

Support vector machines

Fraida Fund

Contents

Maximal margin classifier	2
Binary classification problem	2
Linear separability	2
Separating hyperplane (1)	2
Separating hyperplane (2)	2
Using the hyperplane to classify	2
Which separating hyperplane is best?	3
Margin	3
Maximal margin classifier	3
Support vectors	4
Constructing the maximal margin classifier	4
Constructing the maximal margin classifier (1)	5
Constructing the maximal margin classifier (2)	6
Problems with MM classifier (1)	6
Problems with MM classifier (2)	6
Support vector classifier	7
Basic idea	7
Constructing the support vector classifier	7
Support vector	8
Illustration of effect of K	8
K controls bias-variance tradeoff	8
Loss function	8
Compared to logistic regression	8
Solution	9
Problem formulation - original	9
Problem formulation - equivalent	9
Problem formulation - equivalent (2)	9
Background: constrained optimization	10
Background: Illustration	10
Background: Solving with Lagrangian (1)	10
Background: Solving with Lagrangian (2)	10
Background: Active/inactive constraint	11
Background: Primal and dual formulation	11
Problem formulation - Lagrangian primal	12
Problem formulation - Lagrangian dual	12
Problem formulation - Lagrangian dual (2)	12
Solution (1)	13
Solution (2)	13
Solution (3)	13
Why solve dual problem?	13
Correlation interpretation (1)	13

Correlation interpretation (2)	13
Relationship between SVM and other models	14

Math prerequisites for this lecture: Constrained optimization (Appendix C in in Boyd and Vandenberghe).

Maximal margin classifier

Binary classification problem

- n training samples, each with p features $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^p$
- Class labels $y_1, \dots, y_n \in \{-1, 1\}$

Linear separability

The problem is **perfectly linearly separable** if there exists a **separating hyperplane** H_i such that

- all $\mathbf{x} \in C_i$ lie on its positive side, and
- all $\mathbf{x} \in C_j, j \neq i$ lie on its negative side.

Separating hyperplane (1)

The separating hyperplane has the property that for all $i = 1, \dots, n$,

$$w_0 + \sum_{j=1}^p w_j x_{ij} > 0 \text{ if } y_i = 1$$

$$w_0 + \sum_{j=1}^p w_j x_{ij} < 0 \text{ if } y_i = -1$$

Separating hyperplane (2)

Equivalently:

$$y_i \left(w_0 + \sum_{j=1}^p w_j x_{ij} \right) > 0 \quad (1)$$

Using the hyperplane to classify

Then, we can classify a new sample \mathbf{x} using the sign of

$$z = w_0 + \sum_{j=1}^p w_j x_{ij}$$

and we can use the magnitude of z to determine how confident we are about our classification. (Larger z = farther from hyperplane = more confident about classification.)

Which separating hyperplane is best?

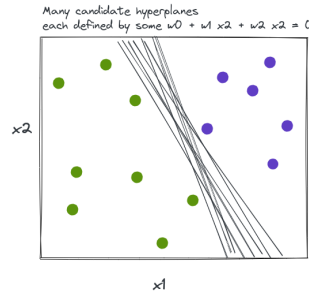


Figure 1: If the data is linearly separable, there are many separating hyperplanes.

Previously, with the logistic regression classifier, we found the maximum likelihood classifier: the hyperplane that maximizes the probability of these particular observations.

Margin

For any “candidate” hyperplane,

- Compute perpendicular distance from each sample to separating hyperplane.
- Smallest distance among all samples is called the **margin**.

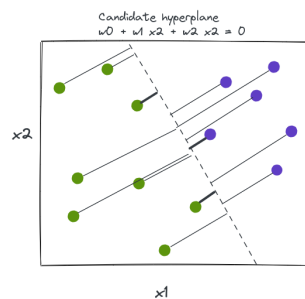


Figure 2: For this hyperplane, bold lines show the smallest distance (tie among several samples).

Maximal margin classifier

- Choose the line that maximizes the margin!
- Find the widest “slab” we can fit between the two classes.
- Choose the midline of this “slab” as the decision boundary.

