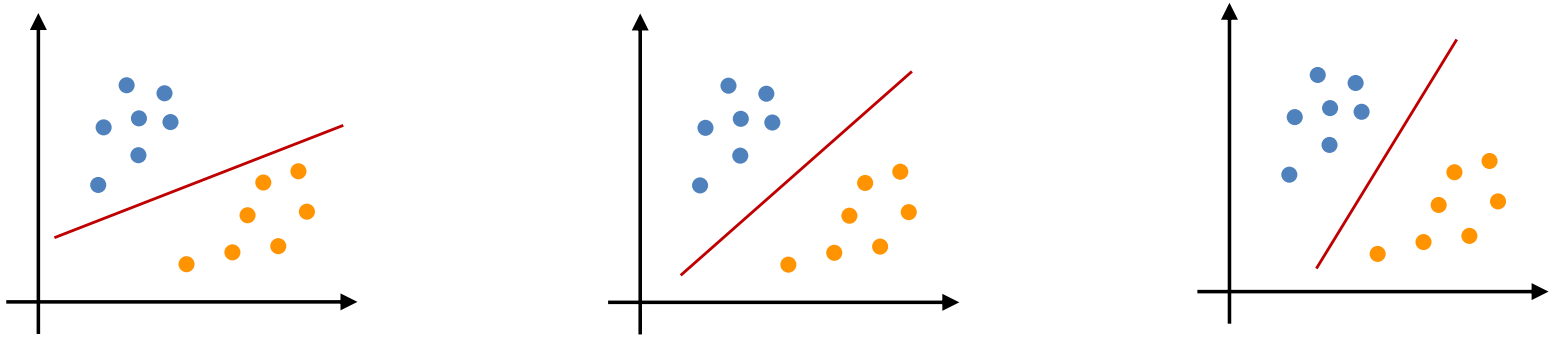
Issues in linear classification

Different classification learning algorithms use different criteria:

- Basic linear classifier finds class means (centroids), joins them by a straight line, and its perpendicular bisector is the separating hyperplane
- Perceptron training uses iterative reweighting (gradient descent)
- Perceptron may find different models depending on starting conditions

Issues in linear classification

Which line is a better classifier?

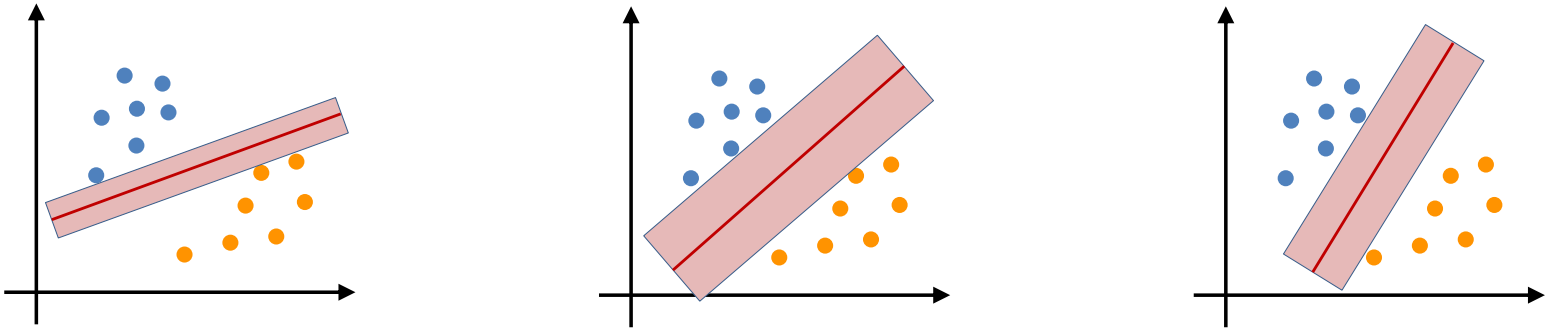


What is the advantage?

Issues in linear classification

Which line is a better classifier?

– The line with bigger margin



- Why bigger margin is better?
- Can I find a w that maximizes the margin?

Issues in linear classification

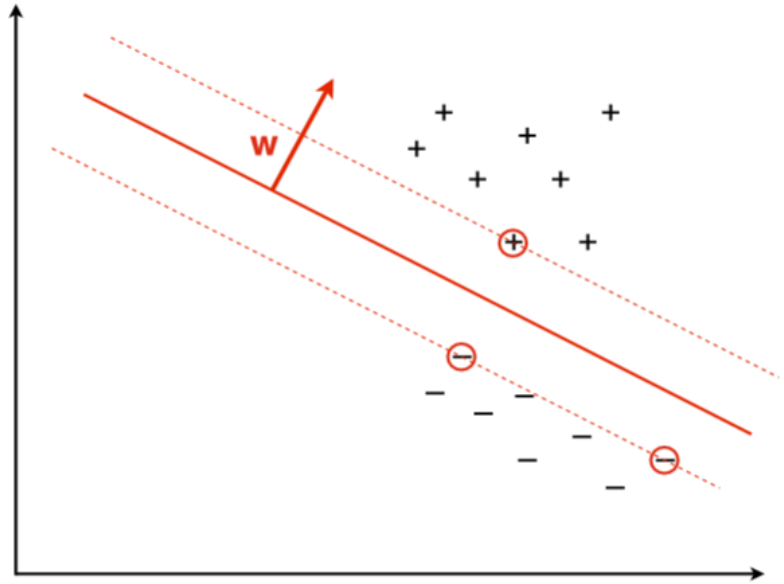
Is there an optimal linear classification learning method ?

answer: Yes, under Vapnik's framework for statistical learning which is called Support Vector Machine (SVM)

- define the empirical risk, or error on the data
- maximum margin separating hyperplane
- minimizes empirical risk
- SVM lets to make a trade-off between model complexity and the error
- unique solution
- structural risk minimization

