

# Machine Learning in Speech and Language Processing

Jeff A. Bilmes

Patrick Haffner

Tutorial presented on March 19, 2005

ICASSP'05, PHILADELPHIA

*2<sup>nd</sup>* Half: Kernel and Margin Classifiers

Presented by Patrick Haffner

## 2<sup>nd</sup> Half: Kernel and Margin Classifiers

- Good classifiers for speech and language applications?
- Huge range of approaches: focus on *recent advances*.
- ☞ Stronger emphasis on *natural language* applications:  
received a lot of recent attention from the Machine Learning community.
- ☞ Applications such as Neural Networks for Speech processing:  
Considered as an established technique■
- Key challenges:
  - ☞ Beat the curse of dimensionality: good *generalization* to test data.
  - ☞ Classifying structured objects.
  - ☞ Scaling learning to very large corpora.

# 2<sup>nd</sup> Half: Outline

▣ Statistical learning theory

▣ Kernel Classifiers

☞ SVMs: Maximize the margin

☞ Making kernels, kernels for Structured Data

☞ String and Weighted Automata Kernels

▣ Margin Classifiers

☞ Margin-based loss functions

☞ Generalize the concept of margin to a large class of classifier, including Boosting and Maxent.

☞ Large Scale Experiments

▣ Software and References

# Binary Classifiers

Using training data  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$ .

- Associate output  $y$  to input  $\mathbf{x} \in \mathcal{X}$
- Most of our presentation assume *binary classification*:  
 $y \in \{-1, 1\}$  or  $\{0, 1\}$ .

The following generalizations are usually possible:

- **Probability estimator**  $y \in [0, 1]$ .
- **Regression**  $y \in \mathbb{R}$ .
- **Multi-category classification**  $y \in \{1, \dots, l\}$ .
- **Output codes**  $y \in \{-1, 1\}^l$ .
- **Graphical Models**  $\mathbf{x}$  is the set of variables  $y$  depends upon.

# Statistical learning theory

## ► Purpose:

- ☞ Techniques to predict the generalization error.

- ☞ Methods to minimize this error.■

## ► Model: we observe data generated by an unknown IID (Independent Identically Distributed) distribution.■

## ► Analysis of the learning problem

- ☞ What functions can the machine implement?

- ☞ Capacity of this class of function?

# Empirical risk minimization

Training examples  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$  drawn from  $P(\mathbf{x}, y)$ .

▮ Learn  $f : \mathcal{X} \rightarrow \{-1, +1\}$ .

▮ Minimize the *expected risk* on a test set drawn from the *same*  $P(\mathbf{x}, y)$ .

$$R_{exp}[f] = \int \mathcal{L}(f(\mathbf{x}), y) dP(\mathbf{x}, y) \text{ where } \mathcal{L}(f(\mathbf{x}), y) = 2|f(\mathbf{x}) - y|$$

**Problem:**

☞  $P$  is unknown

☞ Need an *induction* principle

Find  $f \in \mathcal{F}$  minimizing the *Empirical Risk*:

$$R_m[f] = \frac{1}{m} \sum_m \mathcal{L}(f(\mathbf{x}_i), y_i)$$

# Uniform convergence bounds

Law of large numbers:  $R_m[f] \rightarrow R_{exp}[f]$  when  $m \rightarrow \infty$ .

But convergence must be *Uniform* over all functions  $f \in \mathcal{F}$ .

$$\lim_{m \rightarrow \infty} P \left( \sup_{f \in \mathcal{F}} (R_{exp}[f] - R_m[f]) > \epsilon \right) = 0 \text{ for all } \epsilon > 0$$

Bounds to guarantee this convergence established by

Vapnik (Vapnik, 1998) and others.

They typically establish that with probability  $1 - \delta$

$$R_{exp}[f] \leq R_m[f] + \frac{\tilde{O}(h)}{m^\alpha}$$

$h$  is a measure of *representation power* of the function class  $\mathcal{F}$

Example: VC dimension.

# Putting bounds into practice

▣▣▣▣ *Worst case* bounds are over-pessimistic:

☞ Huge number of examples required to guarantee good generalization■

▣▣▣▣ Most bounds are dimension free:

☞ they do not depend on the dimension of the sample space.■

▣▣▣▣ While they cannot be used for accurate predictions

☞ They suggest how to constrain the class of function to beat the curse of dimensionality■

▣▣▣▣ First success story: Support Vector Machines and Kernel methods.

# 2<sup>nd</sup> Half: Kernel Classifiers

▣ Statistical learning theory

▣ *Kernel Classifiers*

☞ *SVMs: Maximize the margin*

☞ *Making kernels, kernels for Structured Data*

☞ *String and Weighted Automata Kernels*

▣ Margin Classifiers

☞ Margin-based loss functions

☞ Generalize the concept of margin to a large class of classifier, including Boosting and Maxent.

☞ Large Scale Experiments

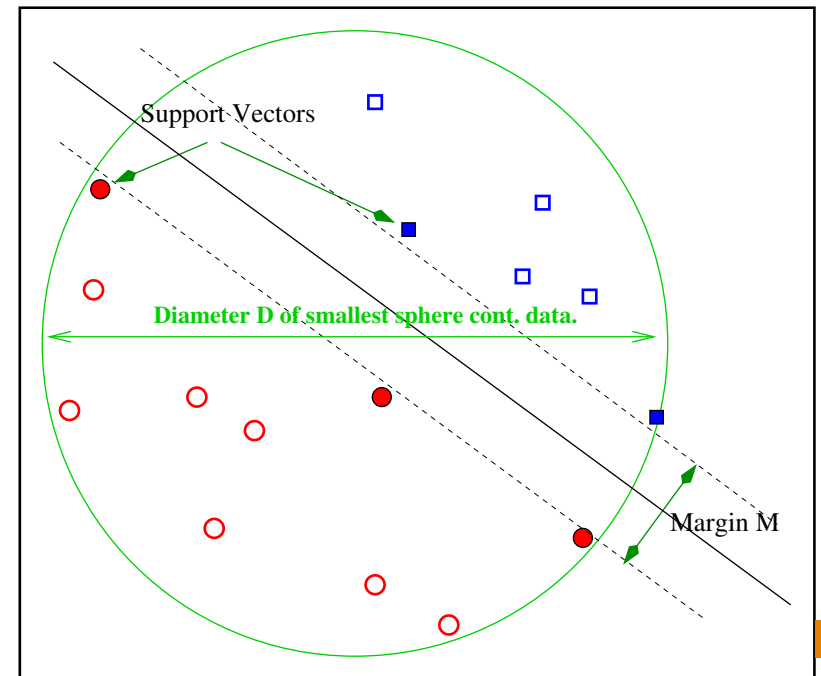
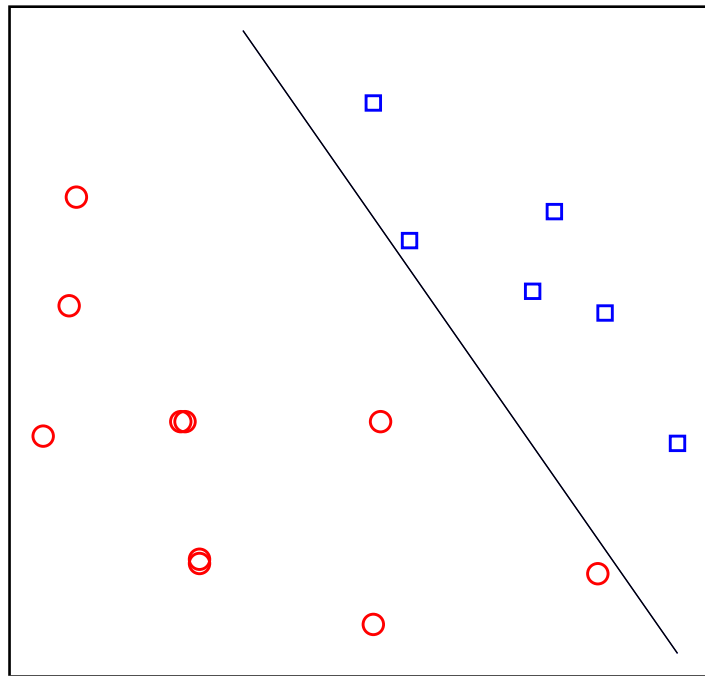
▣ Software and References

# Large-Margin Classifiers, Support Vector Machines

Classifiers of type  $f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$ .

Bound  $R_{exp}[f] \leq R_m[f] + \frac{1}{m} \tilde{O}\left(\frac{D^2}{M^2}\right)$

Separate classes with the **maximum margin**  $M$ :



*Soft margin* to allow training errors: described later.