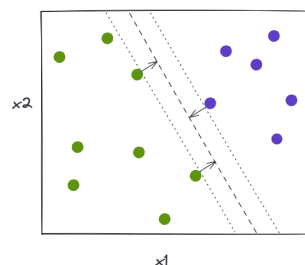


Figure 3: Maximal margin classifier. Width of the “slab” is 2x the margin.

Support vectors

- Points that lie on the border of maximal margin hyperplane are **support vectors**
- They “support” the maximal margin hyperplane: if these points move, then the maximal margin hyperplane moves
- Maximal margin hyperplane is not affected by movement of any other point, as long as it doesn't cross borders!

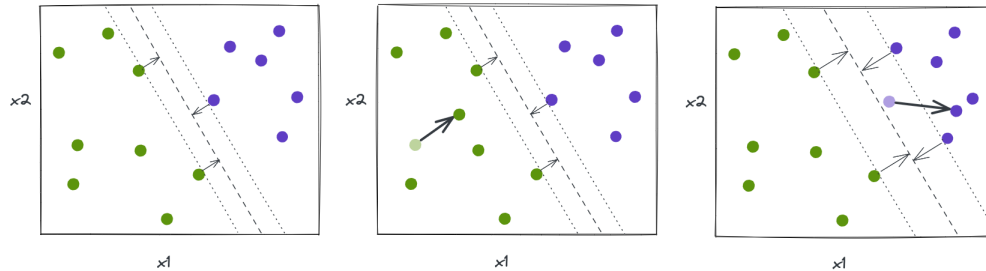


Figure 4: Maximal margin classifier (left) is not affected by movement of a point that is not a support vector (middle) but the hyperplane and/or margin are affected by movement of a support vector (right).

Constructing the maximal margin classifier

To construct this classifier, we will set up a *constrained optimization* problem with:

- an objective
- one or more constraints to satisfy

What should the objective/constraints be in this scenario?

Constructing the maximal margin classifier (1)

$$\underset{\mathbf{w}, \gamma}{\text{maximize}} \gamma \quad (2)$$

$$\text{subject to: } \sum_{j=1}^p w_j^2 = 1 \quad (3)$$

$$\text{and } y_i \left(w_0 + \sum_{j=1}^p w_j x_{ij} \right) \geq \gamma, \forall i \quad (4)$$

The constraint

$$y_i \left(w_0 + \sum_{j=1}^p w_j x_{ij} \right) \geq \gamma, \forall i$$

guarantees that each observation is on the correct side of the hyperplane *and* on the correct side of the margin, if margin γ is positive. (This is analogous to Equation 1, but we have added a margin.)

The constraint

$$\text{and } \sum_{j=1}^p w_j^2 = 1$$

is not really a constraint: if a separating hyperplane is defined by $w_0 + \sum_{j=1}^p w_j x_{ij} = 0$, then for any $k \neq 0$, $k \left(w_0 + \sum_{j=1}^p w_j x_{ij} \right) = 0$ is also a separating hyperplane.

This “constraint” just scales weights so that distance from i th sample to the hyperplane is given by $y_i \left(w_0 + \sum_{j=1}^p w_j x_{ij} \right)$. This is what make the previous constraint meaningful!

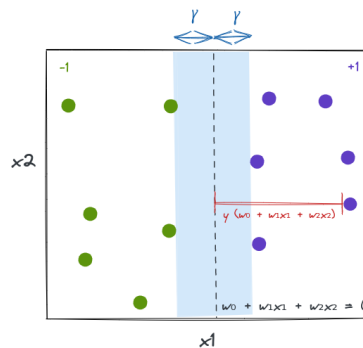


Figure 5: Maximal margin classifier.

Constructing the maximal margin classifier (2)

The constraints ensure that

- Each observation is on the correct side of the hyperplane, and
- at least γ away from the hyperplane

and γ is maximized.