Support Vector Machine

Support vector machine

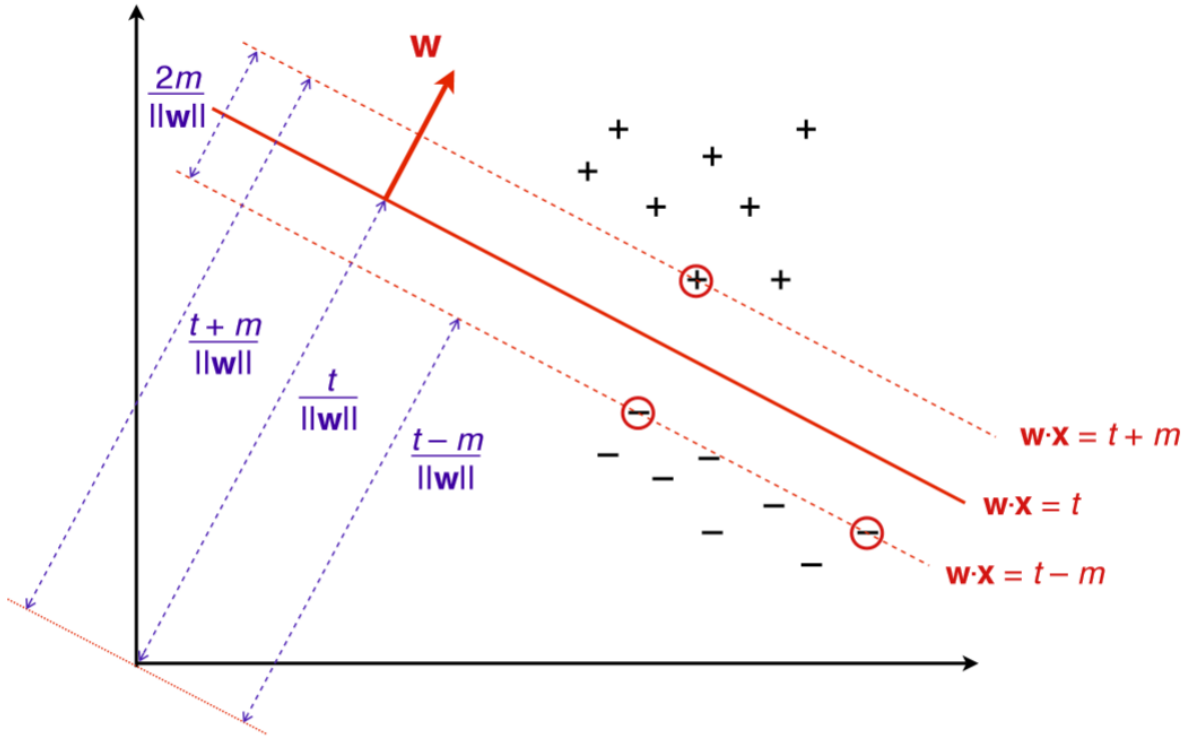


The decision boundary learned by a support vector machine maximizes the margin, which is indicated by the dotted lines. The circled data points are the support vectors.

Support vector machine

- Support vector machines (machine ≡ algorithm) learn linear classifiers
- Can avoid overfitting – learn a form of decision boundary called the maximum margin hyperplane
- Fast for mappings to nonlinear spaces
  - employ a mathematical trick (kernel) to avoid the actual creation of new “pseudo-attributes” in transformed instance space
  - i.e. the nonlinear space is created implicitly

Support vector machine



The geometry of a support vector classifier. The circled data points are the support vectors, which are the training examples nearest to the decision boundary. The support vector machine finds the decision boundary that maximizes the margin $m/||w||$.

Training a support vector machine

- Learning problem: fit maximum margin hyperplane, i.e. a kind of linear model
- For a linearly separable two-class data set the maximum margin hyperplane is the classification surface which
  - correctly classifies all examples in the data set
  - has the greatest separation between classes

Support vector machine

Let's x_s be the closest point to the separating hyperplane (line in 2D) with the following equation:

$$w.\mathrm{x} = t$$

Let's have 2 minor technicalities to simplify the math later:

1. Pull out w_0: $w = [w_1, \ldots, w_n]$ and $w_0 = -t$, therefore we will have:

2. Normalize w:

   We know that $|\mathrm{w}.\mathrm{x}_s - \mathrm{t}| > 0$ and we know that we can scale w and t together without having any effect on the hyperplane, so we choose the scale such that:

   $$|\mathrm{w}.\mathrm{x}_s - \mathrm{t}| = 1$$

   This means $m = 1$

Support vector machine

- We have the line equation which is: $w.\mathrm{x} - t = 0$
- And we have the constraint $|\mathrm{w}.\mathrm{x}_s - \mathrm{t}| = 1$ where x_s is the closest point to the separating line (hyperplane)

- w is perpendicular to the line (hyperplane). How?

For every two points x' and x'' on the line (hyperplane), we can write:

$$w.\mathrm{x}' - t = 0 \quad \text{and } w.\mathrm{x}'' - t = 0$$
$$\Longrightarrow \qquad w.(\mathrm{x}' - \mathrm{x}'') = 0$$

When the dot product of two vectors is zero, it means they are perpendicular

Support vector machine

From geometry, we know that the distance between point x_s and line (hyperplane) $w.\mathrm{x} - t = 0$ is:

$$distance = \frac{|w.\mathrm{x}_s - t|}{||w||}$$

And we have $|\mathrm{w}.\mathrm{x}_s - \mathrm{t}| = 1$, so:

$$distance = \frac{1}{||w||}$$

This $distance$ is the *margin* of our classifier which we want to maximize:

$$\max \frac{1}{||w||} \text{ subject to } \min_{i=1,\ldots,m} |w.\mathrm{x}_i - t| = 1$$

在保证最小的最小规格化距离为1的情况下，找到最大的distance！

This is not a friendly optimisation as it has "min" in the constraint!