# Finding the maximum margin separation

➡ Find weight vector  $\mathbf{w}$  that

☞ maximizes the margin  $\rho$

☞ with constraints  $\|\mathbf{w}\| = 1$  and  $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq M$ .■

➡ Laplace optimization with choice of parameters:

☞ **Primal** Coefficients of the weight vector  $\mathbf{w}$ .■

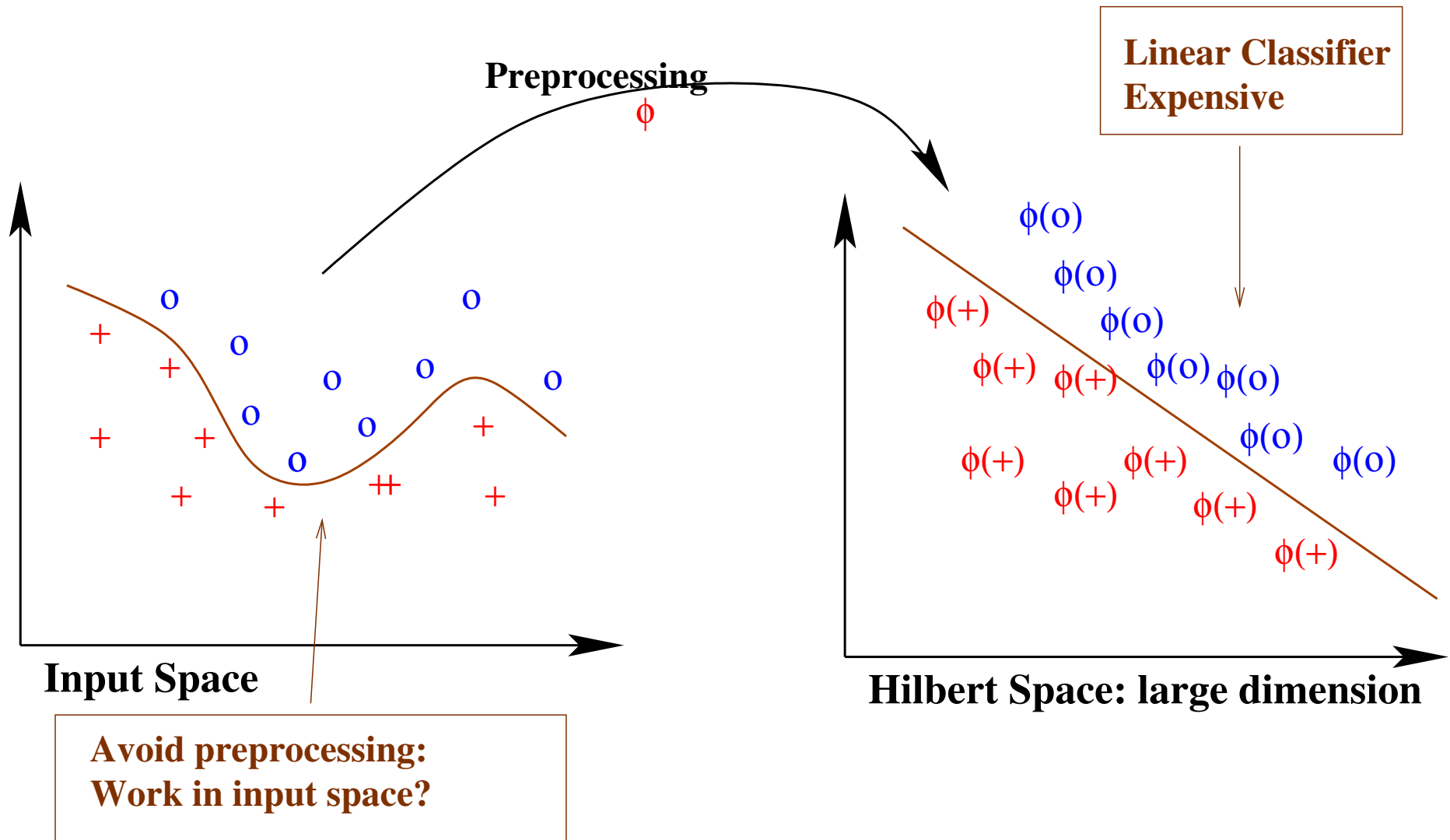
☞ **Dual** Coefficient of combination of training vectors

$$\mathbf{w} = \sum_{i=1}^N \alpha_i \mathbf{x}_i \text{ where } \alpha_i \neq 0 \text{ are } \textit{Support Vectors}.$$

➡ With dual representation, all computations use inner products with support vectors only:  $f(\mathbf{x}) = \sum_{i=1}^N \alpha_i (\mathbf{x}_i \cdot \mathbf{x}) + b$

# Non-Linear SVMs

Mapping from input space  $\mathcal{X}$  to a higher-dimensional *Hilbert* space.



# The Kernel Trick

## ► Idea

- ➡ Mapping  $\Phi$  from *input space*  $X$  to high-dimensional *Hilbert space*  $F$ :  $\mathbf{x} \rightarrow \Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_d(\mathbf{x}))$ .
- ➡ Define *kernel*  $K$  such that:  $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) = K(\mathbf{x}, \mathbf{z})$ . ■

## ► Benefits

- ➡ No need to define or compute  $\Phi$  explicitly.
- ➡ Efficiency:  $K$  may be much more efficient to compute than  $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z})$  when Hilbert dimension is large.
- ➡ Flexibility:  $K$  can be chosen arbitrarily, so long as the existence of  $\Phi$  is guaranteed.
- ➡ Condition: **Positive Definite Symmetric (PDS) Kernels**

# Positive Definite Symmetric (PDS) Kernels

Define Kernel Matrix  $K$

$$K_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$$

$K$  must be *symmetric* with *positive eigenvalues*.

$$\begin{array}{cccc} K_{1,1} & K_{1,2} & \dots & K_{1,m} \\ K_{2,1} & K_{2,2} & \dots & K_{2,m} \\ \dots & \dots & \dots & \dots \\ K_{m,1} & K_{m,2} & \dots & K_{m,m} \end{array}$$

Property must be true for all  $m \geq 1$ ,  $\{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subseteq \mathcal{X}$  and also writes:

$$\forall \{c_1, \dots, c_m\} \subseteq \mathbb{R} : \sum_{i,j=1}^m c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

# Example with Polynomial Kernels

Imagine a linear classifier over all pairs of input features:

▣▣▣▣  $d(d + 1)/2$  features.

▣▣▣▣ Dot product in Hilbert space:  $O(d^2)$  operations.■

$$K(\mathbf{x}, \mathbf{z}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) \text{ with } \Phi(\mathbf{x}) = (x_i x_j; 1 \leq i, j \leq d)$$

$$\blacksquare = \sum_{1 \leq i, j \leq d} (x_i x_j)(z_i z_j)$$

$$\blacksquare = \left( \sum_i x_i z_i \right)^2 = (\mathbf{x} \cdot \mathbf{z})^2$$

▣▣▣▣ Kernel in input space:  $O(d)$  operations.

▣▣▣▣ Generalize to any degree polynomial.

▣▣▣▣ Control over number of features in Hilbert space: *capacity*.

# Polynomial Kernel Success Story: *MNIST*

## Optical Character Recognition

- Reference dataset.
- 60,000 training samples
- 28x28 images: 784 pixels
- 9th degree polynomial kernel:  
Use all  $3 \times 10^{20}$  pixel 9-uple as features
- SVMs beat all other methods



| Kernel   | Error |
|----------|-------|
| Linear   | 8%    |
| Poly N=9 | 0.6%  |

# Making kernels from distances

We want to imagine what kernels do

- ▣▣▣▣ PDS kernels can be interpreted as measures of similarity, but...
- ▣▣▣▣ Distances are much easier to visualize.■

We have tools to turn distances into kernels provided that they are ...

- ▣▣▣▣ Symmetric and *negative definite*.

For all  $\mathbf{x}_1, \dots, \mathbf{x}_n$  and  $c_1, \dots, c_n \in \mathbb{R}^n$  satisfying  $\sum_i c_i = 0$

$$\sum_i \sum_j c_i c_j K_d(\mathbf{x}_i, \mathbf{x}_j) \leq 0$$

■

- ▣▣▣▣ Key property:  $\exp(-tK_d)$  positive definite for all  $t > 0$ .

# Distance Kernels for Distributions

**Gaussian** kernel based on *Euclidean* distance is standard:

$$K(\mathbf{x}, \mathbf{z}) = \exp -\rho(\mathbf{x} - \mathbf{z})^2$$

But is it the best distance for a distribution?

▣▣▣▣ Input vectors  $\mathbf{x}$  represent distributions:  $\sum_i x_i = 1$ .