Problems with MM classifier (1)

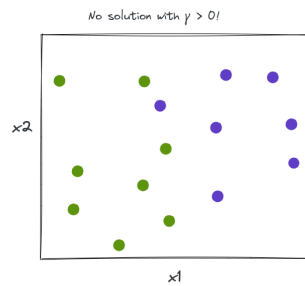


Figure 6: When data is not linearly separable, optimization problem has no solution with $\gamma > 0$.

Problems with MM classifier (2)

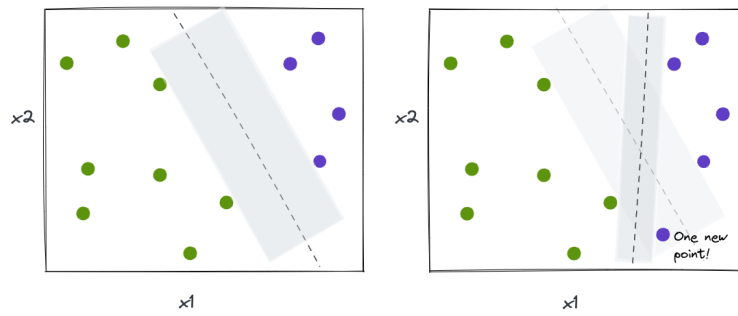


Figure 7: The classifier is not robust - one new observation can dramatically shift the hyperplane.

Support vector classifier

Basic idea

- Generalization of MM classifier to non-separable case
- Use a hyperplane that *almost* separates the data
- “Soft margin”

Constructing the support vector classifier

$$\underset{\mathbf{w}, \epsilon, \gamma}{\text{maximize}} \gamma \quad (5)$$

$$\text{subject to: } \sum_{j=1}^p w_j^2 = 1 \quad (6)$$

$$y_i \left(w_0 + \sum_{j=1}^p w_j x_{ij} \right) \geq \gamma(1 - \epsilon_i), \forall i \quad (7)$$

$$\epsilon_i \geq 0 \forall i, \quad \sum_{i=1}^n \epsilon_i \leq K \quad (8)$$

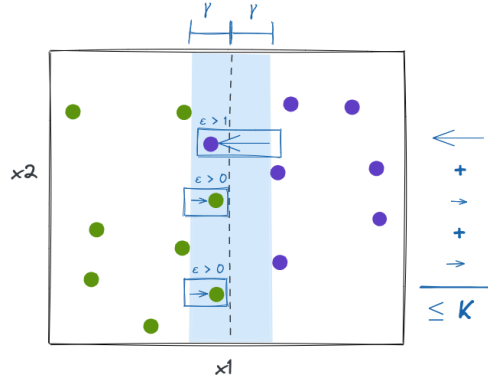


Figure 8: Support vector classifier. Note: the blue arrows show $y_i \gamma \epsilon_i$.

K is a non-negative tuning parameter.

Slack variable ϵ_i determines where a point lies:

- If $\epsilon_i = 0$, point is on the correct side of margin
- If $\epsilon_i > 0$, point has *violated* the margin (wrong side of margin)
- If $\epsilon_i > 1$, point is on wrong side of hyperplane and is misclassified

K is the **budget** that determines the number and severity of margin violations we will tolerate.

- $K = 0 \rightarrow$ same as MM classifier
- $K > 0$, no more than K observations may be on wrong side of hyperplane
- As K increases, margin widens; as K decreases, margin narrows.

Support vector

For a support vector classifier, the only points that affect the classifier are:

- Points that lie on the margin boundary
- Points that violate margin

These are the *support vectors*.

Illustration of effect of K

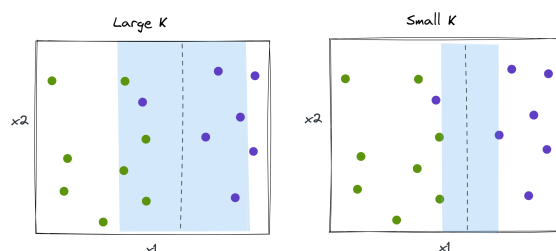


Figure 9: The margin shrinks as K decreases.