Support vector machine

Our focus is on the points which the line (hyperplane) can predict them correctly. So we can have:

$$|w.\mathrm{x}_i - t| = y_i(w.\mathrm{x}_i - t)$$

And we can change the “min” in constraint as follow:

$$y_i(w.\mathrm{x}_i - t) \geq 1 \quad for\ i = 1, \ldots, m$$

We can also transform the maximization problem into the following minimization problem:

$$\min \frac{1}{2}||w||^2 \quad subject\ to \quad y_i(w.\mathrm{x}_i - t) \geq 1 \quad for\ i = 1, \ldots, m$$

$$w \in \mathbb{R}^n, t \in \mathbb{R}$$

How to solve this? Largangian multipliers

Lagrangian multipliers

- Constrained optimization problems are generally expressed as:

$$\min_{x_1,\ldots,x_n} f(x_1, \ldots, x_n)$$

Subject to:

$$g_1(x_1, \ldots, x_n) \leq 0, g_2(x_1, \ldots, x_n) \leq 0, \ldots, g_k(x_1, \ldots, x_n) \leq 0$$

- Lagrange multiplier methods involve the modification of the objective function through the addition of terms that describe the constraints. The objective function $f(\mathrm{x})$ is augmented by the constraint equations through a set of non-negative multiplicative Lagrange multipliers,$\alpha_j \geq 0$ and it is called the dual Lagrangian:

$$\mathcal{L}(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m) = f(x_1, \ldots, x_n) + \sum_{j=1}^{k} \alpha_j g_j(x_1, \ldots, x_n)$$

Lagrangian multipliers

In Lagrangian form, the optimization problem becomes:

$$\max_{\alpha_1,\ldots,\alpha_m} \min_{x_1,\ldots,x_n} \mathcal{L}(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m) \quad such\ that\ \ \alpha_j \geq 0\ \forall\ j$$

- first minimizing with respect to $x_1, \ldots, x_n$
- then maximize with respect to $\alpha_1, \ldots, \alpha_m$

Deriving the dual problem

Adding the constraints with multipliers α_i for each training example gives the Lagrange function:

$$\mathcal{L}(w, t, \alpha_1, \ldots, \alpha_m) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i (y_i (w.\mathrm{x}_i - t) - 1)$$

$$= \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i y_i (w.\mathrm{x}_i) + \sum_{i=1}^{m} \alpha_i y_i t + \sum_{i=1}^{m} \alpha_i$$

$$= \frac{1}{2} w.w - w.\left(\sum_{i=1}^{m} \alpha_i y_i \mathrm{x}_i\right) + t\left(\sum_{i=1}^{m} \alpha_i y_i\right) + \sum_{i=1}^{m} \alpha_i$$

Deriving the dual problem

- First we have to minimize $\mathcal{L}$ with respect to w and t
- By taking the partial derivative of the Lagrange function with respect to t and setting it to 0 we find:

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

- Similarly, by taking the partial derivative of the Lagrange function with respect to w and setting to 0 we obtain:

$$\mathrm{w} = \sum_{i=1}^{m} \alpha_i y_i \mathrm{x_i}$$

– the same expression as we derived for the perceptron.

Deriving the dual problem

- These expressions allow us to eliminate w and t and lead to the dual Lagrangian

$$\mathcal{L}(\alpha_1, \ldots, \alpha_n) = -\frac{1}{2}\left(\sum_{i=1}^{m} \alpha_i y_i \mathrm{x}_i\right).\left(\sum_{i=1}^{m} \alpha_i y_i \mathrm{x}_i\right) + \sum_{i=1}^{m} \alpha_i$$

$$= -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathrm{x}_i . \mathrm{x}_j + \sum_{i=1}^{m} \alpha_i$$

Deriving the dual problem

- For the perceptron, the instance weights α_i are non-negative integers denoting the number of times an example has been misclassified in training. For a support vector machine, the α_i are non-negative reals.

- What they have in common is that, if $\alpha_i = 0$ for a particular example x_i, that example could be removed from the training set without affecting the learned decision boundary. In the case of support vector machines this means that $\alpha_i > 0$ only for the support vectors: the training examples nearest to the decision boundary.

SVM in dual form

- The dual optimization problem for support vector machines is to maximize the dual Lagrangian under positivity constraints and one equality constraint:

$$\alpha_1^*, \ldots, \alpha_m^* = \underset{\alpha_1,\ldots,\alpha_n}{\text{argmax}} -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \mathrm{x}_i.\mathrm{x}_j + \sum_{i=1}^{m}\alpha_i$$

$$\text{subject to } \alpha_i \geq 0, 1 \leq i \leq m, \sum_{i=1}^{m}\alpha_i\, y_i = 0$$

Finding support vectors

$$\alpha_1^*, \ldots, \alpha_m^* = \underset{\alpha_1,\ldots,\alpha_n}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \mathrm{x}_i.\mathrm{x}_j - \sum_{i=1}^{m}\alpha_i$$

$$\text{subject to } \alpha_i \geq 0, 1 \leq i \leq m, \sum_{i=1}^{m}\alpha_i\, y_i = 0$$

- Determining parameters is a constrained quadratic optimization problem
- standard algorithms, or special-purpose algorithms (usually faster, e.g. Platt's sequential minimal optimization (SMO), or LibSVM)

Training a support vector machine

When you solve this optimization problem and get $\alpha_1^*, \ldots, \alpha_m^*$ back, you will see that it will be mostly zero and only for the points which are the closest to the separating line, α_i will be non-zero and positive. Those points are called the support vectors.

