The support vector machine

Nuno Vasconcelos

*ECE Department, UCSD*
Outline

- we have talked about classification and linear discriminants
- then we did a detour to talk about kernels
  - how do we implement a non-linear boundary on the linear discriminant framework
- this led us to
  - RKHS, the reproducing theorem, regularization
  - the idea that all these learning problems boil down to the solution of an optimization problem
- we have seen how to solve optimization problems
  - with and without constraints, equalities, inequalities
- today we will put everything together by studying the SVM
Classification

⚠️ a classification problem has two types of variables

- \( X \) - vector of observations (features) in the world
- \( Y \) - state (class) of the world

⚠️ e.g.

- \( x \in \mathcal{X} \subseteq \mathbb{R}^2 = \) (fever, blood pressure)
- \( y \in \mathcal{Y} = \) \{disease, no disease\}

⚠️ \( X, Y \) related by a (unknown) function

\[ y = f(x) \]

⚠️ goal: design a classifier \( h: \mathcal{X} \rightarrow \mathcal{Y} \) such that \( h(x) = f(x) \ \forall x \)
Linear classifier

- implements the decision rule

\[ h^*(x) = \begin{cases} 
1 & \text{if } g(x) > 0 \\
-1 & \text{if } g(x) < 0 
\end{cases} = \text{sgn}[g(x)] \quad \text{with} \quad g(x) = w^T x + b \]

- has the properties
  
  • it divides \( \mathcal{X} \) into two “half-spaces”
  
  • boundary is the plane with:
    
    • normal \( w \)
    
    • distance to the origin \( b/\|w\| \)
  
  • \( g(x)/\|w\| \) is the distance from point \( x \) to the boundary
    
    • \( g(x) = 0 \) for points on the plane
    
    • \( g(x) > 0 \) on the side \( w \) points to (“positive side”)
    
    • \( g(x) < 0 \) on the “negative side”
Linear classifier

we have a classification error if

- $y = 1$ and $g(x) < 0$  or  $y = -1$ and $g(x) > 0$
- i.e $y.g(x) < 0$

and a correct classification if

- $y = 1$ and $g(x) > 0$  or  $y = -1$ and $g(x) < 0$
- i.e $y.g(x) > 0$

note that, for a linearly separable training set

$D = \{(x_1,y_1), \ldots, (x_n,y_n)\}$

we can have zero empirical risk

the necessary and sufficient condition is that

$y_i(w^T x_i + b) > 0$,  $\forall i$
The margin

- is the distance from the boundary to the closest point

\[ \gamma = \min_i \frac{\|w^T x_i + b\|}{\|w\|} \]

- there will be no error if it is strictly greater than zero

\[ y_i (w^T x_i + b) > 0, \quad \forall i \iff \gamma > 0 \]

- note that this is ill-defined in the sense that \( \gamma \) does not change if both \( w \) and \( b \) are scaled by \( \lambda \)

- we need a normalization
Maximizing the margin

- this is similar to what we have seen for Fisher discriminants
- let's assume we have selected some normalization, e.g. $||w||=1$
- the next question is: what is the cost that we are going to optimize?
- there are several planes that separate the classes, which one is best?
- recall that in the case of the Perceptron, we have seen that the margin determines the complexity of the learning problem
  - the Perceptron converges in less than $(k/\gamma)^2$ iterations
- it sounds like maximizing the margin is a good idea.
Maximizing the margin

intuition 1:

- think of each point in the training set as a sample from a probability density centered on it
- if we draw another sample, we will not get the same points
- each point is really a pdf with a certain variance
- this is a kernel density estimate
- if we leave a margin of $\gamma$ on the training set, we are safe against this uncertainty
- (as long as the radius of support of the pdf is smaller than $\gamma$)
- the larger $\gamma$, the more robust the classifier!
Maximizing the margin

intuition 2:

