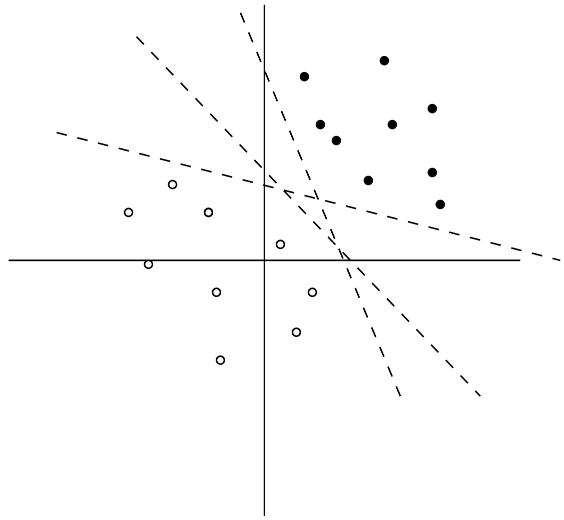
Some Teasers

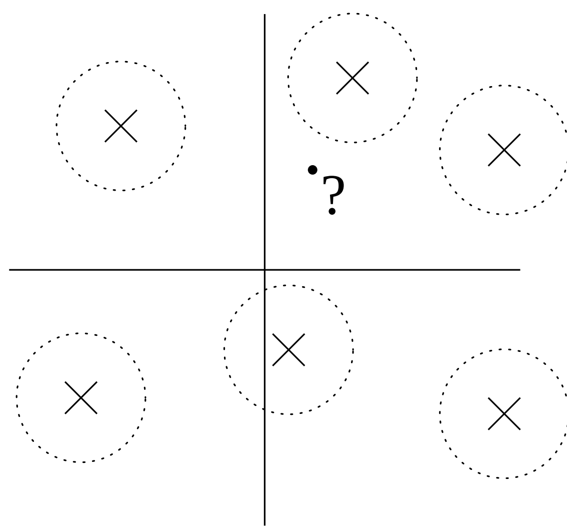
- Which is the "best" hyperplane that separates the data ?



- In a template based classifier with decision rule:

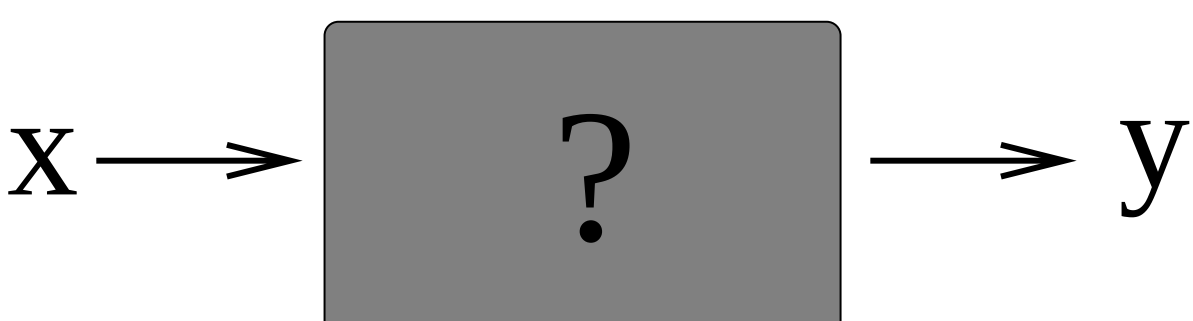
$$c(x) = sign\left[\sum_{i=1}^{m} \alpha_i K(x, w_i) \; - b\right]$$

how many and where should the templates w_i be and how should they be weighted by α_i ?



Motivation: Learning Theory

- Learning from examples an attractive alternative to direct design, many real world applications.
- Advantage: can make progress when systems or tasks are too complex to model explicitly. Disadvantage: cannot understand what the resulting "black box" is doing.
- We would like to provide *performance guarantees* (e.g. as in control theory) for two things:
  1. Data (Sample) Complexity:
     How many training examples are required for a network to generalize "well" ?
  2. Computational Complexity:
     How long it will take to train the network on a certain amount of data ?



