Feature engineering

Léon Bottou

COS 424 – 4/22/2010

Summary

Summary

I. The importance of features
II. Feature relevance
III. Selecting features
IV. Learning features

I. The importance of features

Simple linear models

People like simple linear models with convex loss functions

– Training has a unique solution.

– Easy to analyze and easy to debug.

Which basis functions Φ?

– Also called the *features*.

Many basis functions

– Poor testing performance.

Few basis functions

– Poor training performance, in general.

– Good training performance if we pick the right ones.

– The testing performance is then good as well.

Explainable models

Modelling for prediction

– Sometimes one builds a model for its predictions.

– The model is the operational system.

– Better prediction $\Longrightarrow$ \$\$\$.

Modelling for explanations

– Sometimes one builds a model for interpreting its structure.

– The human acquires knowledge from the model.

– The human then design the operational system.

(we need humans because our modelling technology is insufficient.)

Selecting the important features

– More compact models are usually easier to interpret.

– A model optimized for explanability is not optimized for accuracy.

– Identification problem vs. emulation problem.

Feature explosion

Initial features

– The initial pick of feature is always an expression of prior knowledge.

images	⟶	pixels, contours, textures, etc.
signal	⟶	samples, spectrograms, etc.
time series	⟶	ticks, trends, reversals, etc.
biological data	⟶	dna, marker sequences, genes, etc.
text data	⟶	words, grammatical classes and relations, etc.

Combining features

– Combinations that linear system cannot represent:
  polynomial combinations, logical conjunctions, decision trees.

– Total number of features then grows very quickly.

Solutions

– Kernels (with caveats, see later)

– Feature selection (but why should it work at all?)

II. Relevant features

Assume we know distribution $p(X, Y)$.

$$\begin{aligned} Y &: \text{ output} \\ X &: \text{ input, all features} \\ X_i &: \text{ one feature} \\ R_i = X \setminus X_i &: \text{ all features but } X_i, \end{aligned}$$

Probabilistic feature relevance

Strongly relevant feature

– Definition: $X_i \not\perp\!\!\!\perp Y \mid R_i$

Feature X_i brings information that no other feature contains.

Weakly relevant feature

– Definition: $X_i \not\perp\!\!\!\perp Y \mid S$ for some strict subset S of R_i.

Feature X_i brings information that also exists in other features.

Feature X_i brings information in conjunction with other features.

Irrelevant feature

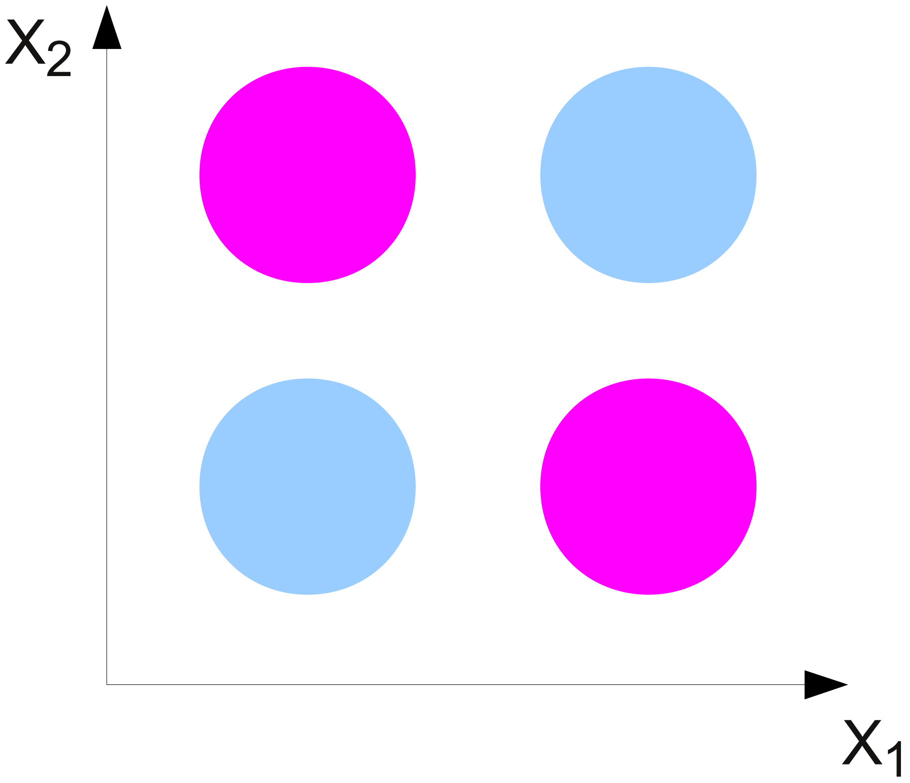
– Definition: neither strongly relevant nor weakly relevant.