- think of the plane as an uncertain estimate because it is learned from a sample drawn at random
- since the sample changes from draw to draw, the plane parameters are random variables of non-zero variance
- instead of a single plane we have a probability distribution over planes
- the larger the margin, the larger the number of planes that will not originate errors
- the larger $\gamma$, the larger the variance allowed on the plane parameter estimates!
Normalization

we will go over a formal proof in a few classes
for now let’s look at the normalization
natural choice is $||w|| = 1$
the margin is maximized by solving

$$\max_{w,b} \min_i |w^T x_i + b|$$
subject to $||w||^2 = 1$

this is somewhat complex

- need to find the points that achieve the min without knowing what $w$ and $b$ are
- optimization problem with quadratic constraints
Normalization

- a more convenient normalization is to make $|g(x)| = 1$ for the closest point, i.e.

$$\min_i |\mathbf{w}^T \mathbf{x}_i + b| \equiv 1$$

under which

$$\gamma = \frac{1}{\|\mathbf{w}\|}$$

- the SVM is the classifier that maximizes the margin under these constraints

$$\min_{\mathbf{w}, b} \|\mathbf{w}\|^2 \text{ subject to } y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \ \forall i$$
Support vector machine

\[
\min_{w, b} \|w\|^2 \quad \text{subject to} \quad y_i (w^T x_i + b) \geq 1 \quad \forall i
\]

last time we proved the following theorem

**Theorem: (strong duality)** Consider the problem

\[
x^* = \arg \min_{x \in X} f(x) \quad \text{subject to} \quad e_j^T x - d_j \leq 0
\]

where \( X, \) and \( f \) are convex, and the optimal value \( f^* \) finite. Then there is at least one Lagrange multiplier vector and there is no duality gap

this means the SVM problem can be solved by solving its dual
The dual problem

For the primal
\[
\min_{w, b} \frac{1}{2} \|w\|^2 \quad \text{subject to } y_i(w^T x_i + b) \geq 1 \quad \forall i
\]

the dual problem is

\[
\max_{\alpha \geq 0} q(\alpha) = \max_{\alpha \geq 0} \left\{ \min_w L(w, b, \alpha) \right\}
\]

where the Lagrangian is

\[
L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i} \alpha_i \left[ y_i (w^T x_i + b) - 1 \right]
\]

Setting derivatives to zero

\[
\nabla_w L = 0 \iff L(w, b, \alpha) = w - \sum_{i} \alpha_i y_i x_i = 0 \iff w^* = \sum_{i} \alpha_i y_i x_i
\]

\[
\nabla_b L = 0 \iff \sum_{i} y_i \alpha_i = 0
\]
The dual problem

Plugging back \( w^* = \sum_i \alpha_i y_i x_i \), \( \sum_i y_i \alpha_i = 0 \) we get the Lagrangian

\[
L(w^*, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_i \alpha_i [y_i (w^T x_i + b) - 1]
\]

\[
= \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
- \sum_i \alpha_i y_i b + \sum_i \alpha_i = 0
\]

\[
= -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i
\]
The dual problem

and the dual problem is

$$\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \right\}$$

subject to $$\sum_i y_i \alpha_i = 0$$

once this is solved, the vector

$$w^* = \sum_i \alpha_i y_i x_i$$

is the normal to the maximum margin plane

note:

- the dual solution does not determine the optimal $$b^*$$, since $$b$$ drops off when we take derivatives
The dual problem

- determining \( b^* \)
  - various possibilities, for example
  - pick one point \( x^+ \) on the margin on the \( y=1 \) side and one point \( x^- \) on the \( y=-1 \) side
  - use the margin constraint

\[
\begin{align*}
  w^T x^+ + b &= 1 \\
  w^T x^- + b &= -1 
\end{align*} \quad \Leftrightarrow \quad b^* = -\frac{w^T (x^+ + x^-)}{2}
\]

- note:
  - the maximum margin solution guarantees that there is always at least one point “on the margin” on each side
  - if not, we could move the plane and get a larger margin
Support vectors