▣▣▣▣ Measure of divergence for Distribution: Kullback-Liebler. ▣

Not a distance, but can be approximated by:

☞ The Hellinger/Bhattacharyya distance  $d = \sum_i |\sqrt{x_i} - \sqrt{z_i}|^2$ .

☞ the L1 distance  $d = \sum_i |x_i - z_i|$  (*Laplacian* Kernel) ▣

These distances are *Negative Definite Symmetric*:

☞  $\rightarrow K = e^{-\rho d}$  is a valid kernel.

# Distance Kernel Success Story: *WebKB*

- ➡ Home pages gathered from university computer science departments
- ➡ Classify them as *student* or *faculty*.■

Features: *Bag of Words*

- ➡ Put all the words from a web page in a bag.
- ➡ Distribution:  $P(w)$  is the probability to take word  $w$  out of the bag.■

*Good distances can be turned into good kernels*

| Kernel    | Error |
|-----------|-------|
| Gaussian  | 10%   |
| Laplacian | 7%    |
| Hellinger | 6.5%  |

# Complexity of runtime implementation

➡ Classifier with set of support vectors  $SV$ .

➡ Classify example  $\mathbf{x}$

☞  $f(\mathbf{x}) = \sum_{i \in SV} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$

☞ Complexity  $|SV| \times T_K$ , where  $T_K$  is the time to compute the kernel.

➡ Suppose Kernel is linear  $K(\mathbf{x}_i, \mathbf{x}) = \mathbf{x}_i \cdot \mathbf{x}$ :

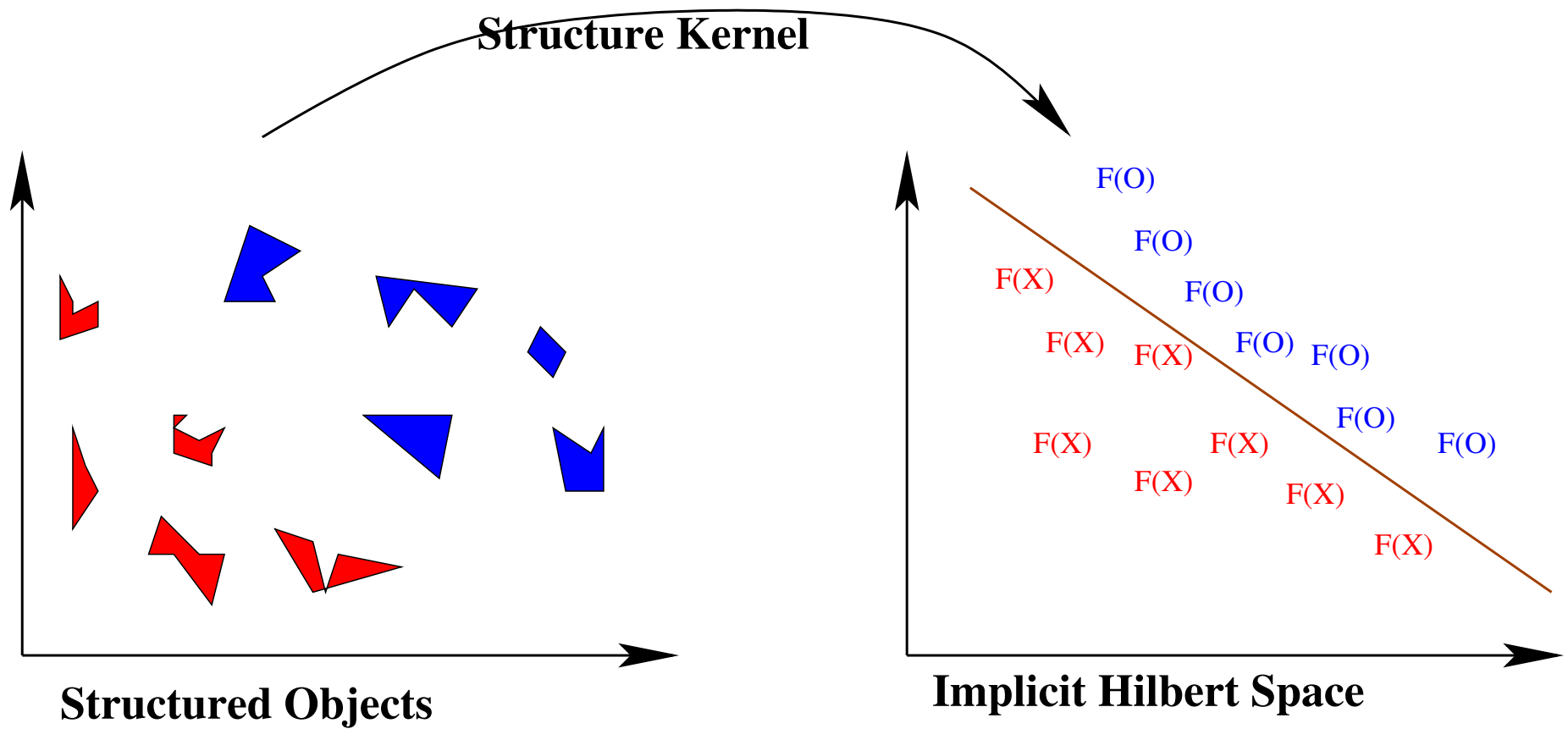
☞  $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$  with  $\mathbf{w} = \sum_{i \in SV} \alpha_i \mathbf{x}_i$ .

☞ Complexity: number of non-zero features in  $\mathbf{x}$ .

# Classifying complex objects

- ▣▣▣▣ Classification algorithms on *flat* fixed-size input vectors.
- ▣▣▣▣ However, the world is more complex in **areas** such as:
  - ☞ Spoken-dialog classification (strings, graphs)
  - ☞ Text Categorization (strings)
  - ☞ Parsing/reranking (trees)
  - ☞ Part-of-speech tagging, named entity extraction (strings, trees)
- ▣▣▣▣ Traditional method: feature extraction.
- ▣▣▣▣ Kernels are a measure of similarity between objects, or a distance that:
  - ☞ Avoid explicit computation a large-dimension feature space.
  - ☞ By factorizing these computations.

# How to project objects into a Hilbert Space?



# Kernels for structured data

- ▣ **Model Driven:** knowledge about objects encoded in models or relationships.
  - ☞ *Fisher* Kernels: built from a generative model or a *graphical model*.
  - ☞ *Diffusion* Kernels: local relationships encoded in a graph.■
- ▣ **Syntax Driven:** objects generated by syntactic rules.
  - ☞ Trees
  - ☞ Variable-length sequences.
  - ☞ Distributions of variable-length sequences.

# Tree Kernels

**Subtree tree kernels:**(Viswanathan and Smola, 2002)

☞ Encode tree as a string by traversing it

☞  $s = (A(B(C)(D))(E))$

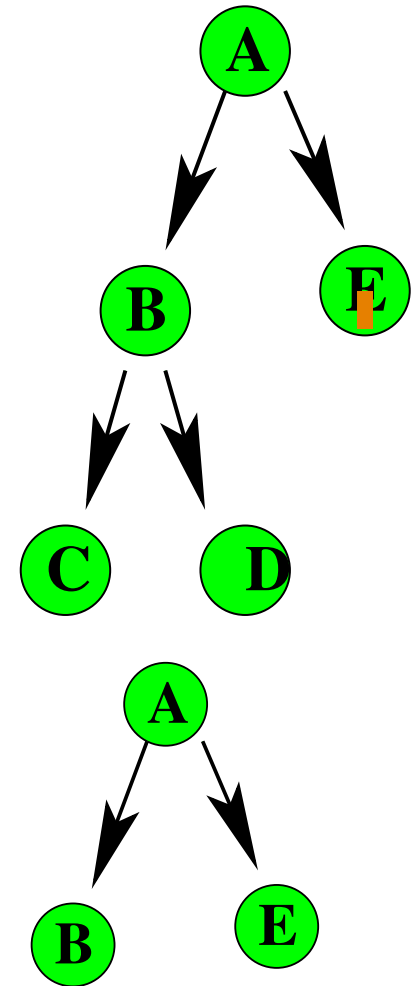
☞ Use kernel for strings..

**Subset tree kernels:**(Collins and Duffy, 2002)

☞ More general: match any subset of connected nodes.

☞ Recursive construction

Applications: reranking of parsing trees, shallow semantic parsing.