Stronger than $X_i \perp\!\!\!\perp Y$. See the XOR example.

Relevant feature

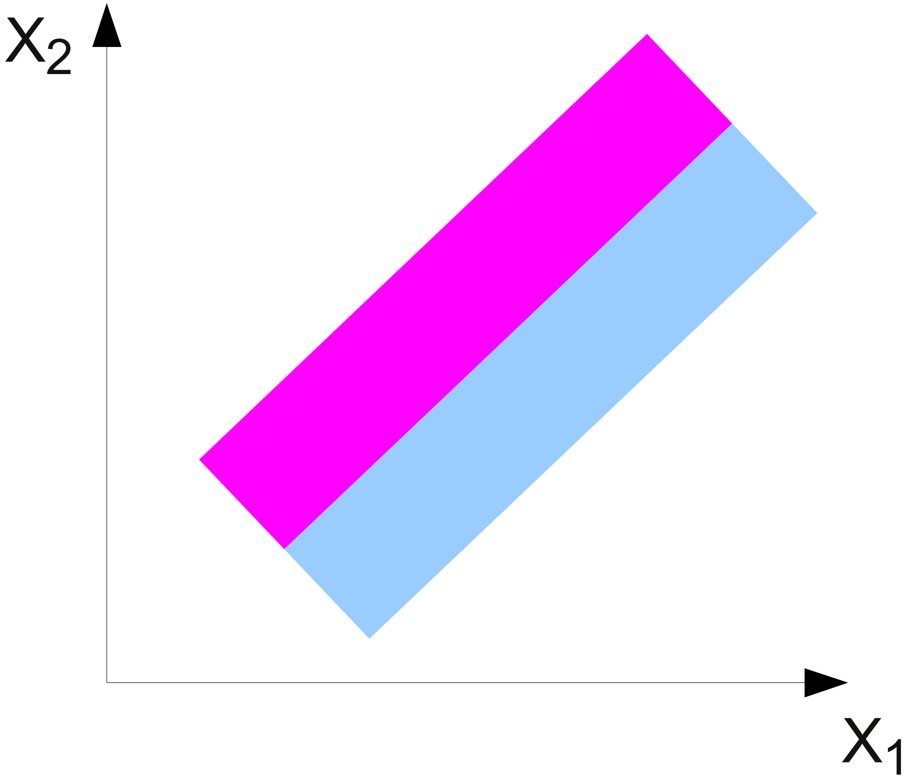
– Definition: not irrelevant.