The Learning Problem

- We are stuck with:
  1. Input Space $\mathcal{X}$ (say $\Re^n$)
  2. Binary Output Space $\mathcal{Y}$ (say $\{-1, 1\}$)
  3. Input Distribution $P_{\mathcal{X}}(x)$
  4. Target Conditional Distribution $F_{\mathcal{Y}}(y|x)$=Prob$[y = 1|x]$
- We are given a *training set* $(x_1, y_1), \ldots, (x_\ell, y_\ell)$ of $\mathbf{i.i.d}$ examples from $P(x, y) = P(x) \cdot F(y|x)$
- Goal: classify future inputs from $P_{\mathcal{X}}(x)$ with minimum expected future error.
- We propose *Function Learning*:
  1. Hypothesis Space $\mathcal{H}$: Set of binary functions $f_\alpha(x)$ parameterized by $\alpha \in \Lambda$
  2. Learning algorithm $LA : \{\mathcal{X} \times \mathcal{Y}\}^\ell \to \Lambda$ Takes the training set and produces a guess $f_\alpha(x)$ that we hope "best approximates" the target conditional distribution (Bayes)
- Goal: find α_{opt} that minimizes expected error: $\min_{\alpha \in \Lambda} E(\alpha) =$ prob. under $P(x, y)$ that $y \neq f_\alpha(x)$

Empirical Risk Minimization

- Q: Hopeless since we do not know $P(x,y)$?
  A: No, because we have $(x_1, y_1), \ldots, (x_\ell, y_\ell)$ that are drawn i.i.d from $P(x,y)$.
- *Empirical risk minimization* (ERM) method:

  Pick α_ℓ which minimizes the risk functional:

  $$\min_{\alpha \in \Lambda} \left\{ \hat{E}(\alpha) = \frac{1}{\ell} \sum_{i=1}^{\ell} [f_\alpha(x_i) \neq y_i] \right\}$$

  1. Use $f_{\alpha_\ell}(x)$ as our "best guess" function.
  2. This gives $E(\alpha_\ell)$ near $E(\alpha_{opt})$ for big ℓ (we hope).
- The method of ERM is not new:
  Least Squares is ERM for Regression loss fn.
  Maximum Likelihood is ERM for Density loss fn.
  Minimizing training error (e.g. in ANN) is ERM for Classification loss fn.
- Need to know: is picking α_ℓ the right thing ?

How can we make ERM work ?

- Problem: will it work ?
  If $\mathcal{H}$ is too "rich" then we fit noise/memorize and don't generalize well: $E(\alpha_\ell) >> E(\alpha_{opt})$.
  If $\mathcal{H}$ is too "poor" then it may not contain any good functions: even $E(\alpha_{opt})$ may be bad.
- Approach: assure ourselves somehow that for large enough ℓ, $\hat{E}(\alpha_\ell) \approx E(\alpha_\ell)$ and then take $\mathcal{H}$ rich enough so that $\hat{E}(\alpha_\ell)$ is satisfactorily small.
- Two issues:
  1. Asymptotic Correctness:
     Are we OK as $\ell \to \infty$?
  2. Convergence Rates:
     How big does ℓ need to be ?

Asymptotic Correctness

- Empirical Risk Minimization is *consistent* for hypothesis class $f_\alpha(x)\ \alpha \in \Lambda$, and input distribution $P(x, y)$ if:

  $E(\alpha_\ell) \rightarrow E(\alpha_{opt})$ as $\ell \rightarrow \infty$ AND
  $\hat{E}(\alpha_\ell) \rightarrow E(\alpha_{opt})$ as $\ell \rightarrow \infty$

  in other words, we guess the *right* function AND we get the *right* estimate of the true risk

- KEY THEOREM [Vapnik & Chervonenkis '89]: Consistency $\Leftrightarrow$ "Uniform convergence"

- "Uniform convergence" means that each $\hat{E}(\alpha)$ converges to the true $E(\alpha)$ uniformly over all $\alpha \in \Lambda$ as $\ell \rightarrow \infty$ formally,

  ERM is consistent for $R(\alpha)\ \alpha \in \Lambda, P(x, y)$ iff:

  $$\lim_{\ell \rightarrow \infty} P\left\{\sup_{\alpha \in \Lambda} |E(\alpha) - \hat{E}(\alpha)| > \epsilon\right\} = 0 \quad \forall \epsilon > 0$$

- Why key ? Tells us that when considering method of ERM only the *worst case* deviations matter. No *average case* theory of consistency is possible.

