Data $x$ in some parameter space classify as $y \in \{\pm 1\}$.

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

Example: separation by hyperplane.

Rule: $x$ above plane label as $y = 1$
$x$ below plane label as $y = -1$

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

Rule $\phi(x) = \pm 1$, $\ell \geq a \cdot x + b \geq 0$. 
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.

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

Lagrangian multipliers.

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

\[ - \sum_{i=1}^{N} z_i \left( y_i (x_i a + b) - (1 - z_i) \right) - \frac{\gamma}{2} \sum_{i=1}^{N} z_i z_i. \]

Minimize \( L_p \) w.r.t. \( a, b, z \)

maximize w.r.t. \( a, i \).

\( a, b \) 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

\[ \hat{\alpha} = \sum_{i=1}^{N} \alpha_i x_i y_i \]

where \( \alpha_i = 0 \), unless \( x_i \) is on the margin (after \( z_i \)).

Here \( \hat{\alpha} \) depends only on the support vectors → i.e., only on the data near the separating bounding (ignores data away from the boundary).

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

Planes, Margin, Support Vectors.

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

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

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

Support Vector Machine
Risk & Empirical Risk.

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

\[ \text{Empirical Risk} \]

\[ \text{Empirical Risk} = \sum_{c=1}^{C} L(y_i, d(x_i)) \]

In the limit as \( N \to \infty \)

\[ \text{Empirical Risk} \to R(x) \]

Technical Constraint

Discriminative approaches (e.g., SVM)

minimize \( \text{Empirical Risk} \) directly to get the decision rule \( \hat{d} \).

Bayesian approaches use the data \((x_i, y_i)\)

to learn the distribution

\[ P(x, y) = P(x \mid y) P(y) \]

then finds \( \hat{d} \) to minimize the risk.
AdaBoost.

Learn a classifier from a set of weak classifiers \( \{ \phi_i(x) \} \)
\[ \phi_i(x) \in \{ \pm 1 \} \]

Weak classifiers are correct > 50% time

Build a strong classifier:
\[ H(x) = \text{sign} \sum_{\mu=1}^{\Theta} \lambda_{\mu} \phi_{\mu}(x) \]

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

Define \( \mathcal{Z} = \prod_{r=1}^{n} J \)
\[ \frac{\partial}{\partial \lambda_i} \mathcal{Z} = \sum_{i=1}^{n} e^{-y_i \sum_{\mu=1}^{\Theta} \lambda_{\mu} \phi_{\mu}(x)} \]
Initialize \( \lambda_i = 0, \forall i \).

Time step \( t \): solve \( \frac{\partial}{\partial \lambda_i} \mathcal{Z} = 0 \), for each \( \lambda_i \) (other \( \lambda \)'s fixed).

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

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 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.
4. Vapnik's Bounds & VC dimension.
5. AdaBoost
6. AdaBoost to learn the posterior.