Interesting example



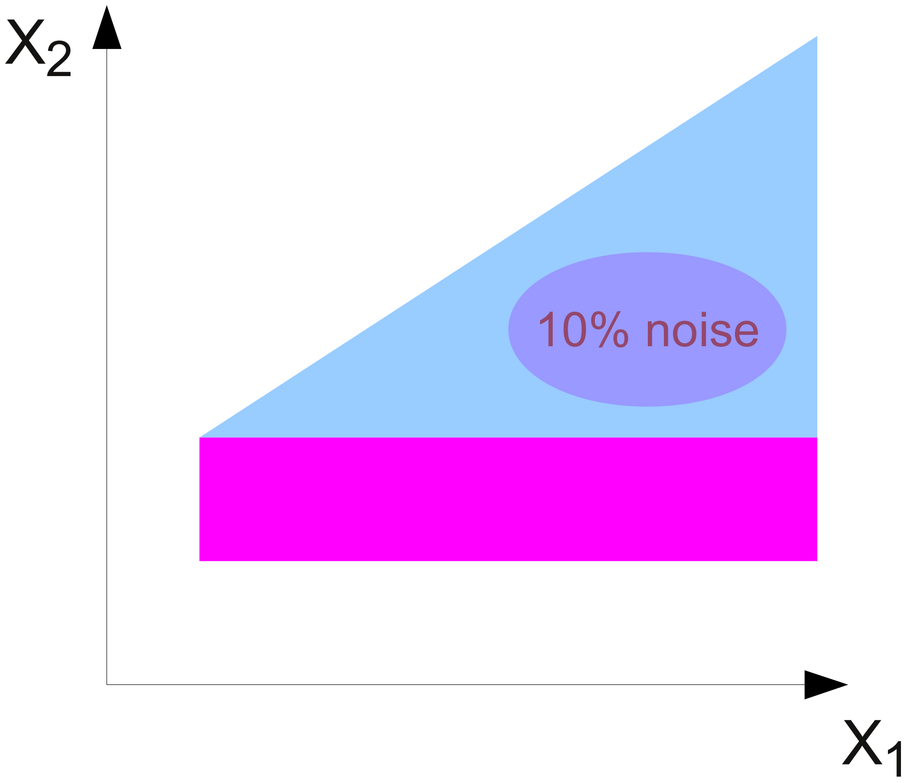
Two variables can be useless by themselves but informative together.

Interesting example



Correlated variables may be useless by themselves.

Interesting example



Strongly relevant variables may be useless for classification.

Bad news

Forward selection

– Start with empty set of features $S_0 = \emptyset$.

– Incrementally add features X_t such that $X_t \not\perp\!\!\!\perp Y \mid S_{t-1}$.

Will find all strongly relevant features.

May not find some weakly relevant features (e.g. xor).

Backward selection

– Start with full set of features $S_0 = X$.

– Incrementally remove features X_i such that $X_t \perp\!\!\!\perp Y \mid S_{t-1} \setminus X_t$.

Will keep all strongly relevant features.

May eliminate some weakly relevant features (e.g. redundant).

Finding all relevant features is NP-hard.

– Possible to construct a distribution that demands an exhaustive search through all the subsets of features.

III. Selecting features

How to select relevant features
when $p(x, y)$ is unknown
but data is available?

Selecting features from data

Training data is limited

– Restricting the number of features is a capactity control mechanism.

– We may want to use only a subset of the relevant features.

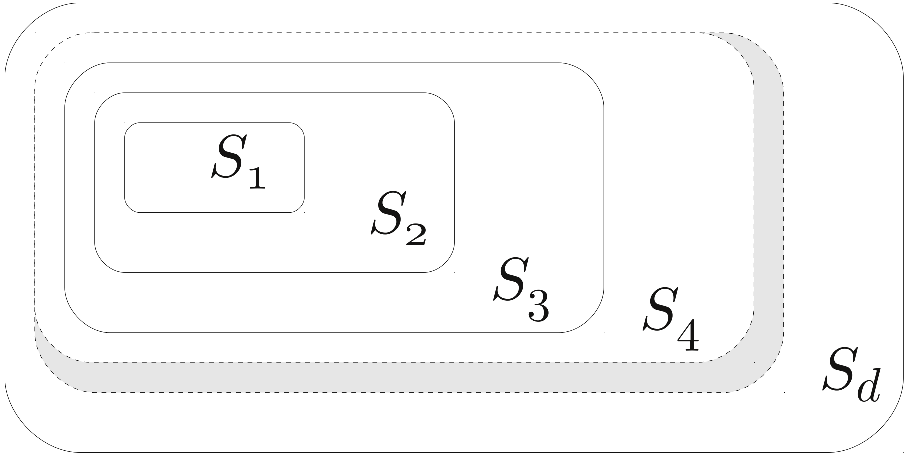
Notable approaches

– Feature selection using regularization.

– Feature selection using wrappers.

– Feature selection using greedy algorithms.

L_0 structural risk minimization



S_r : classifiers with at most r features.