$$w = \sum_{\mathrm{x}_i \in \{support\ vectors\}} \alpha_i y_i x_i$$

Solve for t using any of support vectors:

$$y_i(w.\mathrm{x}_i - t) = 1$$

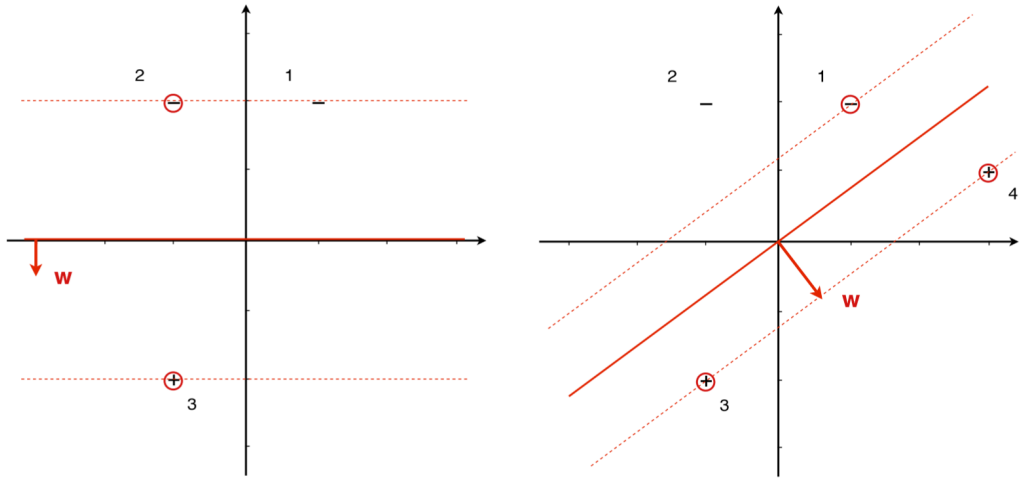
It can be shown that error is bounded by number of support vectors, irrespective of dimensionality

Note: all this assumes separable data!

Training a support vector machine

- The more “separated” the classes, the larger the margin, the better the generalization
- Instances closest to maximum margin hyperplane are support vectors
- Important observation: support vectors define maximum margin hyperplane!
  - All other instances can be deleted without changing position and orientation of the hyperplane!

Two maximum-margin classifiers



(left) A maximum-margin classifier built from three examples, with $w = (0, -1/2)$ and margin 2. The circled examples are the support vectors: they receive non-zero Lagrange multipliers and define the decision boundary. (right) By adding a second positive the decision boundary is rotated to $w = (3/5, -4/5)$ and the margin decreases to 1.

Two maximum-margin classifiers

$$X = \begin{pmatrix} 1 & 2 \\ -1 & 2 \\ -1 & -2 \end{pmatrix} \quad y = \begin{pmatrix} -1 \\ -1 \\ +1 \end{pmatrix} \quad X' = \begin{pmatrix} -1 & -2 \\ 1 & -2 \\ -1 & -2 \end{pmatrix}$$

The matrix X' on the right incorporates the class labels; i.e., the rows are $y_i x_i$. The Gram matrix is (without and with class labels):

$$XX^T = \begin{pmatrix} 5 & 3 & -5 \\ 3 & 5 & -3 \\ -5 & -3 & 5 \end{pmatrix} \quad X'X'^T = \begin{pmatrix} 5 & 3 & 5 \\ 3 & 5 & 3 \\ 5 & 3 & 5 \end{pmatrix}$$

Two maximum-margin classifiers

The dual optimization problem is thus

$$\underset{\alpha_1,\alpha_2,\alpha_3}{\text{argmax}} -\frac{1}{2}(5\alpha_1^2 + 3\alpha_1\alpha_2 + 5\alpha_1\alpha_3 + 3\alpha_2\alpha_1 + 5\alpha_2^2 + 3\alpha_2\alpha_3 + 5\alpha_3\alpha_1 + 3\alpha_3\alpha_2 + 5\alpha_3^2) + \alpha_1 + \alpha_2 + \alpha_3$$

$$= \underset{\alpha_1,\alpha_2,\alpha_3}{\text{argmax}} -\frac{1}{2}(5\alpha_1^2 + 6\alpha_1\alpha_2 + 10\alpha_1\alpha_3 + 5\alpha_2^2 + 36\alpha_3 + 5\alpha_3^2) + \alpha_1 + \alpha_2 + \alpha_3$$

subject to $\alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_3 \geq 0$ and $-\alpha_1 - \alpha_2 + \alpha_3 = 0$.

Two maximum-margin classifiers

- Using the equality constraint we can eliminate one of the variables, say α_3, and simplify the objective function to

$$\underset{\alpha_1,\alpha_2}{\operatorname{argmax}} -\frac{1}{2}(20\alpha_1^2 + 32\alpha_1\alpha_2 + 16\alpha_2^2) + 2\alpha_1 + 2\alpha_2$$

- Setting partial derivatives to 0 we obtain $-20\alpha_1 - 16\alpha_2 + 2 = 0$ and $-16\alpha_1 - 16\alpha_2 + 2 = 0$ (notice that, because the objective function is quadratic, these equations are guaranteed to be linear).

- We therefore obtain the solution $\alpha_1 = 0$ and $\alpha_2 = \alpha_3 = 1/8$. We then have $w = 1/8\,(x_3 - x_2) = \begin{pmatrix} 0 \\ -1/2 \end{pmatrix}$, resulting in a margin of $1/||w|| = 2$.

- Finally, t can be obtained from any support vector, say x_2, since $y_2(w.x_2 - t) = 1$; this gives $-1 \cdot (-1 - t) = 1$, hence $t = 0$.