- another possibility is to average over all points on the margin
- these are called support vectors
- from the KKT conditions, an inactive constraint has zero Lagrange multiplier $\alpha_i$. That is,
  - i) $\alpha_i > 0$ and $y_i(w^T x_i + b^*) = 1$
  - or
  - ii) $\alpha_i = 0$ and $y_i(w^T x_i + b^*) \geq 1$
- hence $\alpha_i > 0$ only for points $|w^T x_i + b^*| = 1$
- which are those that lie at a distance equal to the margin
Support vectors

- points with $\alpha_i > 0$ support the optimal plane $(w^*, b^*)$.
- for this they are called “support vectors”
- note that the decision rule is

$$f(x) = \text{sgn}(w^* \cdot x + b^*)$$

$$= \text{sgn}(\sum_i y_i \alpha_i^* x_i^T x + b^*)$$

$$= \text{sgn}\left[ \sum_{i \in SV} y_i \alpha_i^* x_i^T x + b^* \right]$$

where $SV = \{i \mid \alpha_i^* > 0\}$ is the set of support vectors
Support vectors

since the decision rule is

\[ f(x) = \text{sgn}\left[ \sum_{i \in SV} y_i \alpha_i^* x_i^T x + b^* \right] \]

we only need the support vectors to completely define the classifier

we can literally throw away all other points!

the Lagrange multipliers can also be seen as a measure of importance of each point

points with \( \alpha_i = 0 \) have no influence, small perturbation does not change solution
Perceptron learning

note the similarities with the dual Perceptron

set $\alpha_i = 0, b = 0$
set $R = \max_i ||x_i||$
do {
  for $i = 1:n$
  
  if $y_i \left( \sum_{j=1}^{n} \alpha_j y_j x_j^T x_i + b \right) \leq 0$ then {
    $\alpha_i = \alpha_i + 1$
    $b = b + y_i R^2$
  }
}

until no errors

In this case:

- $\alpha_i = 0$ means that the point was never misclassified
- this means that we have an “easy” point, far from the boundary
- very unlikely to happen for a support vector
- but the Perceptron does not maximize the margin!
The robustness of SVMs

- in SLI we talked a lot about the “curse of dimensionality”
  - number of examples required to achieve certain precision is exponential in the number of dimensions
- it turns out that SVMs are remarkably robust to dimensionality
  - not uncommon to see successful applications on 1,000D+ spaces
- two main reasons for this:
  - 1) all that the SVM does is to learn a plane.

  Although the number of dimensions may be large, the number of parameters is relatively small and there is no much room for overfitting

  In fact, $d+1$ points are enough to specify the decision rule in $R^d$!
SVMs as feature selectors

The second reason is that the space is not really that large

- 2) the SVM is a feature selector

To see this let’s look at the decision function

\[ f(x) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* x_i^T x + b^* \right]. \]

This is a thresholding of the quantity

\[ \sum_{i \in SV} y_i \alpha_i^* x_i^T x \]

Note that each of the terms \( x_i^T x \) is the projection of the vector to classify \((x)\) into the training vector \( x_i \)
SVMs as feature selectors

- defining \( z \) as the vector of the projection onto all support vectors

\[
Z(X) = \left( x^T x_{i_1}, \ldots, x^T x_{i_k} \right)^T
\]

- the decision function is a plane in the \( z \)-space

\[
f(X) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* x_i^T x + b^* \right] = \text{sgn} \left[ \sum_k w_k^* z_k(x) + b^* \right]
\]

with

\[
w^* = \left( \alpha_{i_1}^* y_{i_1}, \ldots, \alpha_{i_k}^* y_{i_k} \right)^T
\]

- this means that
  - the classifier operates on the span of the support vectors!
  - the SVM performs feature selection automatically
SVMs as feature selectors

- geometrically, we have:
  - 1) projection on the span of the support vectors
  - 2) classifier on this space

\[ w^* = (\alpha_{i_1}y_{i_1}, \ldots, \alpha_{i_k}y_{i_k})^T \]