Why Uniform Convergence ?

- Consider any *single* function $f_\alpha(x)$. Its empirical risk $\hat{E}(\alpha)$ is a random variable which approaches its mean value the true risk $E(\alpha)$ as number of samples increases (Bernoulli).
- However: if all the members of the hypothesis class do not converge uniformly then we may be tricked into selecting one with a low empirical risk that actually has a poor true risk if it is a slower converger than others.
- We need to be able to say:
  "No matter which hypothesis we consider, with so many training examples its empirical risk is (almost) sure to be (very) close to its true risk."
  Then we can proceed to compare hypotheses without fear and we can assert that our training error will be a good reflection of test error.
- This is so called PAC Learning [Valiant '84] ("probably approximately correct") except for computational considerations.

Assuring Uniform Convergence

- If our hypothesis class has only a single element or even any finite number of elements in it then we can find conditions on when uniform convergence will occur and also bounds on the rate with respect to the number of samples in the training set based on the Law of Large Numbers from classical probability.
- But most often our hypothesis class is infinite. When can infinite classes be learned from a finite number of samples ?
- Vapnik and Chervonenkis answered this question for binary valued function classes in 1968 and for real valued functions in 1981.
- They demonstrated necessary and sufficient conditions in the form of an distribution dependent quantities known as the *VC Entropy* and *VC Annealed Entropy* of function classes.
- However, their most exciting results were their distribution *independent* sufficient conditions based on a quantity called the *VC Dimension* of a function class.

VC Dimension

- Vapnik and Chervonenkis ('68) proved the surprising and very powerful result:
  Consistency $\Leftrightarrow$ "Uniform convergence" $\Leftrightarrow$
  Class $\mathcal{H}$ has finite *VC Dimension* $D(\mathcal{H})$
- They also obtained the amazing bound ('71):

$$P\left\{\sup_{\alpha\in\Lambda} |E(\alpha) - \hat{E}(\alpha)| > \epsilon\right\} \leq 4e^{\left(-\epsilon^2\ell + D\ln(2\ell/D)+1\right)}$$

  independent of the input distribution $P(x, y)$!
- It has the nice equivalent form which says that:

$$|E(\alpha) - \hat{E}(\alpha)| \leq \sqrt{\frac{D\ln(2\ell/D) + \ln(4/\delta) + 1}{\ell}}$$

  with probability at least $(1-\delta)$ *simultaneously* for all functions $f_\alpha(x)$ $\alpha \in \Lambda$.
- The single scalar D above characterizes the complexity or capacity of the entire hypothesis class. It is called the *VC Dimension* of the function class. It is the logarithm of an important function called the *Growth Function* of the class.

Growth Function

- Consider a class $\mathcal{H}$ of binary valued functions $f_\alpha(x)$ parameterized by $\alpha \in \Lambda$ and a finite set of data samples $x_1, \ldots, x_\ell$.
- Construct the binary ℓ-vector

$$v_\alpha = [f_\alpha(x_1), \ldots, f_\alpha(x_\ell)]$$

- As the parameter α is moved throughout Λ, v_α changes, visiting various corners of the ℓ-dimensional hypercube. Count the number of corners it visited, call this number

$$N_{\mathcal{H}}(x_1, \ldots, x_\ell)$$

- $N_{\mathcal{H}}$ is the number of *dichotomies induced* by the function class $f_\alpha(x)$ $\alpha \in \Lambda$ on the dataset. Clearly, $N_{\mathcal{H}} \leq 2^\ell$.
- The maximum value of $N_{\mathcal{H}}$ over all possible datasets of size ℓ depends only on the function class and on ℓ. Call it $M_{\mathcal{H}}(\ell)$.

$$M_{\mathcal{H}}(\ell) = \max_{\mathcal{X}^\ell} N_{\mathcal{H}}(x_1, \ldots, x_\ell)$$

- This is the so-called *growth function* $M_{\mathcal{H}}(\ell)$ for the function class.