Two maximum-margin classifiers

- We now add an additional positive at $(3,1)$. This gives the following data matrices:

$$X' = \begin{pmatrix} -1 & -2 \\ 1 & -2 \\ -1 & -2 \\ 3 & 1 \end{pmatrix} \quad X'X'^T = \begin{pmatrix} 5 & 3 & 5 & -5 \\ 3 & 5 & 3 & 1 \\ 5 & 3 & 5 & -5 \\ -5 & 1 & -5 & 10 \end{pmatrix}$$

- It can be verified by similar calculations to those above that the margin decreases to 1 and the decision boundary rotates to

$$w = \begin{pmatrix} 3/5 \\ -4/5 \end{pmatrix}.$$

- The Lagrange multipliers now are $\alpha_1 = 1/2\,, \alpha_2 = 0, \alpha_3 = 1/10$ and $\alpha_4 = 2/5$. Thus, only x_3 is a support vector in both the original and the extended data set.

Nonlinear SVMs

Similar to before, we can transform our input feature $\mathrm{x} \in \mathcal{X}$ into a new feature space $z \in \mathcal{Z}$

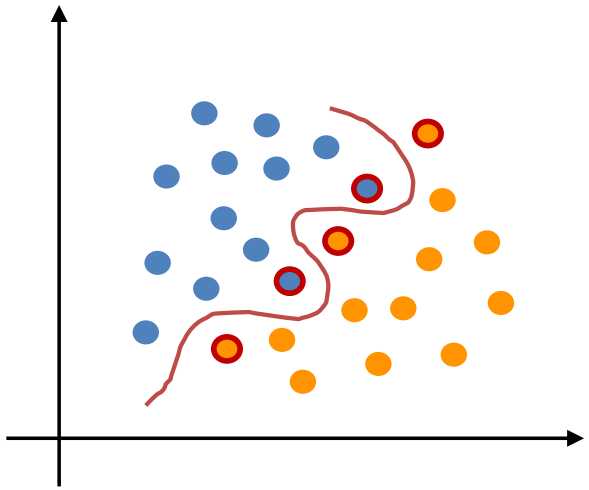
We can use the same Lagrangian function by replacing x:

$$\mathcal{L}(\alpha) = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \mathrm{z}_i.\mathrm{z}_j + \sum_{i=1}^{m}\alpha_i$$

- The number of α_i is still the same even if your new feature space has much higher dimensionality
- As long as we provide the $\mathrm{z}_i.\mathrm{z}_j$, the complexity of the optimization problem is going to be the same

Nonlinear SVMs

- Note that your support vectors are defined in the new feature space ($\mathcal{Z}$)
- The margin is maintained in the $\mathcal{Z}$ space
- They can be recognized by finding α_is which are not zero
- However, they correspond to some of the points in the original space.



Nonlinear SVMs with Kernel trick

If $z = \varphi(\mathrm{x})$, Lagrangian becomes:

$$\mathcal{L}(\alpha) = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \varphi(\mathrm{x}_i).\varphi(\mathrm{x}_j) + \sum_{i=1}^{m}\alpha_i$$

If we can find a kernel function, such that:

$$K(\mathrm{x}_i, \mathrm{x}_j) = \varphi(\mathrm{x}_i).\varphi(\mathrm{x}_j)$$

(Kernel corresponds to a map into a new feature space)

Then:

$$\mathcal{L}(\alpha) = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j K(\mathrm{x}_i, \mathrm{x}_j) + \sum_{i=1}^{m}\alpha_i$$

Nonlinear SVMs

The output of your support vector machine for x is:

$$\hat{y} = sign(w.z - t)$$

$$w = \sum_{z_i \in SV} \alpha_i y_i z_i$$

$$\Rightarrow \hat{y} = sign(\sum_{\alpha_i > 0} \alpha_i y_i K(\mathrm{x}_i, \mathrm{x}) - t)$$

Where:

$t = \sum_{\alpha_i > 0} \alpha_i y_i K(\mathrm{x}_i, \mathrm{x}_j) - y_j$ for any support vector x_j

So this is your model. As you see it is not just one model and it can change depending on the kernel that you choose.

Nonlinear SVMs

$$\hat{y} = sign(\sum_{\alpha_i>0} \alpha_i y_i K(\mathrm{x}_i, \mathrm{x}) - t)$$

- This will let you to compare the results from different Kernels
- You can test any valid kernel and compare the results
- Using the Kernel trick we do not need to do the transformation from original feature space to the new feature space
- Possibility of finding the support vector in high dimensional space without paying the computational price of going to that new space
- Support vector machines are a popular choice as a kernel method, since they naturally promote sparsity in the support vectors.