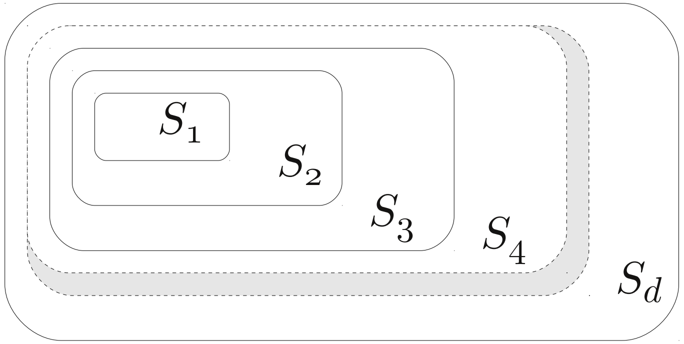
Algorithm

1. For $r = 1 \dots d$, find system $f_r \in S_r$ that minimize training error.
2. Evaluate f_r on a validation set.
3. Pick $f^\star = \arg\min_r E_{\text{valid}}(f_r)$

Note

– The NP-hardness remains hidden in step (1).

L_0 structural risk minimization



S_r : classifiers with at most r features.

Let $E_r = \min_{f \in S_r} E_{\text{test}}(f)$. The following result holds (Ng 1998):

$$E_{\text{test}}(f^\star) \leq \min_{r=1\dots d} \left\{ E_r + \tilde{\mathcal{O}}\left(\sqrt{\frac{h_r}{n_{\text{train}}}}\right) + \tilde{\mathcal{O}}\left(\sqrt{\frac{r \log d}{n_{\text{train}}}}\right) \right\} + \mathcal{O}\left(\sqrt{\frac{\log d}{n_{\text{valid}}}}\right)$$

Assume E_r is quite good for a low number of features r.

Meaning that few features are relevant.

Then we can still find a good classifier if h_r and $\log d$ are reasonable.

We can filter an exponential number of irrelevant features.

L_0 regularisation

$$\min_{\mathbf{w}} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(y, f_{\mathbf{w}}(x)) \; + \; \lambda \; \mathrm{count}\{w_j \neq 0\}$$

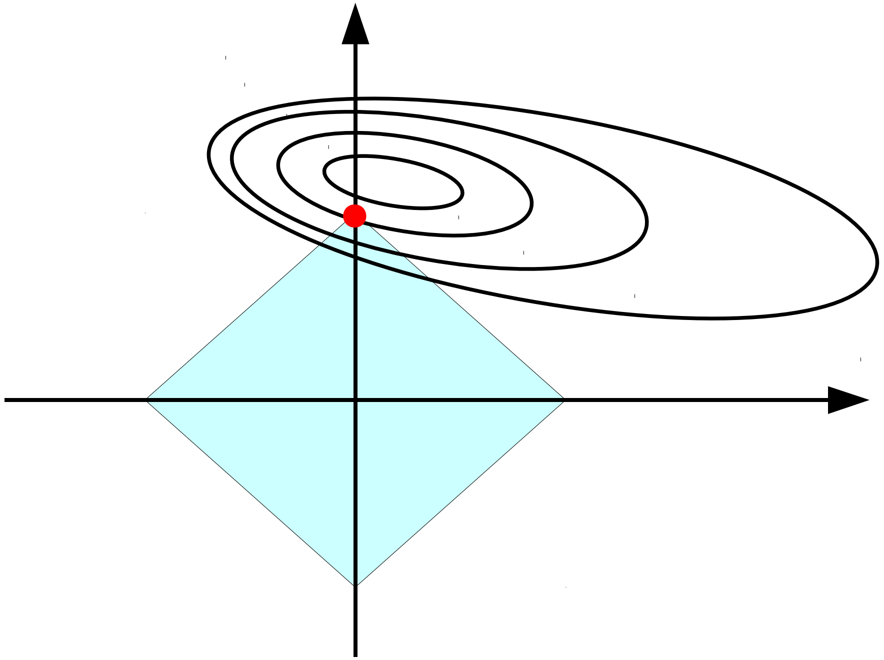
This would be the same as L0-SRM.