Subset, not subtree

# Kernels for strings: a general framework

## What features can be extracted from strings?

▣▣▣▣➤ **Bag of Words** feature sets:

**text categorization** (*Reuters*) (Joachims, 1998)

▣▣▣▣➤ Grammar Fragments or **N-grams**:

**spoken dialog classification** (AT&T *How May I Help You*).

▣▣▣▣➤ Variable length N-grams in **computational biology**.■

Kernels based on matching subsequences? Must generalize to:

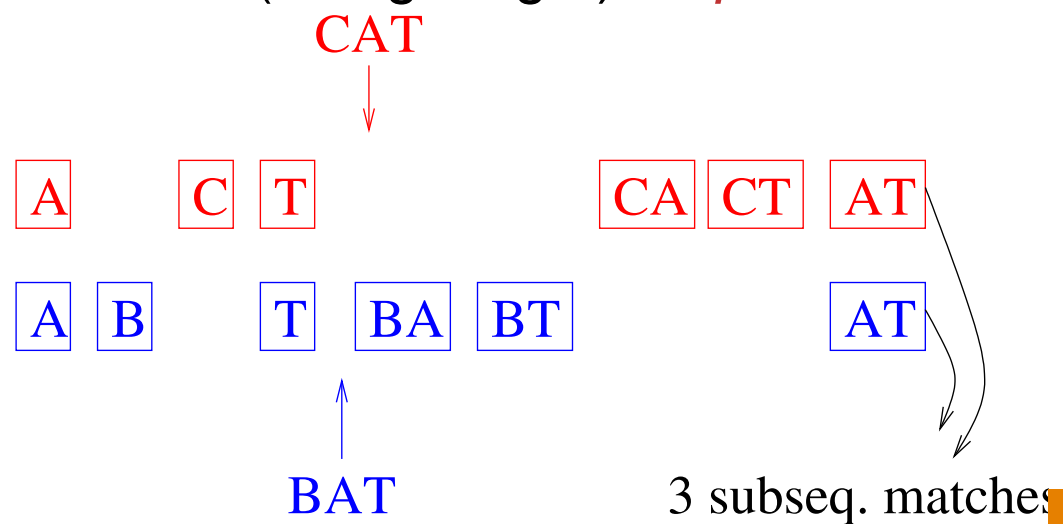
▣▣▣▣➤ sequences with gaps or mismatches

▣▣▣▣➤ distributions over sequences

# String Kernels

Find similarities between *CAT* and *BAT* strings.  
Explicit extraction of all subsequences:

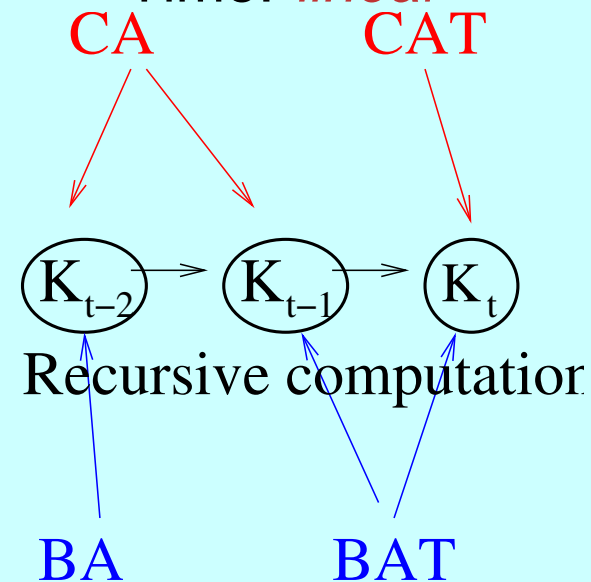
👉 Time(string length): *exponential*



Possible generalization with penalized gaps and mismatches.■

Equivalent Recursive computation:

👉 Time: *linear*



# Examples of String Kernels

## ▣▣▣▣▶ **Convolution kernels** (Haussler, 1999)

☞ Syntax-driven kernels constructed with recursive convolutions.

## ▣▣▣▣▶ **String Kernels** for text mining applications (Lodhi et al., 2001)

☞ Count common substrings (using recursion).

## ▣▣▣▣▶ **Mismatch Kernels** in computational biology (Leslie et al., 2003)

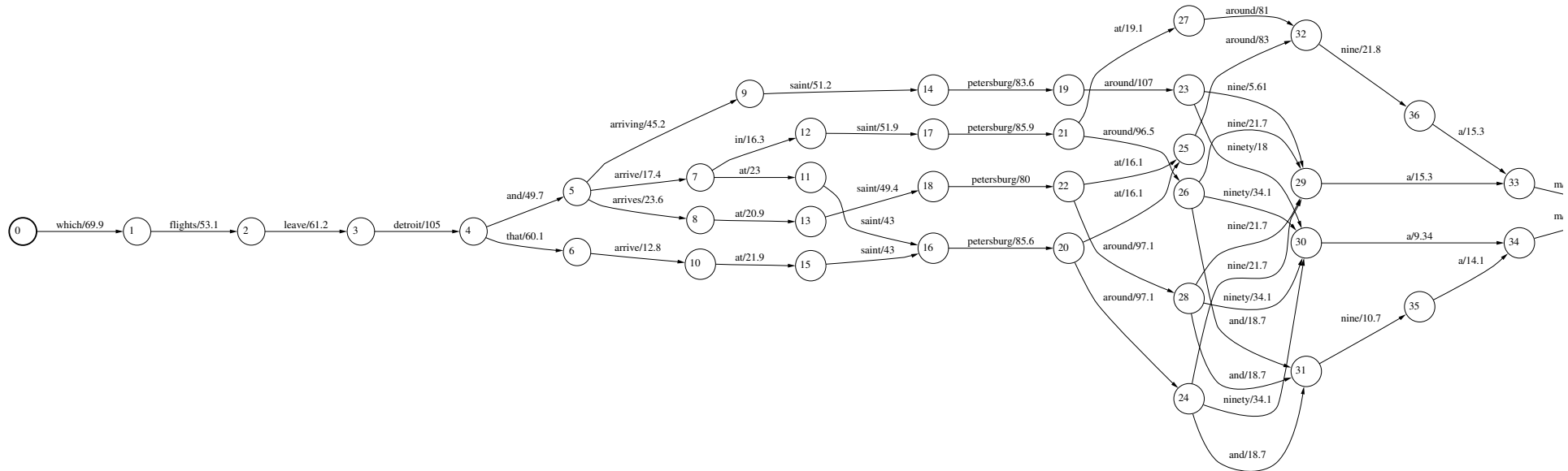
☞ Allows mismatches between common subsequences.■

## ▣▣▣▣▶ **Lattice Kernels** for spoken language understanding

☞ Lattice: weighted alternative of sequences represented as a Finite State Automata (Cortes, Haffner, and Mohri, 2004).

☞ Average count of common word N-grams in lattice.

# A structured object from Speech recognition



- Output of Speech Recognition Systems : *word or phone lattice*.
- Very large number of paths, or alternative sequences.
- Represented as *Weighted Finite State Machines (FSM)*.
- Weights used to rank alternative sequences.

# A generic generalization of String kernels

- ▣ Each string kernel:
  - ☞ Specific, usually recursive, implementation
  - ☞ Does not generalize to FSMs
- ▣ Rational kernels: General framework covering all these kernels:
  - ☞ Weighted FSM: represent both object and kernel.
  - ☞ Key operation: composition of two FSMs.
  - ☞ Factorization: distributivity property of the semiring used in FSMs.
- ▣ **FSM Software libraries available**: e.g. AT&T FSM library.
  - ☞ Kernel operations implemented with a few library functions.
  - ☞ Existing algorithms to optimize the computation:  
determinization, minimization.

# Spoken-Language Classification – Task

Call-classification in AT&T *How May I Help You* and *VoiceTone*<sup>®</sup> spoken dialog applications.

- ▶▶▶▶ A customer calls a service and speaks naturally.
- ▶▶▶▶ **Speech recognition** outputs: acyclic weighted automata (*word lattices*).■
- ▶▶▶▶ **Goal:** route the call or send it to a **dialog** manager.
- ▶▶▶▶ **Categories:** call-types and named entities such as *Billing Services*, or *Calling Plans*.■

3 deployed customer-care applications for Health-care and Telecommunication industries: 64 to 97 categories.

- ▶▶▶▶ **Data:** word lattices for 10 to 35 thousand training utterances.
- ▶▶▶▶ **Vocabulary** of 5 to 8 thousand words.

# Spoken-Language Classification – Experimental Result

Start from Bag-of-Word features from *one-best* sentence Err=29% ■

▣ Use word *bigrams* Err ↘ 24.5% ■

☞ 18 bigram/sentence on average!

▣ Use word *trigrams* Err → 24.5% ■

▣ Use *lattice* instead of one-best. Err ↘ 23% ■

▣ Fold preprocessing into Rational kernel based on FSM computations.