- the effective dimension is \(|SV|\) and, typically, \(|SV| \ll n\)
In summary

SVM training:

• 1) solve

\[
\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \right\}
\]

subject to \( \sum_i y_i \alpha_i = 0 \)

• 2) then compute

\[
W^* = \sum_{i \in SV} \alpha_i^* y_i x_i \quad \text{and} \quad b^* = -\frac{1}{2} \sum_{i \in SV} y_i \alpha_i^* (x_i^T x^+ + x_i^T x^-)
\]

decision function:

\[
f(x) = \text{sgn}\left[ \sum_{i \in SV} y_i \alpha_i^* x_i^T x + b^* \right]
\]
Practical implementations

- in practice we need an algorithm for solving the optimization problem of the training stage
  - this is still a complex problem
  - there has been a large amount of research in this area
  - coming up with “your own” algorithm is not going to be competitive
  - luckily there are various packages available, e.g.:
    - libSVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
    - SVM light: http://www.cs.cornell.edu/People/tj/svm_light/
    - SVM fu: http://five-percent-nation.mit.edu/SvmFu/
    - various others (see http://www.support-vector.net/software.html)
  - also many papers and books on algorithms (see e.g. B. Schölkopf and A. Smola. Learning with Kernels. MIT Press, 2002)
Kernelization

- note that all equations depend only on $x_i^T x_j$
- the kernel trick is trivial: replace by $K(x_i, x_j)$

1) training

$$\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j k(x_i, x_j) + \sum_i \alpha_i \right\}$$
subject to $\sum_i y_i \alpha_i = 0$

$$b^* = -\frac{1}{2} \sum_{i \in SV} y_i \alpha_i^* (K(x_i, x^+) + K(x_i, x^-))$$

2) decision function:

$$f(x) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* K(x_i, x) + b^* \right]$$
Kernelization

notes:

• as usual this follows from the fact that nothing of what we did really requires us to be in $\mathbb{R}^d$.

• we could have simply used the notation $<x_i, x_j>$ for the dot product and all the equations would still hold.

• the only difference is that we can no longer recover $w^*$ explicitly without determining the feature transformation $\Phi$, since

$$w^* = \sum_{i \in SV} \alpha_i^* y_i \Phi(x_i)$$

• this could have infinite dimension, e.g. we have seen that it is a sum of Gaussians when we use the Gaussian kernel.

• but, luckily, we don’t really need $w^*$, only the decision function

$$f(x) = \text{sgn}\left[ \sum_{i \in SV} y_i \alpha_i^* K(x_i, x) + b^* \right]$$
Input space interpretation

- when we introduce a kernel, what is the SVM doing in the input space?

- let’s look again at the decision function

\[
f(x) = \text{sgn}\left[ \sum_{i \in SV} y_i \alpha_i^* K(x_i, x) + b^* \right]
\]

with

\[
b^* = -\frac{1}{2} \sum_{i \in SV} y_i \alpha_i^* (K(x_i, x^+) + K(x_i, x^-))
\]

- note that
  - \(x^+\) and \(x^-\) are support vectors
  - assuming that the kernel has reduced support when compared to the distance between support vectors
Input space interpretation

• note that
  
• assuming that the kernel as reduced support when compared to the distance between support vectors

\[ b^* = -\frac{1}{2} \sum_{i \in SV} y_i \alpha_i^* (K(x_i, x^+) + K(x_i, x^-)) \]

\[ \approx -\frac{1}{2} [\alpha_+^* K(x^+, x^+) - \alpha_-^* K(x^-, x^-)] \]

\[ \approx 0 \]

• where we have also assumed that \( \alpha_+ \sim \alpha_- \)

