers	Training Size per Classifier
One-vs-One	K(K-1)/2	Smaller (only 2 classes)
One-vs-All	K	Larger (all data, imbalanced)

5. Relationship to Logistic Regression

SVMs and logistic regression are both 'Loss + Penalty' methods, but with different loss functions.

5.1 The Hinge Loss (SVM)

$$L(y, f(x)) = \max[0, 1 - y \cdot f(x)]$$

Key property:

* Loss = 0 when observation is on correct side of margin (y*f(x) >= 1)
* Loss increases linearly when observation violates margin
* Observations far on the correct side contribute **nothing** to the loss

This is why SVMs depend only on support vectors - all other observations have zero loss and zero gradient!

5.2 The Logistic Loss

$$L(y, f(x)) = \log(1 + e^{-y \cdot f(x)})$$

Key property:

* Loss is **never exactly zero** (always positive)
* All observations contribute to the loss (though distant ones contribute little)
* Gives probabilities, not just classifications

5.3 When to Use Which?

Criterion	SVM	Logistic Regression
Class separation	Better when well-separated	Better when overlapping
Probabilities needed?	No (though can approximate)	Yes (natural output)
Sparse solution	Yes (depends on support vectors)	No (all observations matter)
Outlier sensitivity	Robust (ignores distant points)	More affected
Kernels	Natural extension	Less common

6. Key Takeaways

* **Maximal Margin**: Find the hyperplane with largest gap between classes. Only works for perfectly separable data.
* **Support Vectors**: The critical observations that define the boundary. All other points can be ignored!
* **Soft Margin (C)**: Allow some violations. C controls the trade-off: small C = strict, narrow margins; large C = lenient, wide margins.
* **Kernel Trick**: Get non-linear boundaries by computing inner products in high-dimensional space without explicit transformation.
* **RBF Kernel**: Most flexible, works locally. gamma controls the 'reach' of each support vector.
* **Hinge Loss**: Zero loss for correctly classified points beyond the margin. This creates sparsity (dependence on support vectors only).
* SVMs work best when classes are well-separated; logistic regression may be better for overlapping classes.
* Always tune C (and gamma for RBF) using cross-validation.

Quick Reference: Key Formulas

Hyperplane equation:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

Polynomial kernel:

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$

RBF kernel:

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right)$$

Hinge loss:

$$L(y, f(x)) = \max[0, 1 - y \cdot f(x)]$$

7. Practical Tips (sklearn)

* Use **SVC()** from sklearn.svm
* **kernel='linear'** for linear SVC, **'poly'** for polynomial, **'rbf'** for Gaussian
* **C**: Regularization parameter. Start with C=1.0 and tune via CV
* **gamma**: For RBF kernel. 'scale' (default) or 'auto' are good starting points
* **degree**: For polynomial kernel. Usually 2 or 3
* Use **decision_function()** to get distances from hyperplane (useful for ROC curves)

* **Scale your features!** SVMs are sensitive to feature scales