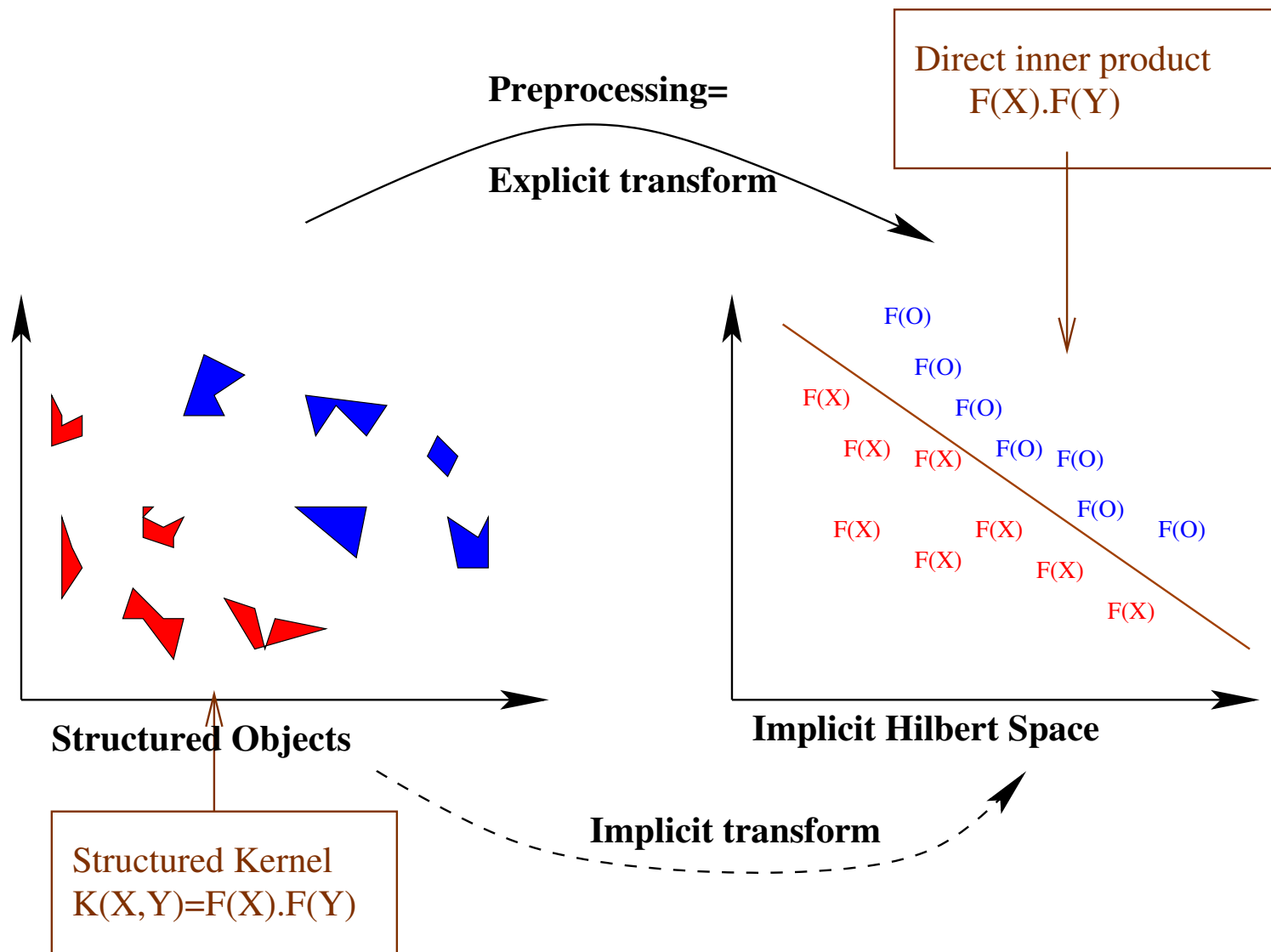
☞ Equivalent models Err → 23% ■

☞ Faster implementation.

▣ Combine rational with *polynomial* kernel. Err ↘ 22% ■

☞ But slow runtime performance.

# Kernel vs. Preprocessing Interpretation



# Kernel vs. Preprocessing Alternative

Structured kernels such as string, rational or tree kernels provide

- ▣▣▣▣▣ Frameworks to understand what type of PDS kernels can be built
- ▣▣▣▣▣ Efficient and generic implementations■

They avoid computing features whose number:

- ▣▣▣▣▣ *in theory*, grow exponentially
- ▣▣▣▣▣ *in practice*, number of non-zero features surprisingly small.■

When to prefer a combination of preprocessing and linear classifiers?

- ▣▣▣▣▣ **Sparsity of Features**: attractive in Natural Language applications.
- ▣▣▣▣▣ Much faster runtime implementation.

# 2<sup>nd</sup> Half: Margin Classifiers

▣ Statistical learning theory

▣ Kernel Classifiers

☞ SVMs: Maximize the margin

☞ Making kernels, kernels for Structured Data

☞ String and Weighted Automata Kernels

▣ *Margin Classifiers*

☞ *Margin-based linear classifiers*

☞ *Generalize the concept of margin to a large class of classifier, including Boosting and Maxent.*

☞ *Large Scale Experiments*

▣ Software and References

# Classification for Natural Language Applications

- ▶▶▶▶ Large number of training examples and classes.
- ▶▶▶▶ Large number of *possible* features
- ▶▶▶▶ Each example: small number of non zero features.

## Examples

- ▶▶▶▶ Spoken Language Understanding
- ▶▶▶▶ Text categorization.
- ▶▶▶▶ Part-of-speech tagging: *millions of examples.*
- ▶▶▶▶ Machine Translation: *> 100,000 classes.*

# Margin-based linear classifiers

Linear classifier  $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$  with trainable weight  $\mathbf{w}$  and bias  $b$ .■

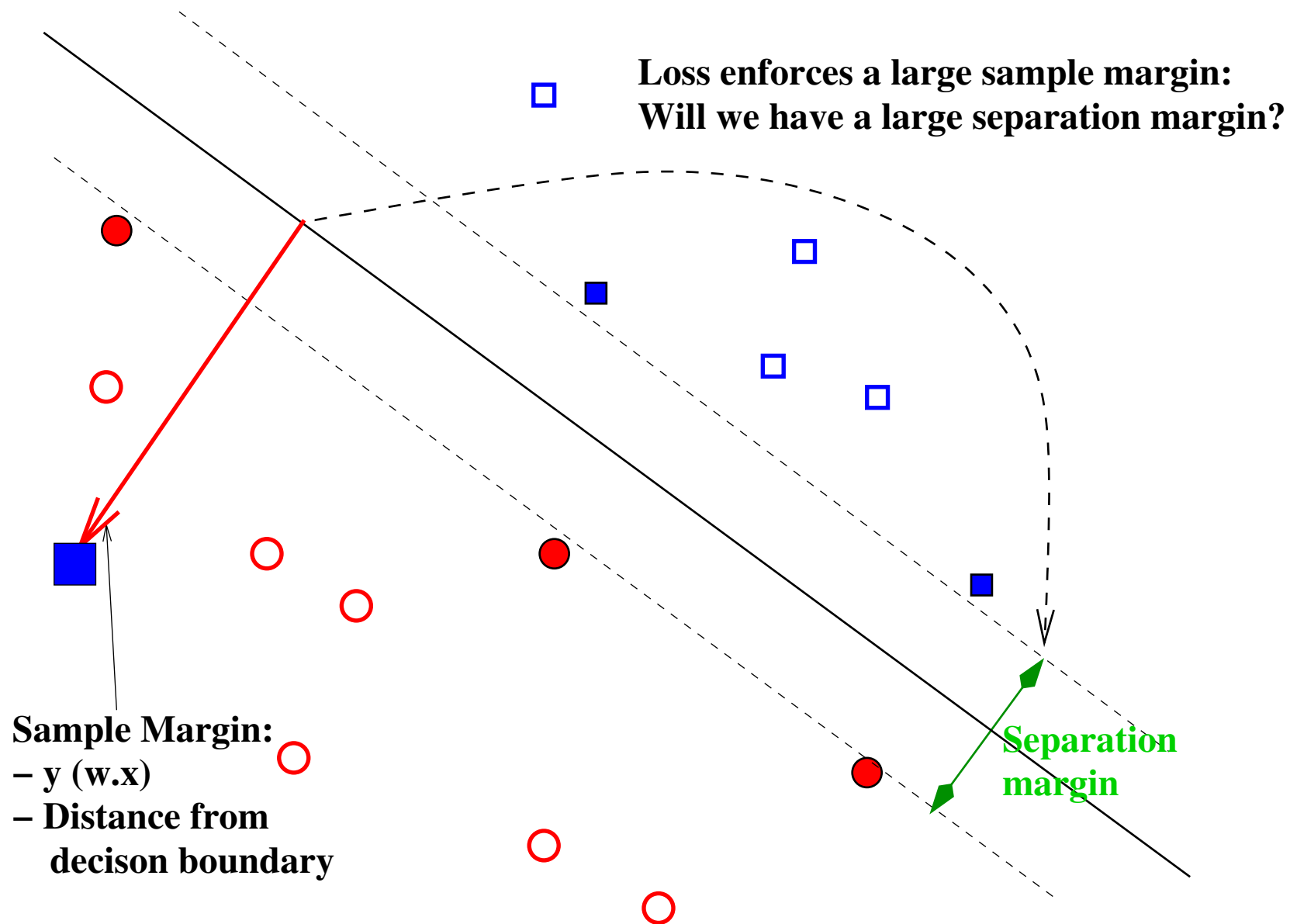
▣▣▣▣▶ The total loss function is a combination of 2 terms

$$\mathcal{L}_{total} = \sum_i \mathcal{L}(f(\mathbf{x}_i), y_i) + \mathcal{R}(\mathbf{w})$$