Kernel trick

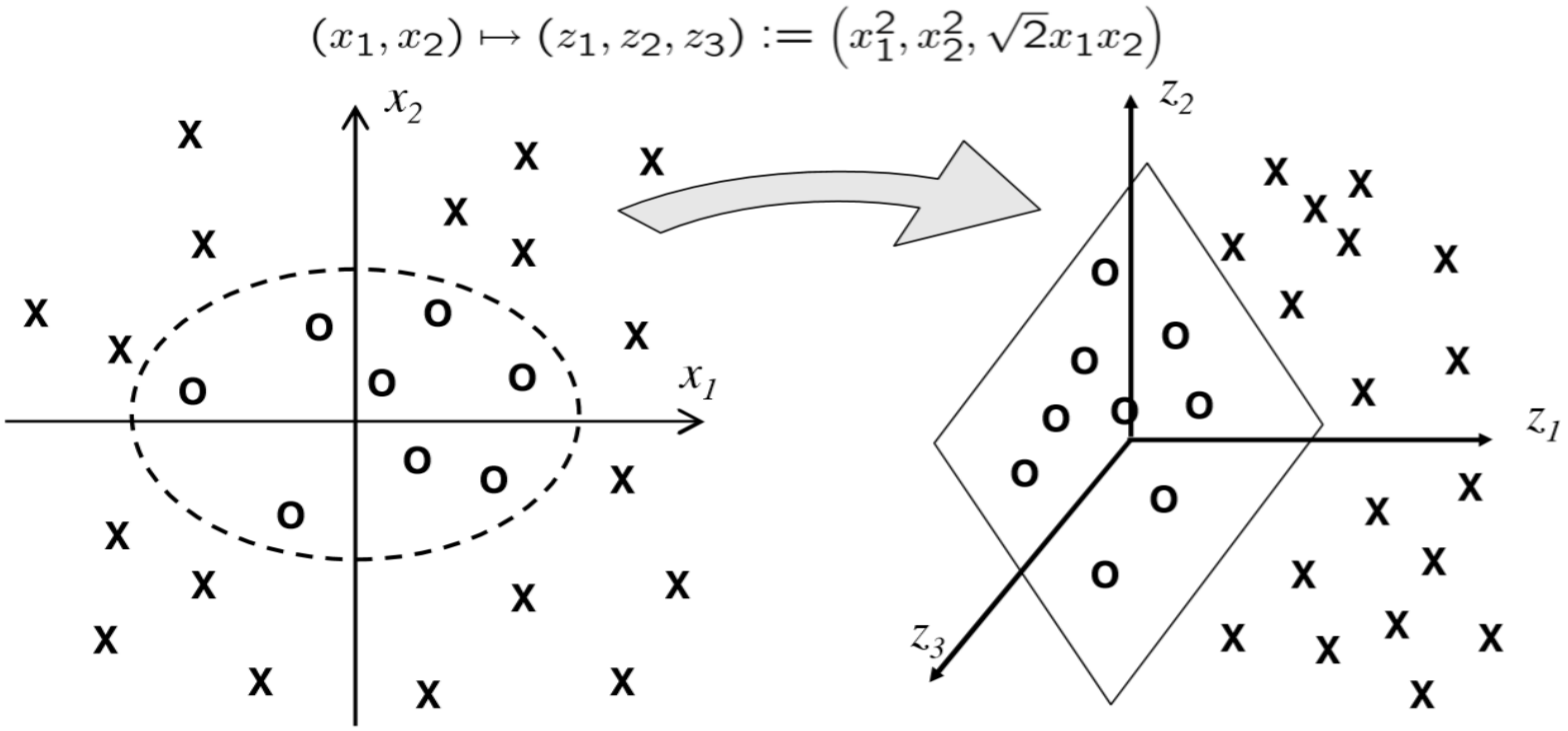
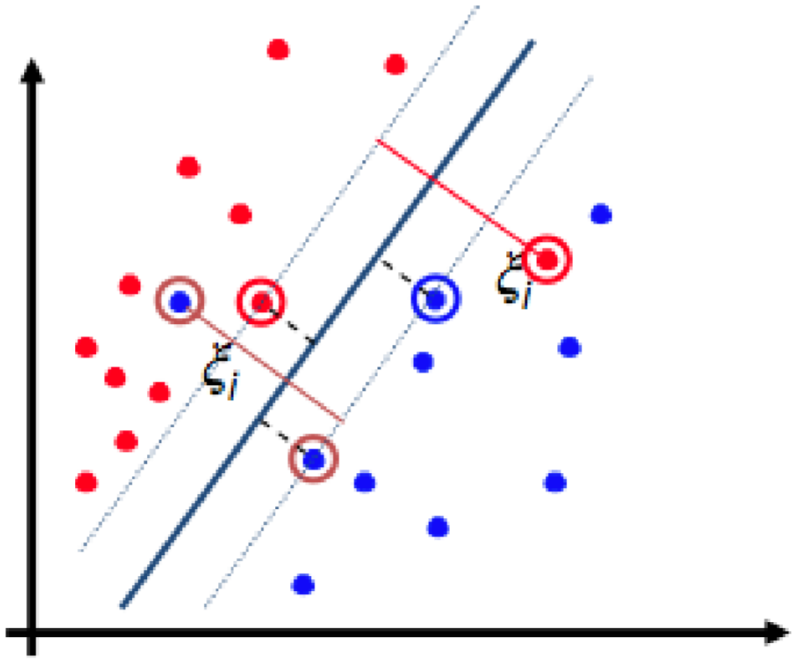


Figure by Avrim Blum, CS Dept, CMU.

Noise

- Misclassified examples may break the separability assumption



- Introduce “slack” variables ξ_i to allow misclassification of instances

Soft margin SVM

When classes were linearly separable, we had:

$$y_i(w.\mathrm{x}_i - t) \geq 1$$

But if we get some data that violate this slack value:

$$y_i(w.\mathrm{x}_i - t) \geq 1 - \xi_i \;\; and \; \xi_i \geq 0$$

So, the total violation:

$$total\ violation = \sum_{i=1}^{m} \xi_i$$

This is a measure of violation of margin and now, we optimize for:

$$\min \frac{1}{2} \parallel w \parallel^2 + C \sum \xi_i$$

Soft margin SVM

- "soft margin" SVMs to handle noisy data
- Parameter C bounds influence of any one training instance on decision boundary
- If C goes to infinity you go back to the hard margin SVM (doesn't tolerate error)
- Still a quadratic optimization problem

Allowing margin errors

- The idea is to introduce slack variables ξ_i, one for each example, which allow some of them to be inside the margin or even at the wrong side of the decision boundary. We will call these margin errors.

$$w^*, t^*, \xi^* = \underset{w,t,\xi_i}{\operatorname{argmin}} \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{m} \xi_i$$

$$\text{sbject to } y_i(w \cdot x_i - t) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, 1 \leq i \leq m.$$

Allowing margin errors

- C is a user-defined parameter trading off margin maximization against slack variable minimization: a high value of C means that margin errors incur a high penalty, while a low value permits more margin errors (possibly including misclassifications) in order to achieve a large margin.
- If we allow more margin errors we need fewer support vectors, hence C controls to some extent the 'complexity' of the SVM and hence is often referred to as the *complexity* parameter.