But how can we optimize that?

L_1 regularisation

The L_1 norm is the first convex L_p norm.

$$\min_{\mathbf{w}} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(y, f_{\mathbf{w}}(x)) \;+\; \lambda|\mathbf{w}|_1$$



Same logarithmic property (Tsybakov 2006).

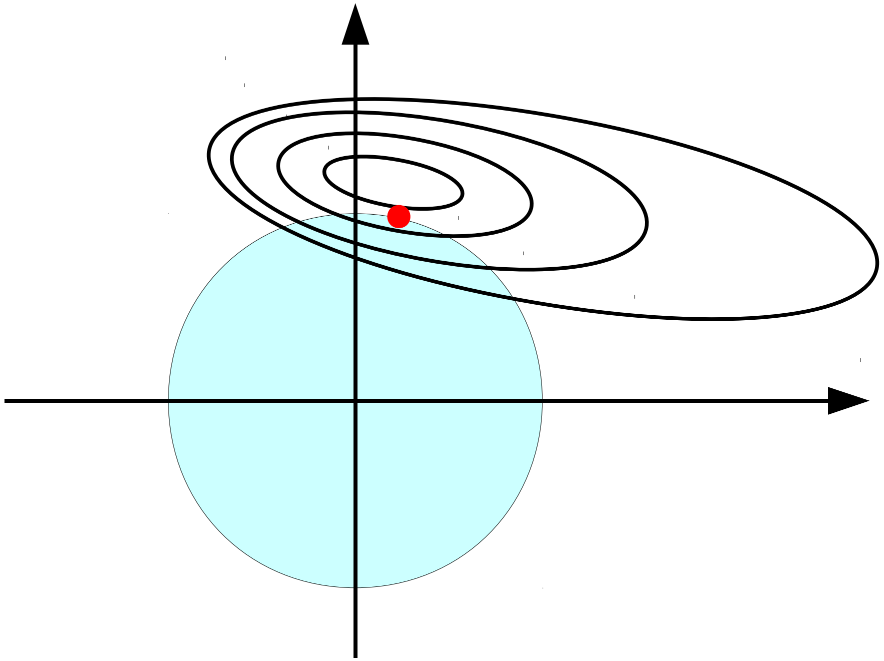
L_1 regulatization can weed an exponential number of irrelevant features.

See also *"compressed sensing"*.

L_2 regularisation

The L_2 norm is the same as the maximum margin idea.

$$\min_{\mathbf{w}} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(y, f_{\mathbf{w}}(x)) \;+\; \lambda\|\mathbf{w}\|_2$$



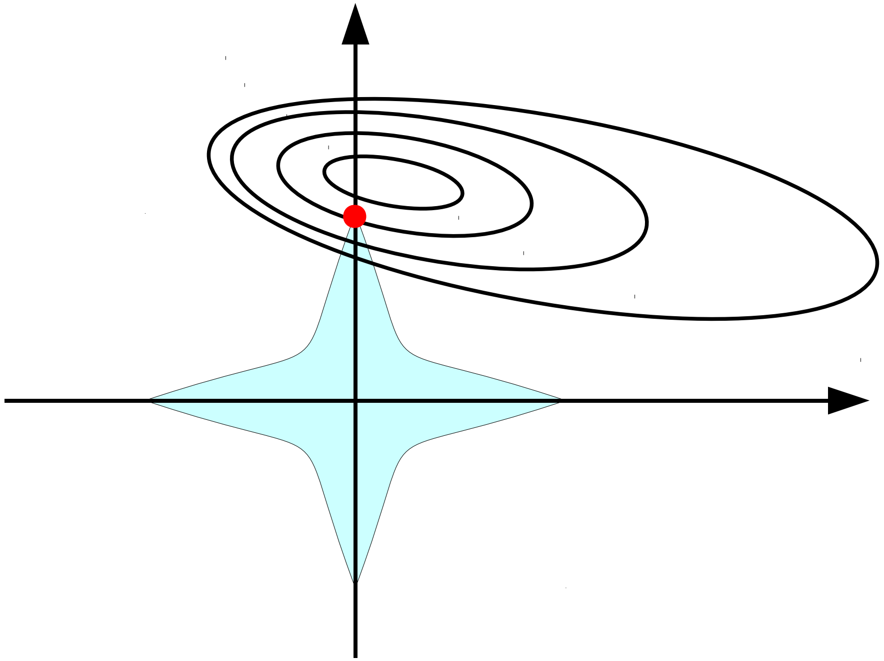
Logarithmic property is lost.