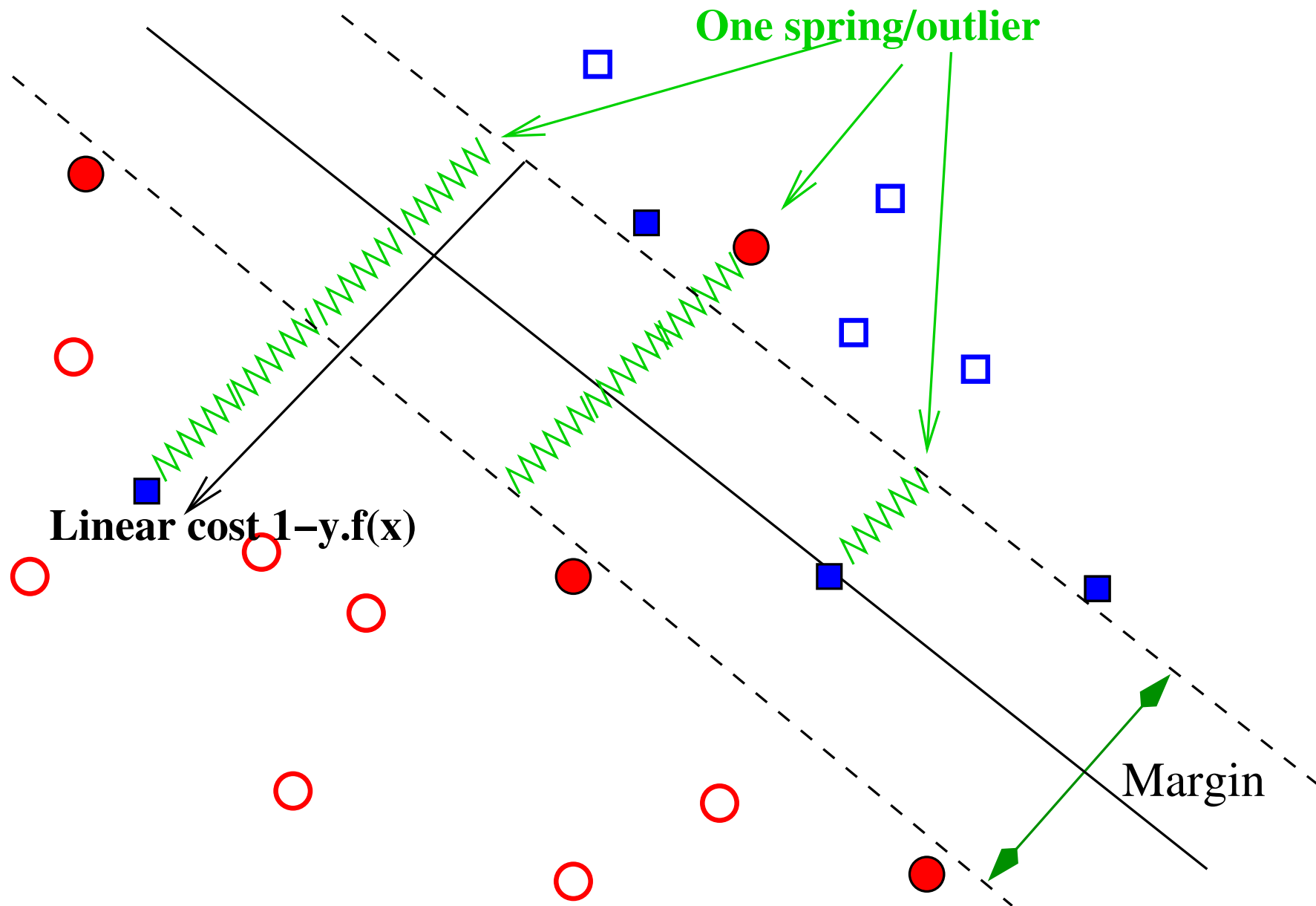
▣▣▣▣▶ The sample loss  $\mathcal{L}(f(\mathbf{x}), y)$  is computed from the sample margin  $y \cdot f(x)$ , hence the name *margin-based*.

▣▣▣▣▶ Regularization term  $\mathcal{R}$  measures the complexity of the model, and encourages simple models which cannot over-train.

# Sample vs. Separation margins



# Example of soft margin support vectors



# Another interpretation of SVMs

## Sample Loss

Hinge loss for soft-margin SVMs:

$$C \sum_i \max(0, 1 - y_i f(\mathbf{x}_i))$$

The cost parameter C:

☞  $\infty$  for hard-margin SVMs

☞ Small C allow more errors

## Regularizer

L2-norm of the weight vector

$$\mathcal{R} = \sum_k w_k^2$$

→ inverse of the SVM margin.

L1-norm:  $\mathcal{R} = \sum_k |w_k|$

→ no Kernel trick possible.

# Feature-based learning

▣▣▣▣▶ *Adaboost*: Features as **weak classifiers**.

☞ incrementally refines a weighted combination of weak classifiers.

☞ importance of examples that remain erroneous is “adaptively boosted”.

▣▣▣▣▶ *Maximum Entropy*: Features as **constraints**.

☞ looks for a distribution over training samples with maximum entropy

☞ satisfying a set of user-defined constraints.

☞ Regularization: relax constraints or use Gaussian prior.

# Boosting for dummies

Examples of calls to classify as *Pay\_Bill*:

- ▣▣▣▣▣ *I'd like to make a payment*
- ▣▣▣▣▣ *I wanna pay my bill*
- ▣▣▣▣▣ *I would like to pay the bill*
- ▣▣▣▣▣ *I would like to pay my mothers long distance bill*
- ▣▣▣▣▣ *I am making a payment with a late payment*
- ▣▣▣▣▣ *I wanna see if I can pay my bill over the phone*

▣ *rules of thumbs* such as the detection of *pay* or *bill* are easy to find.▣

**Boosting:** method for converting rules of thumb or *weak classifiers* into accurate prediction rules

# The boosting family of algorithms

- Training examples  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m) \in \mathcal{X} \times \{-1, 1\}$ .
- for  $t = 1, \dots, T$ :
  - update distribution  $D_t$  on examples  $\{1, \dots, m\}$
  - find *weak hypotheses*  $h_t : \mathcal{X} \rightarrow \{-1, 1\}$
  - with small *error*  $\epsilon_t = \Pr_{D_t}[h_t(\mathbf{x}_i) \neq y_i]$
- Output *final hypotheses*  $H_{final}$ .

Most famous member of the family: **Adaboost** (Freund and Schapire, 1996)

- Using weight-update factor  $\alpha = \frac{1}{2} \log \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$

- $$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \exp(-\alpha_t y_i h_t(\mathbf{x}_i))$$

- $$H_{final}(\mathbf{x}) = \text{sign}(\sum_t \alpha_t h_t(\mathbf{x}))$$

# Adaboost as a Margin Classifier

Adaboost shown to minimize the exponential loss

$$\sum_i \exp(-y_i f(\mathbf{x}_i))$$

## Sample Loss

$$\mathcal{L}(f(\mathbf{x}), y) = \exp(-y_i f(\mathbf{x}))$$

*Logistic* loss also possible:

$$\mathcal{L} = \log(1 + e^{-2yf(\mathbf{x})})$$

## Regularizer

- ➡ No explicit regularization term.
- ➡ Learning concentrates on examples with the smallest margin.
- ➡ Generalization error bound based on L1 margin.

# Conditional Maximum Entropy

Exponential models for + and - examples (with normalization  $Z(x)$ )

$$P(y = +1|\mathbf{x}) = \frac{\exp(yf^+(\mathbf{x}))}{Z(x)} \quad P(y = -1|\mathbf{x}) = \frac{\exp(yf^-(\mathbf{x}))}{Z(x)}$$

Many other models/distributions possible.