Allowing margin errors

The Lagrange function is then as follows:

$$\mathcal{L}(w, t, \xi_i, \alpha_i, \beta_i) = \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i (y_i (w \cdot x_i - t) - (1 - \xi_i)) - \sum_{i=1}^{m} \beta_i \xi_i$$

$$= \mathcal{L}(w, t, \alpha_i) + \sum_{i=1}^{m} (C - \alpha_i - \beta_i) \xi_i$$

Similar to before, we have to first minimize this with respect to w, t and ξ_i

Allowing margin errors

$$\nabla_w \mathcal{L} = w - \sum_{i=1}^{m} \alpha_i y_i \mathrm{x}_i = 0$$

$$\frac{\partial \mathcal{L}}{\partial t} = \sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\frac{\partial \mathcal{L}}{\partial\, \xi_i} = C - \alpha_i - \beta_i = 0 \Longrightarrow C - \alpha_i = \beta_i \geq 0$$

$$0 \leq \alpha_i \leq C$$

The only difference that we get compared to hard margin SVM is that α_i is upper bounded by the value of C

Allowing margin errors

- For an optimal solution every partial derivative with respect to ξ_i should be 0, from which it follows that the added term vanishes from the dual problem.
- Furthermore, since both α_i and β_i are positive, this means that α_i cannot be larger than C
- The next step is to optimize the Lagrangian using quadratic programming similar to before and get estimates for α_i

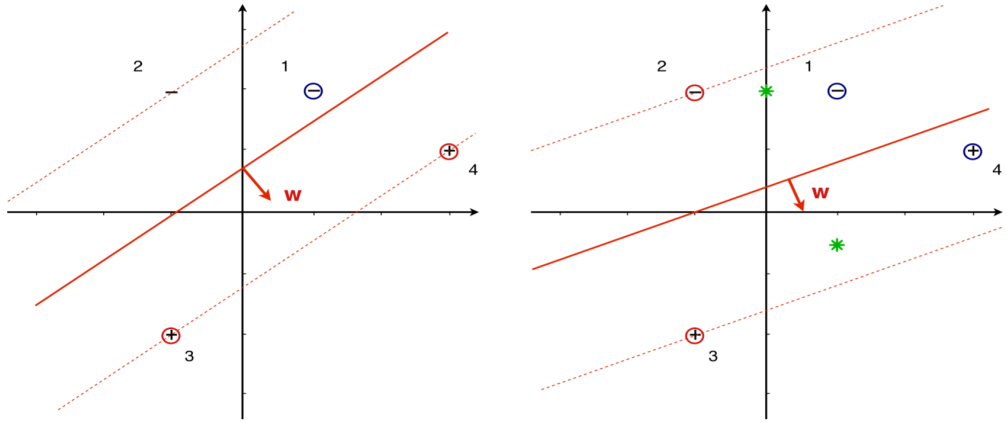
Three cases for the training instances

- a solution to the soft margin optimization problem in dual form divides the training examples into three cases:
  - $\alpha_i = 0$ these are outside or on the margin;
  - $0 < \alpha_i < C$ these are the support vectors on the margin;
  - $\alpha_i = C$ these are on or inside the margin (called non-margin support vectors). All the points that violate the margin are non-margin support vectors and contribute to w.

- Notice that we still have $w = \sum_{i=1}^{n} \alpha_i y_i x_i$, and so both second and third case examples participate in spanning the decision boundary.

- The value of C controls the amount of violation and has to be found by, e.g., cross-validation

Soft margins



(left) The soft margin classifier learned with $C = 5/16$, at which point x_2 is about to become a support vector. (right) The soft margin classifier learned with $C = 1/10$: all examples contribute equally to the weight vector. The asterisks denote the class means, and the decision boundary is parallel to the one learned by the basic linear classifier.

Soft margins

- Recall that the Lagrange multipliers for the classifier above are $\alpha_1 = 1/2$, $\alpha_2 = 0$, $\alpha_3 = 1/10$, $\alpha_4 = 2/5$. So α_1 is the largest multiplier, and as long as $C > \alpha_1 = 1/2$ no margin errors are tolerated.
- For $C = 1/2$ we have$\alpha_1 = C$, and hence for $C < 1/2$ we have that x_1 becomes a margin error and the optimal classifier is a soft margin classifier. Effectively, with decreasing C the decision boundary and the upper margin move upward, while the lower margin stays the same.
- The upper margin reaches x_2 or $C = 5/16$, at which point we have $\mathrm{w} = \begin{pmatrix} 3/8 \\ -1/2 \end{pmatrix}$ and the margin has increased to 1.6. Furthermore, we have $\xi_1 = 6/8$, $\alpha_1 = C = 5/16$, $\alpha_2 = 0$, $\alpha_3 = 1/16$ and $\alpha_4 = 1/4$.

Soft margins

- If we now decrease C further, the decision boundary starts to rotate clockwise, so that x_4 becomes a margin error as well, and only x_2 and x_3 are support vectors. The boundary rotates until $C = 1/10$, at which point we have $w = \begin{pmatrix} 1/5 \\ -1/2 \end{pmatrix}$, $t = 1/5$ and the margin has increased to 1.86. Furthermore, we have $\xi_1 = 4/10$ and $\xi_4 = 7/10$, and all multipliers have become equal to C.
- Finally, when C decreases further the decision boundary stays where it is, but the norm of the weight vector gradually decreases and all points become margin errors.

Soft margins

NOTE: a minimal-complexity soft margin classifier summarizes the classes by their class means in a way very similar to the basic linear classifier.