Properties of the Growth Function

- Cover ('65) noted this amazing fact: Either $M_{\mathcal{H}}(\ell)$ is identically equal to 2^ℓ or it is bounded above by $M_{\mathcal{H}} \leq \ell^D + 1$.
- If the growth function is equal to 2^ℓ then the function class has *infinite* VC dimension otherwise its VC dimension is D
- The growth function proceeds as 2^ℓ for $\ell \leq D$ and then begins to grow only polynomially.
- An *equivalent* definition of the VC dimension is the size of the largest dataset that can be found so that the function class *shatters* the dataset (induce all possible dichotomies).
- The VC dimension depends only on the function class and is not necessarily equal to the number of free parameters in α. It may be equal to, more than, or less than this number.
- The results can be generalized to strictly bounded and merely non-negative real valued functions.

Complexity Control for Small Sample Size

- The main VC result independent of the input distribution is the bound:
$$E(\alpha_\ell) \leq \hat{E}(\alpha_\ell) + \Phi\left(\frac{\ell}{d}, \frac{-\ln\delta}{\ell}\right)$$
which is true with probability at least $(1-\delta)$.
- If the sample size is large compared to the VC dimension then ℓ/d will be large and so the second term (confidence interval) will be small. If we can get a low value of empirical risk we can guarantee small actual risk.
- However is the sample size is small compared to the VC dimension (say $\ell/d < 20$) then the confidence interval will be large and the bound will be poor – our test error may not be near our training error.
- To get the best possible bound on the true risk we minimize over *both* terms in the bound simultaneously; this implies that we need to introduce the VC dimension as a *control variable*.

Optimal Hyperplanes

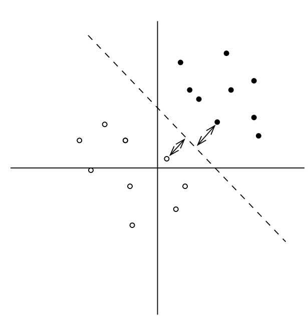
- Vapnik ('95) made the important observation that the VC dimension of a set of "canonical" hyperplanes with bounded weight vectors $\|w\| \leq A$ is:

$$\min\left\{n+1, R^2A^2+1\right\}$$

- The hyperplanes are defined on a space $\mathcal{X}$ which is bounded by a sphere of radius R.
- "Canonical" sets the scale factor so that:

$$\min_{x \in \mathcal{X}} |w \cdot x - b| = 1$$

- So the "optimal" (smallest bound on error) hyperplane is the one which separates the data and has the smallest VC dimension; namely the smallest weight vector.
- This is *equivalent* to picking the hyperplane that maximizes the distance to the nearest data point: the *maximal margin* hyperplane.



Finding an Optimal Hyperplane is a Quadratic Programming Problem

- The optimal canonical hyperplane with respect to some training data (x_i, y_i) is given by the solution to the constrained optimization:
$\min \frac{1}{2}\|w\|$ subject to $y_i(x_i \cdot w - b) \geq 1 \quad \forall i$
- By the Karush-Kuhn-Tucker theorem if we define:
$$L = \frac{1}{2}\|w\| - \sum_{i=1}^{\ell} \alpha_i \left[y_i(x_i \cdot w - b) - 1\right]$$
then minimize L with respect to w and b and maximize with respect to $\alpha_i \geq 0$ we will solve the original constrained optimization problem.
- Furthermore, the multipliers for all inactive constraint points are zero:
$$\alpha_i \left[y_i(x_i \cdot w - b) - 1\right] = 0 \quad \forall i$$
- Imposing $\partial L = 0$ conditions gives the (easy!) quadratic maximization:
$$L = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$
with respect to $\alpha_i \geq 0$ subject to $\sum_i \alpha_i y_i = 0$