## Sample Loss

Log likelihood

$$\mathcal{L}(f(\mathbf{x}), y) = -\log(P(y|\mathbf{x}))$$

$$\mathcal{L} = \log(1 + e^{-2y(f^+(\mathbf{x}) - f^-(\mathbf{x}))})$$

Same as logistic regression.

## Regularizer

Relax constraints:

parameterized L1-norm

$$\mathcal{R} = \sum_k \beta_k |w_k|$$

Log of Gaussian prior:

L2-norm

# Optimization space

## Primal- features

Weight vector: 1 parameter per feature

▣▣▣▣▶ Adaboost

▣▣▣▣▶ MaxEnt

Sequential algorithms:

☞ add 1 feature/iteration

☞  $O(N)$  complexity.

Faster than Iterative Scaling.

## Dual - vectors

SVMs allows decomposition:

$$f(\mathbf{x}) = \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x})$$

Optimization over  $\alpha$ .

Sequential algorithms:

☞ Most popular: SMO

☞ add 2 vectors/iteration

☞  $O(N^2)$  complexity.

# Multiclass learning - Previous work

- ▣ Large Margin Classifiers are by essence *binary*.
- ▣ Many combination of binary classifiers are possible:
  - ☞ *1-vs-other*: separate examples belonging to one class from all others (Rifkin and Klautau, 2004).
  - ☞ *1-vs-1*: each classifier is trained to separate a pair of classes.
  - ☞ *ECOC*: error correcting output codes. (Crammer and Singer, 2000)
  - ☞ Hierarchical approaches have only been studied recently.■
- ▣ Problem with methods based on recombination of binary classifiers:
  - ☞ Each classifier corresponds to a different optimization problem.
  - ☞ Do not guarantee a global optimality.
- ▣ Global optimization procedure possible but expensive:  
Often requires  $C$  times more time and memory.

# Specific Multiclass Interpretations

## Definition of a multiclass sample margin:

- ▶▶▶ Target class  $c$ : margin is  $f_c(\mathbf{x}) - \sup_{d \neq c} f_d(\mathbf{x})$ .
- ▶▶▶ Costly global optimization required, without improved results
  - ▶ Adaboost.MR (Schapire and Singer, 2000).
  - ▶ Several SVM formalizations.■

## Probabilistic framework

- ▶▶▶ Maxent: many ways to define distributions over multiple classes
  - ▶ Beyond conditional: *joint* or class conditional.
  - ▶ Assumptions about how classes correlate.
- ▶▶▶ SVMs  $\approx$  probability with univariate logistic regression (Platt, 1999).

# Large Scale Experiments

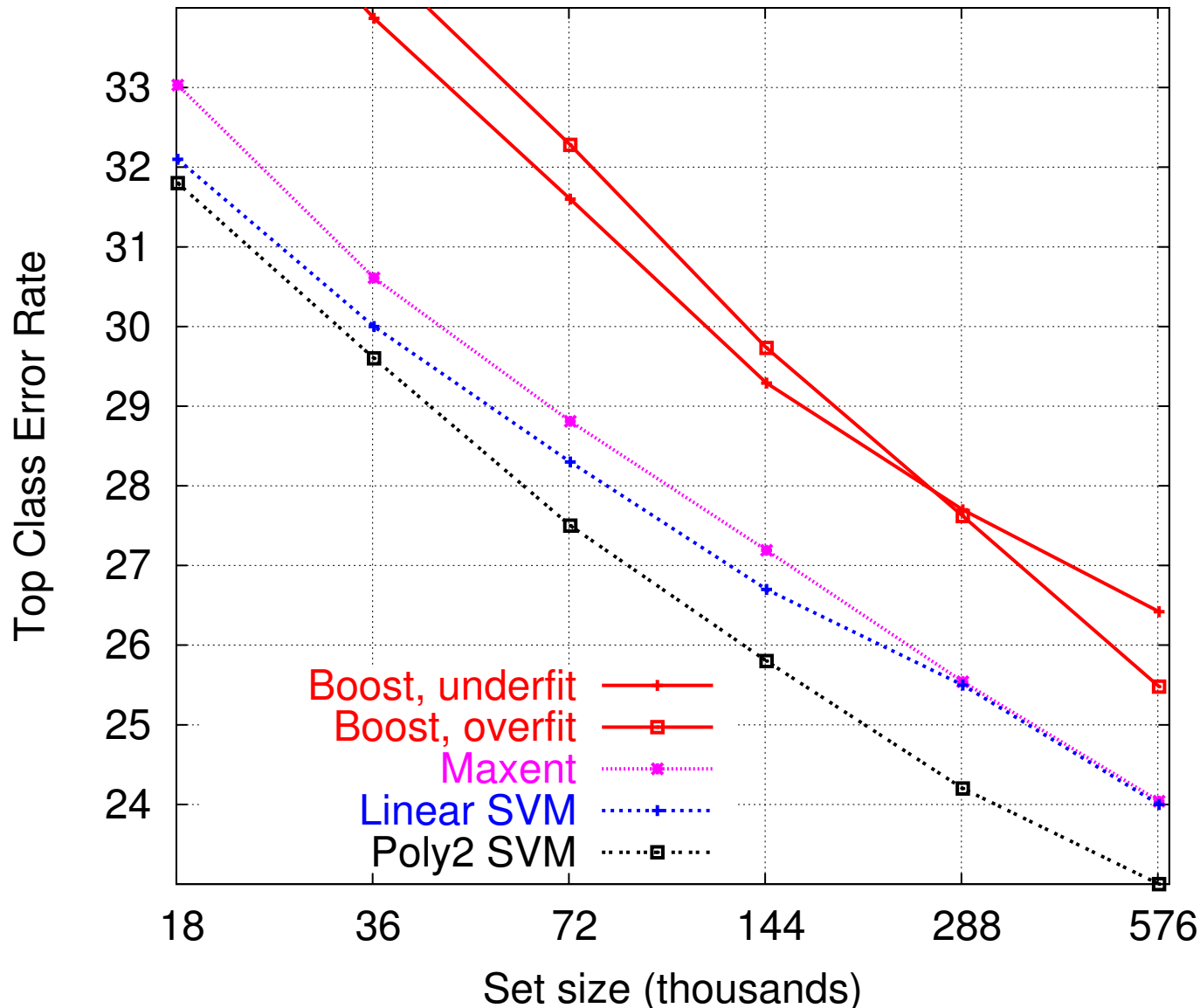
## AT&T *VoiceTone*<sup>®</sup> natural dialog system:

- Collect together datasets from different applications.
- Merge call-types when possible (for instance, all the “Request(customer care)”). *600* total.
- Very large and noisy hand-labeled data with *580,000* training examples.