Rotationally invariant regularizer!

SVMs do not have magic properties for filtering out irrelevant features. They perform best when dealing with lots of relevant features.

$L_{1/2}$ regularization ?

$$\min_{\mathbf{w}} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(y, f_{\mathbf{w}}(x)) \;+\; \lambda\|\mathbf{w}\|_{\frac{1}{2}}$$



This is non convex.

Therefore hard to optimize.

Initialize with L_1 norm solution

then perform gradient steps.

This is surely not optimal,
but gives sparser solutions
than L_1 regularization !

Works better than L_1 in practice.

But this is a *secret*!

Wrapper approaches

Wrappers

- Assume we have chosen a learning system and algorithm.
- Navigate feature subsets by adding/removing features.
- Evaluate on the validation set.

Backward selection wrapper

- Start with all features.
- Try removing each feature and measure validation set impact.
- Remove the feature that causes the least harm.
- Repeat.

Notes

- There are many variants (forward, backtracking, etc.)
- Risk of overfitting the validation set.
- Computationally expensive.
- Quite effective in practice.

Greedy methods

Algorithms that incorporate features one by one.

Decision trees

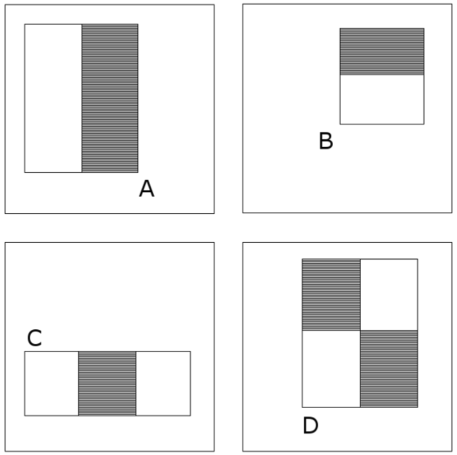
- Each decision can be seen as a feature.
- Pruning the decision tree prunes the features

Ensembles

- Ensembles of classifiers involving few features.
- Random forests.
- Boosting.

Greedy method example

The Viola-Jones face recognizer



Lots of very simple features.

