Summer School.

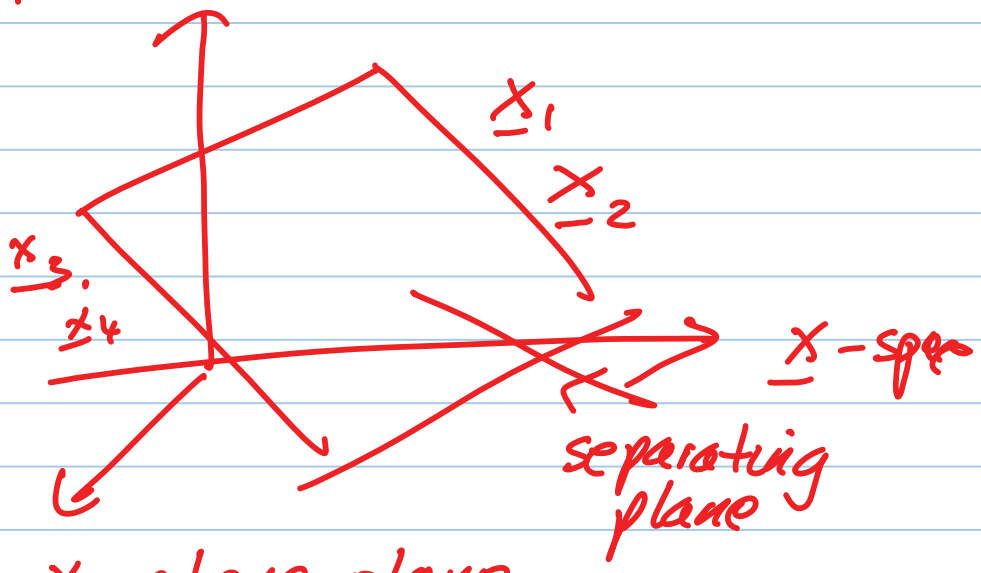
Note Title 7/15/2007

Data $\underline{x}$ in some parameter space

Classify as $y \in \{\pm 1\}$.

Specify a rule $\alpha(\underline{x}) \in \{\pm 1\}$

Example: seperation by hyperplane



Rule: If $\underline{x}$ above plane
label as $y = 1$
$\underline{x}$ below plane
label as $y = -1$

Geometrically define plane by $\underline{a} \cdot \underline{x} + b = 0$

Rule $\alpha(\underline{x}) = \pm 1$, if $\underline{a} \cdot \underline{x} + b \gtrless 0$.

How to determine the plane?

Train with labelled training examples

$(\underline{x}_1, y_1), (\underline{x}_2, y_2), \ldots, (\underline{x}_N, y_N)$.

Need an algorithm to find the "best" plane.

Perceptron Algorithm — guaranteed to converge to a plane that separates the positive ($y=+1$) and the negative ($y=-1$) examples (provided a plane exists).

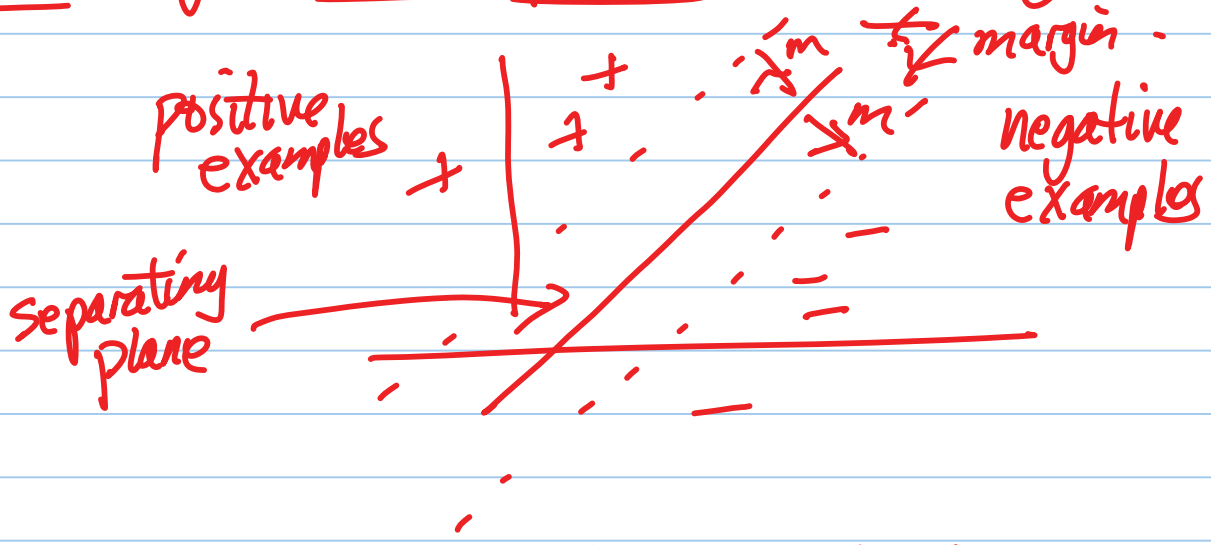
Generalization vs. Memorization.

Generalization: need this plane (rule) to successfully classify data that you havn't trained on (new test dataset).

Memorization: classifies training data perfectly, but fails to generalize to new data.

Want Generalization.

Idea of best plane → margin.



Try to find the plane with the biggest margin.

Intuitively, this will give the best chance of generalizing.

Mathematical theory justifies the intuition.