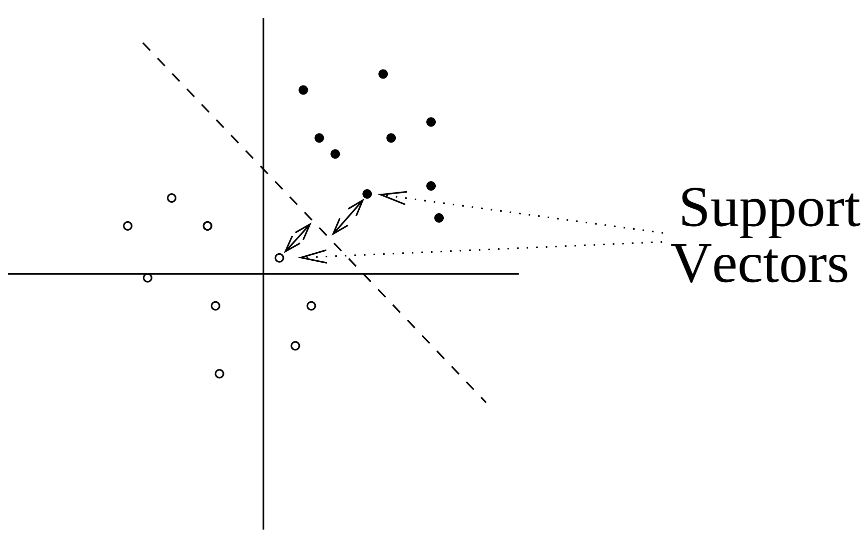
Support Vectors

- It will turn out (especially in high dimensions) that almost all the coefficients α_i will be zero. The few that are nonzero define which datapoints are important. These important points are called the "support vectors".
- The support vectors completely define the optimal hyperplane and threshold:

$$w^{opt} = \sum_i \alpha_i^{opt} y_i x_i = \sum_{supp.vect} \alpha_i^{opt} y_i x_i$$

$$b^{opt} = \frac{1}{2}[w^{opt} x_+ + w^{opt} x_-]$$

- In other words, the optimal weight vector is a linear combination of the points nearest to the dividing line between classes. Nobody else matters.



Support Vector Machines

- *Support Vector Machines* [Vapnik '92] implement *potential function* classifiers (weighted template matching) of the form:

$$c(x) = sign\left[\sum_{i=1}^{m} \alpha_i K(x, w_i) \; - b\right]$$

- The function $K(\cdot, \cdot)$ defines a "distance metric" between any two members of the input space. The vectors w_i are the templates to which we match and the α_i are mixing coefficients that control the weighting between templates.
- Hard problems to answer:
  1. How should we choose the "distance" $K(\cdot, \cdot)$?
  2. How should we pick the number (m) and locations (w_i) of the templates?
  3. How should we select mixing coefficients α_i?
  4. Even if we can answer (1),(2),(3) we still have a hard nonlinear optimization to do.

