Data $x$ in some parameter space classify as $y \in \langle \pm 1 \rangle$.

Specify a rule $\phi(x) \in \langle \pm 1 \rangle$

Example: separation by hyperplane.

$\phi(x)$ above plane label as $y = 1$

$\phi(x)$ below plane label as $y = -1$

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

Rule $\phi(x) = \pm 1$, if $a \cdot x + b \geq 0$. 
How to determine the plane?

Train with labelled training examples 
\((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\).

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 haven't trained on (new test dataset).

Memorization: classifies training data perfectly, but fails to generalize to new data. Want Generalization.
Idea of best plane $\rightarrow$ margin.

Separating plane

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.

\[ L_p(a, b, z; x, y) = \frac{1}{2} a^2 + \frac{\sum_{i=1}^{n} z_i}{2} \]

Minimize \( L_p \) \( \text{wrt.} \) \( a, b, z \)
Maximize \( \text{wrt.} \) \( x, y \).

\( a, b \) specifies the plane
\( |a| \) specifies the inverse margin.
\( z_i \) enables training points to be
misclassified — but pay a penalty +
Intuitively, find the plane
with biggest margin that moves
points by a minimum amount.
Algorithms exist to minimize Lp.
and obtain the "best" plane.

Result: the solution is of form:
\[ \hat{a} = \frac{1}{n} \sum_{i=1}^{n} d_i x_i y_i \]
where \( d_i = 0 \), unless
\( x_i \) is on the margin (after \( z_i \)).

Here \( \hat{a} \) depends only on the support vectors
\( \Rightarrow \) i.e. only on the data near the separating
boundaries (ignores data away from the boundary).

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

Planes, Margin, Support Vectors.
\[ \alpha(x) = \text{sign}(\sum_{i=1}^{n} \alpha_i x_i \cdot x + \hat{b}) \]
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 $x \rightarrow \phi(x)$, $\phi(.)$ arbitrary feature.

Solution depends only on quantities like $\phi(x)$, $\phi(x') = K(x, x')$ defined $\in$ kernel.

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

Support Vector Machine
Risk 2 Empirical Risk.

\[ R(x) = \sum_{x,y} P(x,y) L(y, d(x)) \]

Empirical Risk

\[ \text{Emp} \rho(x) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, d(x_i)) \]

\[ \text{In the limit as } N \to \infty \quad \text{Emp} \rho(x) \to R(x) \]

Technical Condition

Discriminative approaches (e.g., SVM)

minimize \( \text{Emp} \rho(x) \) directly to get the decision rule \( \hat{d} \).

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

\[ P(s, y) = P(x | y) P(y) \]

then finds \( \hat{d} \) 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:

\[ H(x) = \text{sign} \sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x) \]

Algorithm \( \rightarrow \) can be expressed as greedy steepest descent.

Define \( \mathcal{Z} = \prod_{i=1}^{n} \mathcal{J} = \prod_{i=1}^{n} e^{-y_i \sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x)} \)

Initialize \( \lambda_i = 0, \forall i \).

Time step \( t \):

\[ \frac{\partial \mathcal{J}}{\partial \lambda_i} \]

Select \( i \) to maximally decrease \( \mathcal{Z} \mathcal{J} \)

Update \( \lambda_i \).
**AdaBoost (Cont.).**

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

**Generalization — versus Memorization**

Need to keep a training set and a test set. Train on training set, evaluate on test set & tuning set. If results on training set are better than results on test set — then you have overgeneralized.

**Vapnik’s Results**

Bound generalization error in term 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

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

Why not learn $p(y|x)$ directly? (discriminative model)
Some forms of machine learning attempt to directly learn the posterior distribution $p(\theta|x)$ 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 vectors, kernel trick.
4. Vapnik's Bounds & VC dimension.
5. AdaBoost
6. AdaBoost to learn the posterior.