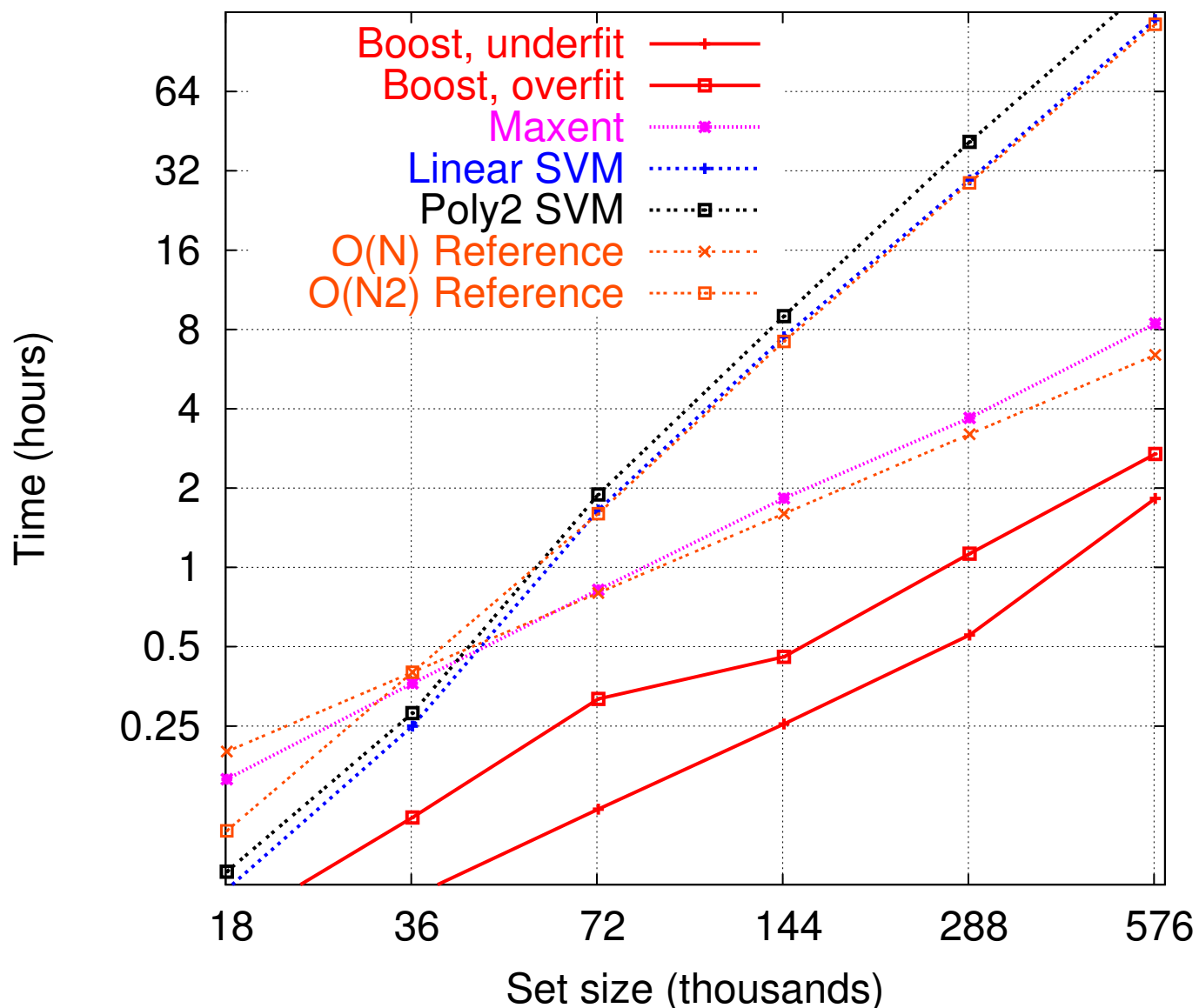
## Methods experimented

- Linear: Adaboost
- Linear, regularized: Maxent, Linear SVM.
- Kernel, Regularized: SVM with  $2^{nd}$  degree polynomial.

# Scaling up from 18K to 580K samples: Test Error



# Scaling up from 18K to 580K samples: Training time



# Comparative Summary

|                  | SVM              | Adaboost        | Maxent    |
|------------------|------------------|-----------------|-----------|
| Combine          | vectors          | features        | features  |
| Regularization   | L2 or L1         | implicit        | L2 or L1  |
| Capacity         | f(kernel)        | linear          | linear    |
| Test error       | f(kernel, regul) | hard to improve | f(regul)  |
| Training time    | $O(m^2)$         | $O(m)$          | $O(m)$    |
| Sparsity/runtime | poor             | excellent       | very good |
| Prob. interpret. | with remapping   | with remapping  | natural   |

Color codes: best performer, worst performer

## 2<sup>nd</sup> half: Concluding remarks

Choosing the right approach:

- ▣▣▣▣➤ Kernels give you more representation power:
  - ☞ Fold your preprocessing into a kernel.
  - ☞ Slower learning and runtime...
- ▣▣▣▣➤ Many sparse features:
  - ☞ Linear classifiers often sufficient.
  - ☞ Use margin classifiers: generalize well.
  - ☞ Use feature-based sequential learning: very fast
- ▣▣▣▣➤ Control your regularization.
- ▣▣▣▣➤ Excellent software available.

# Acknowledgments

This half of the tutorial relies on contributions and help of:  
Srinivas Bangalore, Leon Bottou, Corinna Cortes, Dilek Hakkani-Tur,  
Mazin Gilbert, Yann LeCun, Mehryar Mohri, Steven Phillips, Rob  
Schapire, Juergen Schroeter, Gokhan Tur, Vladimir Vapnik.

# Online References

➡ SVM: <http://www.kernel-machines.org/>

➡ Boosting:

☞ Website: <http://www.boosting.org/>

☞ Schapire's publications:

<http://www.cs.princeton.edu/~schapire/boost.htm>

☞ Freund's applet:

<http://www1.cs.columbia.edu/~freund/adaboost/>

➡ Maxent:

<http://www-2.cs.cmu.edu/~aberger/maxent.html>

# Software

## ➡ **FSM library**

`http://www.research.att.com/sw/tools/fsm`

## ➡ **SVM light** `http://svmlight.joachims.org/`

☞ Ready to use implementation optimized for Natural Language applications. Can handle structured data.

## ➡ **LIBSVM**

`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

☞ An integrated tool implementing support vector classification, regression, and distribution estimation. C++ and Java sources available.

## ➡ **SVM torch** `http://www.torch.ch/`

☞ a modular library supporting SVMs, Adaboost and gradient based learning

## ➡ **BoosTexter** `www.cs.princeton.edu/~schapire/BoosTexter/`

☞ a general purpose machine-learning program based on boosting for building a classifier from text and/or attribute-value data.

# Books

- ▶▶▶▶ Trevor Hastie, Robert Tibshirani and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2004  
Covers a broad range, from supervised learning, to unsupervised learning, including classification trees, neural networks, and support vector machines.
- ▶▶▶▶ John Shawe-Taylor and Nello Cristianini *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.  
An introduction to Kernel methods which is accessible in its description of the theoretical foundations of large margin algorithms (467 pages).
- ▶▶▶▶ Bernhard Schölkopf and Alex Smola *Learning with Kernels*. MIT Press, Cambridge, MA, 2002  
An introduction and overview over SVMs. (650 pages)
- ▶▶▶▶ Vladimir Vapnik *Statistical Learning Theory*. Wiley, NY, 1998.  
The comprehensive treatment of statistical learning theory, including a large amount of material on SVMs (768 pages,)

# References

COLLINS, MICHAEL AND NIGEL DUFFY. 2002. NEW RANKING ALGORITHMS FOR PARSING AND TAGGING: KERNELS OVER DISCRETE STRUCTURES, AND THE VOTED PERCEPTRON . IN *ACL*.

CORTES, CORINNA, PATRICK HAFFNER, AND MEHRYAR MOHRI. 2004. RATIONAL KERNELS: THEORY AND ALGORITHMS. *Journal of Machine Learning Research*, 5:1035–1062, AUGUST.

CRAMMER, KOBY AND YORAM SINGER. 2000. ON THE LEARNABILITY AND DESIGN OF OUTPUT CODES FOR MULTICLASS PROBLEMS. IN *Proceedings of COLT'00*, PAGES 35–46.

FREUND, YOAV AND ROBERT E. SCHAPIRE. 1996. EXPERIMENTS WITH A NEW BOOSTING ALGORITHM. IN *International Conference on Machine Learning*, PAGES 148–156.

HAUSSLER, DAVID. 1999. CONVOLUTION KERNELS ON DISCRETE STRUCTURES. TECHNICAL REPORT UCSC-CRL-99-10, UNIVERSITY OF CALIFORNIA AT SANTA CRUZ.

JOACHIMS, THORSTEN. 1998. TEXT CATEGORIZATION WITH SUPPORT VECTOR MACHINES: LEARNING WITH MANY RELEVANT FEATURES. IN *Proc. of ECML-98*. SPRINGER VERLAG.

LESLIE, CHRISTINA, ELEAZAR ESKIN, JASON WESTON, AND WILLIAM STAFFORD NOBLE. 2003. MISMATCH STRING KERNELS FOR SVM PROTEIN CLASSIFICATION. IN *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, VANCOUVER, CANADA, MARCH. MIT PRESS.

LODHI, HUMA, JOHN SHAWE-TAYLOR, NELLO CRISTIANINI, AND CHRIS WATKINS. 2001. TEXT CLASSIFICATION USING STRING KERNELS. IN *Advances in Neural Information Processing Systems 13 (NIPS 2000)*, PAGES 563–569. MIT PRESS.

PLATT, J. 1999. PROBABILISTIC OUTPUTS FOR SUPPORT VECTOR MACHINES AND COMPARISON TO REGULARIZED LIKELIHOOD METHODS. IN *NIPS*. MIT PRESS.

RIFKIN, RYAN AND ALDEBARO KLAUTAU. 2004. IN DEFENSE OF ONE-VS-ALL CLASSIFICATION. *Journal of Machine Learning Research*, PAGES 101–141.

SCHAPIRE, ROBERT E. AND YORAM SINGER. 2000. BOOSTEXTER: A BOOSTING-BASED SYSTEM FOR TEXT CATEGORIZATION. *Machine Learning*, 39(2/3):135–168.

VAPNIK, VLADIMIR N. 1998. *Statistical Learning Theory*. JOHN WILEY & SONS, NEW-YORK.

VISWANATHAN, S. V. N. AND ALEXANDER J. SMOLA. 2002. FAST KERNELS FOR STRING AND TREE MATCHING. IN *NIPS*, PAGES 569–576.