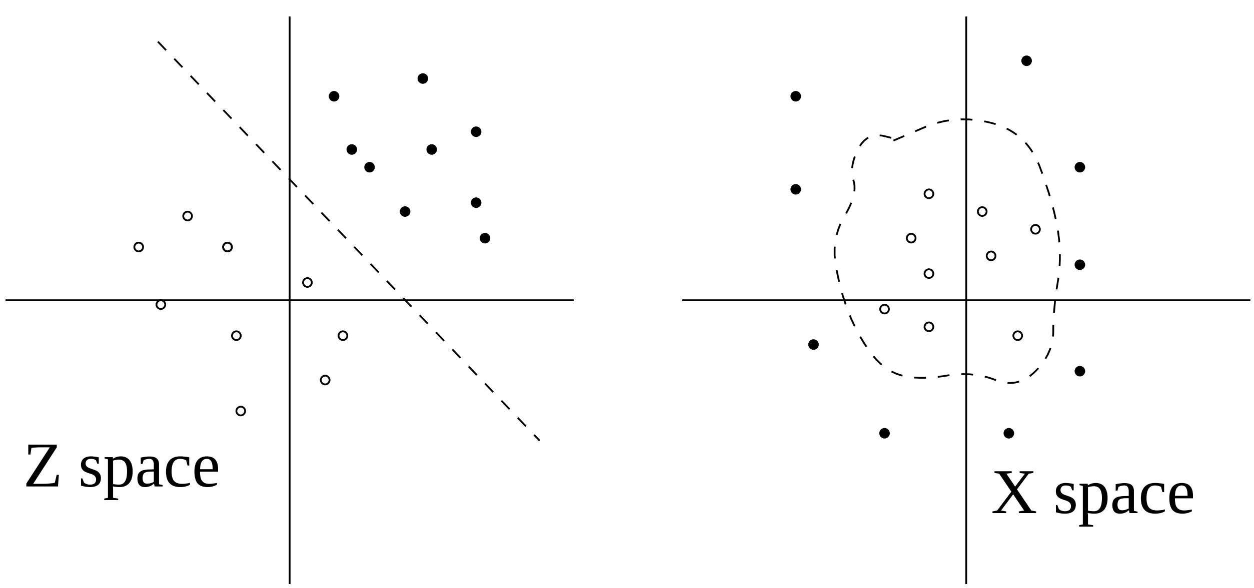
Support Vector Machines

- Under some mild restrictions (Mercer) Courant & Hilbert ('53) showed that every symmetric positive kernel $K(\cdot,\cdot)$ can be written in the form of a dot product in some space $\mathcal{Z}$:

$$K(x_i, x_j) = z_i \cdot z_j$$

  in new variables z for some mapping $z_i = \Psi(x_i)$

- This means that there is a "parallel" feature space (straightening space) $\mathcal{Z}$ in which our metric function is just an inner product.
- So our potential function classifier in $\mathcal{X}$ space is just a hyperplane classifier in $\mathcal{Z}$ space!!
  And we know how to do hyperplanes optimally.



Support Vector Machines

- Conceptually we could map all our original data from $\mathcal{X}$ space to $\mathcal{Z}$ space, find the optimal hyperplane, and map back.
- However, there are two problems:
  1. The $\mathcal{Z}$ space might be VERY big (high-d).
  2. The mapping function $\Psi(\cdot)$ is unknown.
- Luckily we can just interpret the optimal solution in $\mathcal{Z}$ space and formulate the problem directly in the original space. Maximize:

$$L = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

  with respect to $\alpha_i \geq 0$ subject to $\sum_i \alpha_i y_i = 0$
- Thus two surprising results appear for *any* nonlinear potential function classifier regardless of the metric function $K(\cdot, \cdot)$:
  1. Templates must lie on training set points.
  2. The mixing coefficients are given by the solution of a quadratic optimization.

Some Amazing Things

- Once we pick a metric function for our space, we can find the *optimal* potential function classifier, including number and location of templates and their weighting.
- All that is required is the solution to a common optimization for which fast polynomial time algorithms exist.
- There is an easy *on-line* method:
  1. Start with some initial data and find optimal classifier for it.
  2. For each new datapoint, classify it. If you get it right keep going.
  3. If you get it wrong, pool it with the existing support vectors and retrain the machine.
- Finally, Vapnik ('95) claims the astounding theorem to be true:

  $$\frac{\mathrm{avg.}}{\mathrm{P}^{\ell}(\mathrm{x},\mathrm{y})}\mathrm{Prob}(\mathrm{err}) = \frac{\mathrm{avg.}(\#\mathrm{supportvectors})}{(\ell-1)}$$

  independent of dimension of spaces, R or A !!

Extensions and Open Questions

- This can all be extended to cases where the training data cannot be perfectly separated.
- Support Vector Machines technically violate the original VC conditions because they pick the hypothesis class $\mathcal{H}$ on the basis of the training data.
- When they were proposed, they were merely a heuristic not well backed by theory. However, Shaw-Taylor and others ('95) have found bounds which apply even in this case.
- Nobody yet knows if the parameters which minimize these *bounds* on error actually minimize the true error.