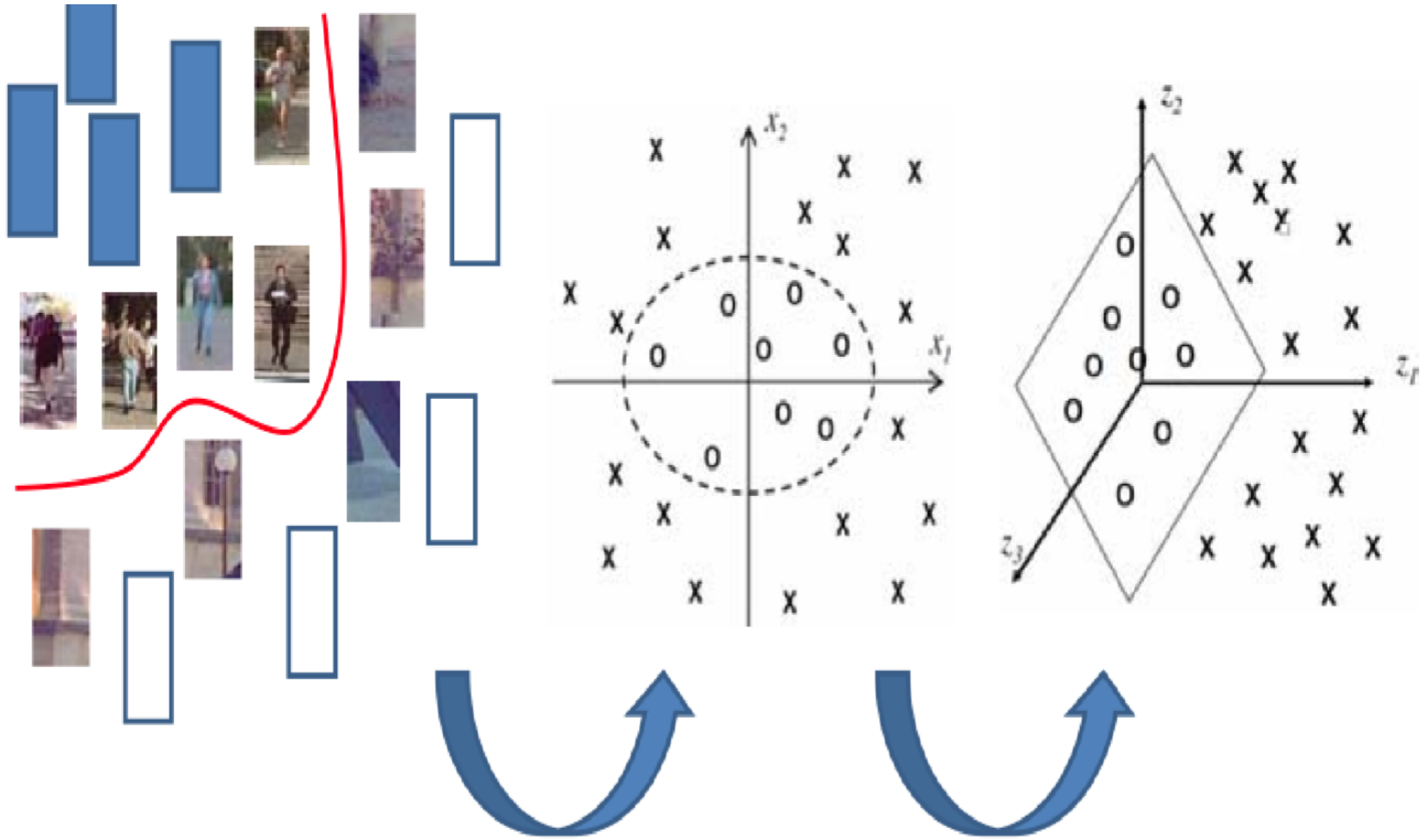
Sparse data

- SVM algorithms can be sped up dramatically if the data is sparse (i.e. many values are 0)
- Why? Because they compute lots and lots of dot products
- With sparse data dot products can be computed very efficiently • Just need to iterate over the values that are non-zero
- SVMs can process sparse datasets with tens of thousands of attributes

SVM Applications

- Machine vision: e.g face identification
  - Prior to deep learning, achieves lowest error
- Handwritten digit recognition:
  - Comparable to best alternative
- Bioinformatics: e.g. prediction of protein secondary structure, microarray classification
- Text classification
- Algorithm can be modified to deal with numeric prediction problems – support vector regression

Example - pedestrian detection



Summary: Learning with Kernel Methods

- Kernel methods around for a long time in statistics
- Kernelization a “modular” approach to machine learning
- Algorithms that can be kernelized can learn different model classes simply by changing the kernel
- SVMs exemplify this – mostly for classification (but also regression, “one-class’ classification, etc.)
- SVMs one of the most widely used “off-the-shelf” classifier learning methods, especially for “small m (examples), large n (dimensionality)” classification problems

Acknowledgements

- "Elements of Statistical Learning (2nd Ed.)" by T. Hastie, R. Tibshirani & J. Friedman. Springer (2009) http://statweb.stanford.edu/~tibs/ElemStatLearn/
- Material derived from slides for the book "Machine Learning: A Probabilistic Perspective" by P. Murphy MIT Press (2012) http://www.cs.ubc.ca/~murphyk/MLbook
- Material derived from slides for the book "Machine Learning" by P. Flach Cambridge University Press (2012) http://cs.bris.ac.uk/~flach/mlbook
- Material derived from slides for the book "Bayesian Reasoning and Machine Learning" by D. Barber Cambridge University Press (2012) http://www.cs.ucl.ac.uk/staff/d.barber/brml
- Material derived from slides for the book "Machine Learning" by T. Mitchell McGraw-Hill (1997) http://www- 2.cs.cmu.edu/~tom/mlbook.html
- Material derived from slides for the course "Machine Learning" by A. Srinivasan BITS Pilani Goa Campus, India (2016)

UNSW AUSTRALIA