• these assumptions are not crucial, but simplify what follows

• namely the decision function is

\[ f(x) = \text{sgn} \left[ \sum_{i \in SV} y_i \alpha_i^* K(x_i, x) \right] \]
Input space interpretation

or

\[
f(x) = \begin{cases} 
1, & \text{if } \sum_{i \in SV} y_i \alpha_i^* K(x_i, x) \geq 0 \\
-1, & \text{if } \sum_{i \in SV} y_i \alpha_i^* K(x_i, x) < 0
\end{cases}
\]

rewriting

\[
\sum_{i \in SV} y_i \alpha_i^* K(x_i, x) = \sum_{i | y_i > 0} \alpha_i^* K(x_i, x) - \sum_{i | y_i < 0} \alpha_i^* K(x_i, x)
\]

this is

\[
f(x) = \begin{cases} 
1, & \text{if } \sum_{i | y_i \geq 0} \alpha_i^* K(x_i, x) \geq \sum_{i | y_i < 0} \alpha_i^* K(x_i, x) \\
-1, & \text{otherwise}
\end{cases}
\]
Input space interpretation

or

\[ f(x) = \begin{cases} 
1, & \text{if } \frac{1}{\sum_{i \mid y_i \geq 0} \alpha_i} \sum_{i \mid y_i \geq 0} \pi_i^* K(x_i, x) \geq \frac{1}{\sum_{i \mid y_i < 0} \alpha_i} \sum_{i \mid y_i < 0} \beta_i^* K(x_i, x) \\
-1, & \text{otherwise}
\end{cases} \]

with

\[ \pi_i^* = \frac{\alpha_i^*}{\sum_{i \mid y_i \geq 0} \alpha_i}, \quad i \mid y_i \geq 0 \]

\[ \beta_i^* = \frac{\alpha_i^*}{\sum_{i \mid y_i < 0} \alpha_i}, \quad i \mid y_i < 0 \]

which is the same as

\[ f(x) = \begin{cases} 
1, & \text{if } \frac{\sum_{i \mid y_i \geq 0} \pi_i^* K(x_i, x)}{\sum_{i \mid y_i < 0} \beta_i^* K(x_i, x)} \geq \frac{\sum_{i \mid y_i \geq 0} \alpha_i}{\sum_{i \mid y_i < 0} \alpha_i} \\
-1, & \text{otherwise}
\end{cases} \]
Input space interpretation

note that this is the Bayesian decision rule for

- 1) class 1 with likelihood
  \[ \sum_{i | y_i \geq 0} \pi_i^* K(x_i, x) \]
  and prior
  \[ \sum_{i | y_i < 0} \alpha_i^* / \sum_i \alpha_i^* \]

- 2) class 2 with likelihood
  \[ \sum_{i | y_i < 0} \beta_i^* K(x_i, x) \]
  and prior
  \[ \sum_{i | y_i \geq 0} \alpha_i^* / \sum_i \alpha_i^* \]

these likelihood functions

- are a kernel density estimate if \( k(.,x_i) \) is a valid pdf
- peculiar kernel estimate that only places kernels around the support vectors
- all other points are ignored
Input space interpretation

- this is a discriminant form of density estimation
  - concentrate modeling power where it matters the most, i.e. near classification boundary
  - smart, since points away from the boundary are always well classified, even if density estimates in their region are poor
  - the SVM can be seen as a highly efficient combination of the BDR with kernel density estimates
  - recall that one major problem of kernel estimates is the complexity of the decision function, $O(n)$.
  - with the SVM, the complexity is only $O(|SV|)$ but nothing is lost
Input space interpretation

- note on the approximations made:
  - this result was derived assuming $b \approx 0$
  - in practice, $b$ is frequently left as a parameter which is used to trade-off false positives for misses
  - here, that can be done by controlling the BDR threshold

$$f(X) = \begin{cases} 
1, & \text{if } \frac{\sum_{i|y_i \geq 0} \pi^*_i K(x_i, x)}{\sum_{i|y_i < 0} \beta^*_i K(x_i, x)} \geq T \\
-1, & \text{otherwise}
\end{cases}$$

- hence, there is really not much practical difference, even when the assumption of $b^* = 0$ does not hold!
Any Questions?