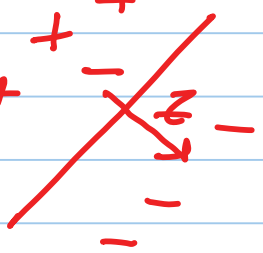
Some Mathematics

Constrained Optimization
Lagrange multipliers.

$$L_P(\underline{a}, b, z; \alpha, \tau) = \frac{1}{2}|\underline{a}|^2 + \gamma \sum_{i=1}^{N} z_i - \sum_{i=1}^{N} \alpha_i \{ y_i (\underline{x}_i \cdot \underline{a} + b) - (1 - z_i) \} - \sum_{i=1}^{N} \tau_i z_i .$$

Minimize L_P wrt. $\underline{a}, b, z$
maximize wrt. α, τ.

$\underline{a}, b$ specifies the plane
$|\underline{a}|$ specifies the inverse margin.
z_i enables training points to be misclassified – but pay a penalty



Intuition: Find the plane with biggest margin that moves points by a minimum amount.

Algorithms exist to minimize L_p. and obtain the "best" plane.

Result: the solution is of form.

$$\hat{\underline{a}} = \sum_{i=1}^{N} \alpha_i \underline{x}_i y_i$$

where $\alpha_i = 0$, unless $\underline{x}_i$ is on the margin (after z_i).

Hence $\hat{\underline{a}}$ depends only on the support vectors

→ i.e. only on the data near the separating boundary (ignores data away from the boundary)

Solution: $\alpha(\underline{x}) = \text{sign}(\hat{\underline{a}} \cdot \underline{x} + \hat{b})$

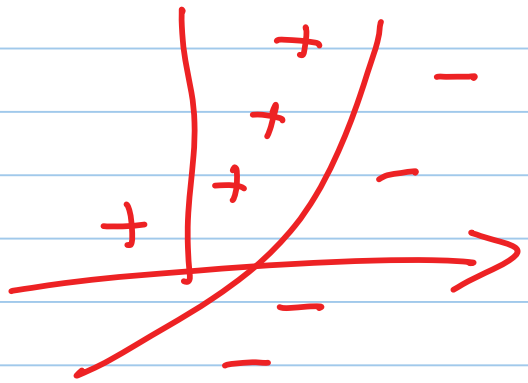
Planes, Margin, Support Vectors.

$$\alpha(\underline{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i \underline{x}_i \cdot \underline{x} + \hat{b}\right) \quad //$$

Kernel Trick:

What if we don't want to use planes?

The kernel trick is a very simple way to greatly extend this method.



Send $\underline{x} \to \underline{\varphi}(\underline{x})$, $\underline{\varphi}(.)$ arbitrary feature.

Solution depends only on quantities like $\underline{\varphi}(\underline{x}) \cdot \underline{\varphi}(\underline{x}') = K(\underline{x}, \underline{x}')$

Definition of kernel.

$$\alpha(\underline{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i K(\underline{x}, \underline{x}_i) + b \right).$$

(Different α).

Support Vector Machine

Risk & Empirical Risk.

Risk $R(\alpha) = \sum_{x,y} P(x,y)\, L(y, \alpha(x))$

L: loss function.

Empirical Risk $R_{emp}(\alpha) = \sum_{i=1}^{N} L(y_i, \alpha(x_i))$.

In the limit as $N \to \infty$ $R_{emp}(\alpha) \to R(\alpha)$

Technical Conditions

Discriminative approaches (e.g. SVM)

minimize $R_{emp}(\alpha)$ directly to get the decision rule $\hat{\alpha}$.

Bayesian approaches use the data $\{(x_i, y_i)\}$ to learn the distribution

$$P(x,y) = P(x|y)P(y)$$

then finds α to minimize the risk.

AdaBoost.

Learn a classifier from a set of weak classifiers $\{\varphi_i(x)\}$

$\varphi_i(x) \in \{\pm 1\}$.

Weak classifiers are correct $> 50\%$ time

Build a strong classifier:

$$H(x) = \text{sign} \sum_{\mu=1}^{n} \lambda_\mu \varphi_\mu(x)$$

Algorithm → can be expressed as greedy steepest descent.

Define $Z[\lambda_1, \ldots, \lambda_n] = \sum_{i=1}^{N} e^{-y_i \sum_{\mu=1}^{n} \lambda_\mu \varphi_\mu(x)}$

Initialize $\lambda_i = 0, \forall i$.

Time step t: solve $\frac{\partial Z}{\partial \lambda_i} = 0$, for each λ_i (other λ's fixed)

select i to maximally decrease $Z[\,]$

update λ_i.

Adaboost (Cont).

You are selecting the choice of weak classifier to use — and its weight.

Generalization — verses Memorization

Need to keep a training set and a test set. Train on training set, evaluate on test set & training set.

If results on training set are better than results on test set — then you have overgeneralized.

Vapnik's Results

bound generalization error in terms of training error + VC dimension.

Nice Mathematically — practical use?

Learning the Posterior.

Adaboost was formulated in terms of classification.

It can be reformulated in terms of estimating the conditional distribution

$p(y|x)$.

Why does this matter?

Bayes says learn $p(x|y)$ generative model and $p(y)$ prior.

Then perform inference to maximize

$$p(y|x) = \frac{p(x|y)\,p(y)}{p(x)}.$$

Why not learn $p(y|x)$ directly? (discriminative model)

Some forms of machine learning attempt to directly learn the posterior distribution $p(y|x)$ directly.

This posterior distribution can become complex — e.g. y can have multiple states — and the distribution can have hidden variables.

These types of models become very similar to generative Bayesian models.

They can be extremely useful in practice — when it is hard to specify a generative model.

Summary

1. Classification
2. Support Vector Machine
   hyperplanes, margin, support vector, kernel trick.
3. Generalization & Memorization.
4. Vapnik's Bounds & VC dimension.
5. AdaBoost
6. AdaBoost to learn the posterior.
7. Risk & Empirical Risk.
8. Machine learning to learn posterior distribution — multiple states & hidden variables.