K controls bias-variance tradeoff

- When K is large: many support vectors, variance is low, but bias may be high.
- When K is small: few support vectors, high variance, but low bias.

Terminology note: In ISLR and in the first part of these notes, meaning of constant is opposite its meaning in Python `sklearn`:

- ISLR and these notes: Large K , wide margin.
- Python `sklearn`: Large C , small margin.

Loss function

This problem is equivalent to minimizing hinge loss:

$$\underset{\mathbf{w}}{\text{minimize}} \left(\sum_{i=1}^n \max[0, 1 - y_i(w_0 + \sum_{j=1}^p w_j x_{ij})] + \lambda \sum_{j=1}^p w_j^2 \right)$$

where λ is non-negative tuning parameter.

Zero loss for observations where

$$y_i \left(w_0 + \sum_{j=1}^p w_j x_{ij} \right) \geq 1$$

and width of margin depends on $\sum w_j^2$.

Compared to logistic regression

- **Hinge loss:** zero for points on correct side of margin.
- **Logistic regression loss:** small for points that are far from decision boundary.

Solution

Problem formulation - original

$$\begin{aligned} & \underset{\mathbf{w}, \epsilon, \gamma}{\text{maximize}} && \gamma \\ & \text{subject to} && \sum_{j=1}^p w_j^2 = 1 \\ & && y_i \left(w_0 + \sum_{j=1}^p w_j x_{ij} \right) \geq \gamma(1 - \epsilon_i), \forall i \\ & && \epsilon_i \geq 0, \quad \forall i \\ & && \sum_{i=1}^n \epsilon_i \leq K \end{aligned}$$

Problem formulation - equivalent

Remember that any scaled version of the hyperplane is the same line. So let's make $\|w\|$ inversely proportional to γ . Then we can formulate the equivalent problem:

$$\begin{aligned} & \underset{\mathbf{w}, \epsilon}{\text{minimize}} && \sum_{j=1}^p w_j^2 \\ & \text{subject to} && y_i \left(w_0 + \sum_{j=1}^p w_j x_{ij} \right) \geq 1 - \epsilon_i, \forall i \\ & && \epsilon_i \geq 0, \quad \forall i \\ & && \sum_{i=1}^n \epsilon_i \leq K \end{aligned}$$

Problem formulation - equivalent (2)

Or, move the "budget" into the objective function:

$$\begin{aligned} & \underset{\mathbf{w}, \epsilon}{\text{minimize}} && \frac{1}{2} \sum_{j=1}^p w_j^2 + C \sum_{i=1}^n \epsilon_i \\ & \text{subject to} && y_i \left(w_0 + \sum_{j=1}^p w_j x_{ij} \right) \geq 1 - \epsilon_i, \quad \forall i \\ & && \epsilon_i \geq 0, \quad \forall i \end{aligned}$$

Background: constrained optimization

Basic formulation of constrained optimization problem:

- **Objective:** Minimize $f(x)$
- **Constraint(s):** subject to $g(x) \leq 0$

Find x^* that satisfies $g(x^*) \leq 0$ and, for any other x that satisfies $g(x) \leq 0$, $f(x) \geq f(x^*)$.

Background: Illustration

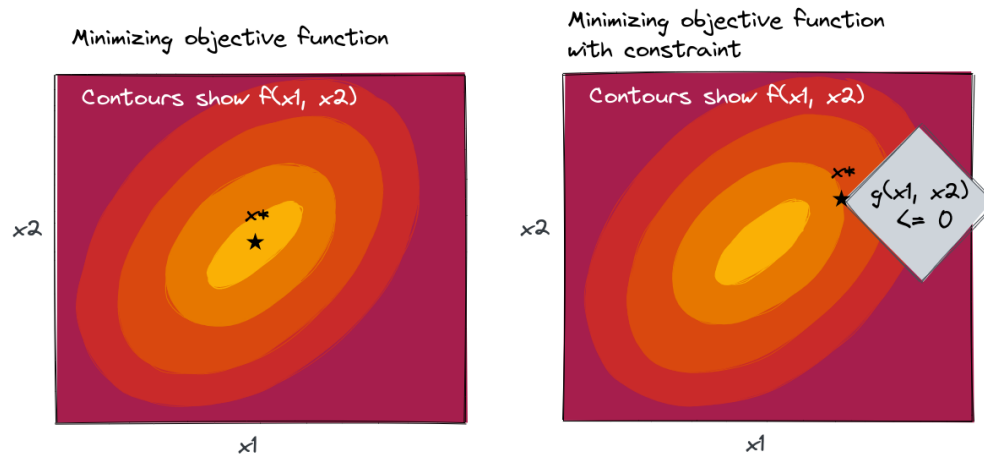


Figure 10: Minimizing objective function, without (left) and with (right) a constraint.

Background: Solving with Lagrangian (1)

To solve, we form the Lagrangian:

$$L(x, \lambda) = f(x) + \lambda_1 g_1(x) + \dots + \lambda_m g_m(x)$$

where each $\lambda \geq 0$ is a *Lagrange multiplier*.

The $\lambda g(x)$ terms “pull” solution toward feasible set, away from non-feasible set.

Background: Solving with Lagrangian (2)

Then, to solve, we use joint optimization over x and λ :

$$\underset{x}{\text{minimize}} \quad \underset{\lambda \geq 0}{\text{maximize}} \quad f(x) + \lambda g(x)$$

over x and λ . If for some x ,

- $g(x) \leq 0$: the constraint is not active, $\lambda = 0$.
- $g(x) > 0$: the constraint is active, $\lambda > 0$. Then we need to change x to make $g(x)$ smaller.

“Solve” in the usual way if the function is convex: by taking partial derivative of $L(x, \lambda)$ with respect to each argument, and setting it to zero. The solution to the original function will be a saddle point in the Lagrangian.

This “pull” between the x that minimizes $f(x)$ and the $\lambda g(x)$ ends up making the constraint “tight”: if $\lambda > 0$ then we’ll make $g(x) = 0$.

This is called the KKT complementary slackness condition: for every constraint, $\lambda g(x) = 0$, either because $\lambda = 0$ (inactive constraint) or $g(x) = 0$ (active constraint).

Background: Active/inactive constraint

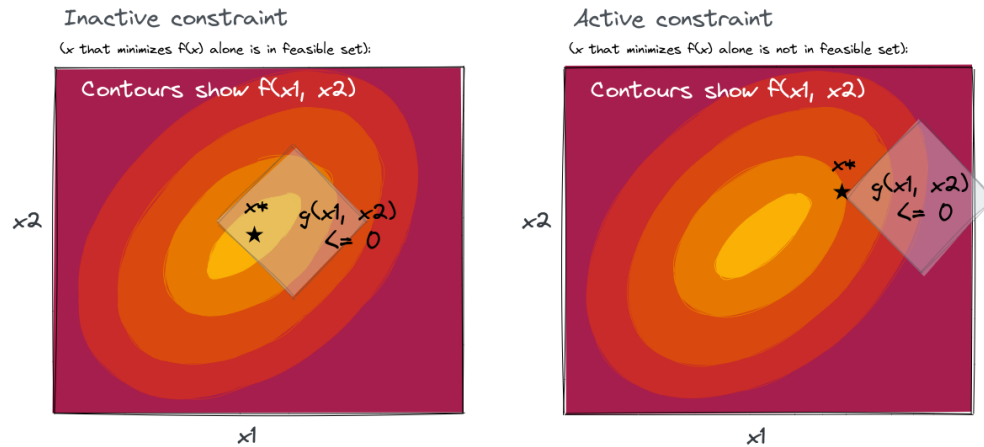


Figure 11: Optimization with inactive, active constraint.

Background: Primal and dual formulation

Under the right conditions, the solution to the *primal* problem:

$$\underset{x}{\text{minimize}} \quad \underset{\lambda \geq 0}{\text{maximize}} \quad L(x, \lambda)$$

is the same as the solution to the *dual* problem:

$$\underset{\lambda \geq 0}{\text{maximize}} \quad \underset{x}{\text{minimize}} \quad L(x, \lambda)$$

Problem formulation - Lagrangian primal

Back to our SVC problem - let's form the Lagrangian and optimize:

$$\begin{aligned} \underset{\mathbf{w}, \epsilon}{\text{minimize}} \quad & \underset{\alpha_i \geq 0, \mu_i \geq 0, \forall i}{\text{maximize}} \quad \frac{1}{2} \sum_{j=1}^p w_j^2 \\ & + C \sum_{i=1}^n \epsilon_i \\ & - \sum_{i=1}^n \alpha_i \left[y_i (w_0 + \sum_{j=1}^p w_j x_{ij}) - (1 - \epsilon_i) \right] \\ & - \sum_{i=1}^n \mu_i \epsilon_i \end{aligned}$$

This is the *primal* problem.

Problem formulation - Lagrangian dual

The equivalent *dual* problem:

$$\begin{aligned} \underset{\alpha_i \geq 0, \mu_i \geq 0, \forall i}{\text{maximize}} \quad & \underset{\mathbf{w}, \epsilon}{\text{minimize}} \quad \frac{1}{2} \sum_{j=1}^p w_j^2 \\ & + C \sum_{i=1}^n \epsilon_i \\ & - \sum_{i=1}^n \alpha_i \left[y_i (w_0 + \sum_{j=1}^p w_j x_{ij}) - (1 - \epsilon_i) \right] \\ & - \sum_{i=1}^n \mu_i \epsilon_i \end{aligned}$$

We solve this by taking the derivatives with respect to \mathbf{w} , ϵ and setting them to zero. Then, we plug those values back into the dual equation...

Problem formulation - Lagrangian dual (2)

$$\begin{aligned} \underset{\alpha_i \geq 0, \forall i}{\text{maximize}} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & \sum_{i=1}^n \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad \forall i \end{aligned}$$

This turns out to be not too terrible to solve.

Solution (1)

Optimal coefficients for $j = 1, \dots, p$ are:

$$\mathbf{w}^* = \sum_{i=1}^n \alpha_i^* y_i \mathbf{x}_i$$

where α_i^* come from the solution to the dual problem.

Solution (2)

- $\alpha_i^* > 0$ only when x_i is a support vector (active constraint).
- Otherwise, $\alpha_i^* = 0$ (inactive constraint).

Solution (3)

That leaves w_0^* - we can solve

$$w_0^* = y_i - \sum_{j=1}^p w_j x_{ij}$$

using any sample i where $\alpha_i^* > 0$, i.e. any support vector.

Why solve dual problem?

For high-dimension problems (many features), dual problem can be much faster to solve than primal problem:

- Primal problem: optimize over $p + 1$ coefficients.
- Dual problem: optimize over n dual variables, but there are only as many non-zero ones as there are support vectors.

Also: the kernel trick, which we'll discuss next...

Correlation interpretation (1)

Given a new sample \mathbf{x} to classify, compute

$$\hat{z}(\mathbf{x}) = w_0 + \sum_{j=1}^p w_j x_j = w_0 + \sum_{i=1}^n \alpha_i y_i \sum_{j=1}^p x_{ij} x_j$$

Measures inner product (a kind of "correlation") between new sample and each support vector.

Correlation interpretation (2)

Classifier output (assuming -1,1 labels):

$$\hat{y}(\mathbf{x}) = \text{sign}(\hat{z}(\mathbf{x}))$$

Predicted label is weighted average of labels for support vectors, with weights proportional to "correlation" of test sample and support vector.

Relationship between SVM and other models

- Like a logistic regression - linear classifier, separating hyperplane is $w_0 + \sum_{j=1}^p w_j x_{ij} = 0$
- Like a weighted KNN - predicted label is weighted average of labels for support vectors, with weights proportional to “similarity” of test sample and support vector.