$$\sum_{R \in \text{Rects}} \alpha_r \sum_{(i,j) \in R} x[i,j]$$

Quickly evaluated by first precomputing

$$X_{i_0 j_0} = \sum_{i \le i_0} \sum_{j \le j_0} x[i,j]$$

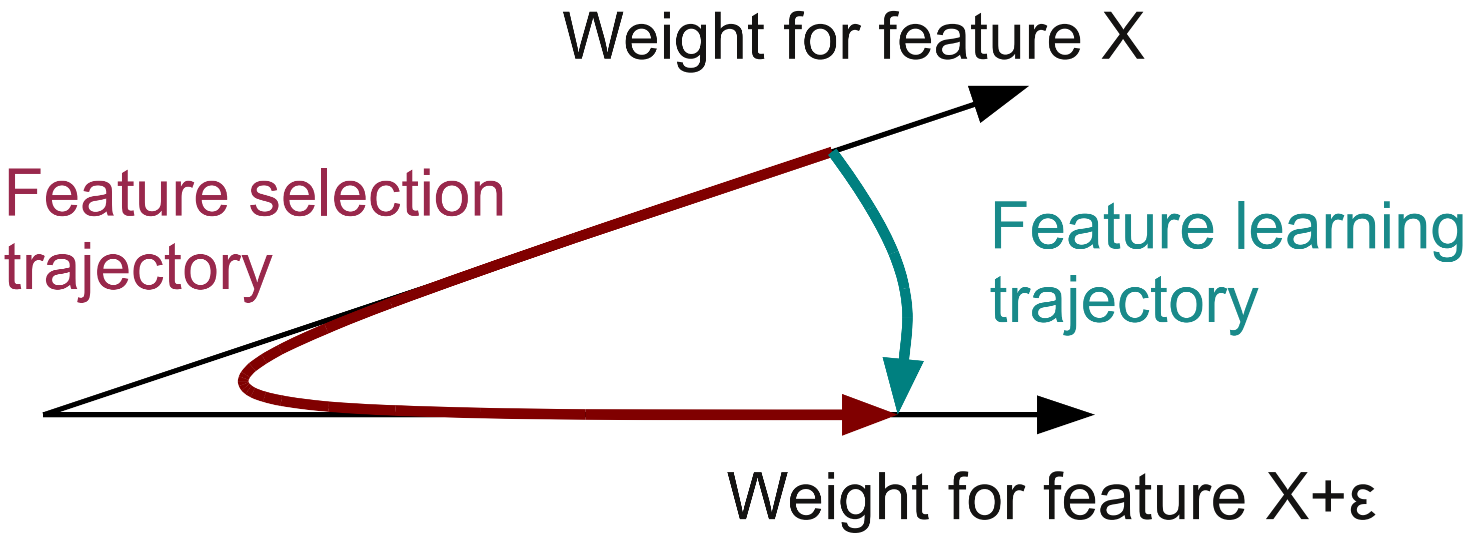
Run AdaBoost with weak classifiers bases on these features.

IV. Feature learning

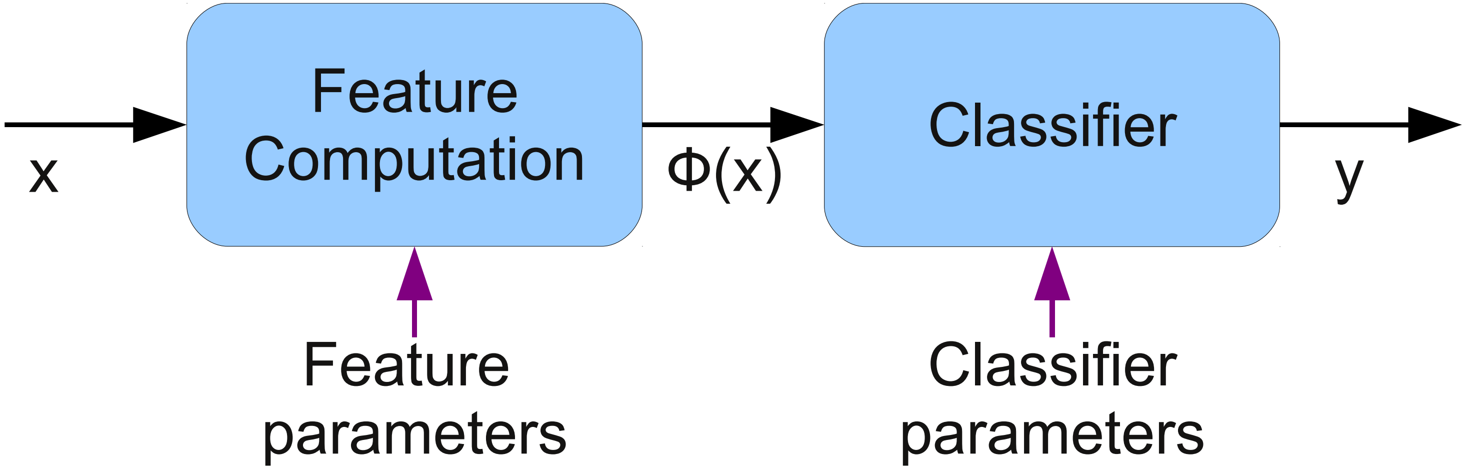
Feature learning in one slide

Suppose we have weight on a feature X.

Suppose we prefer a closely related feature $X + \epsilon$.



Feature learning and multilayer models



Feature learning for image analysis

2D Convolutional Neural Networks

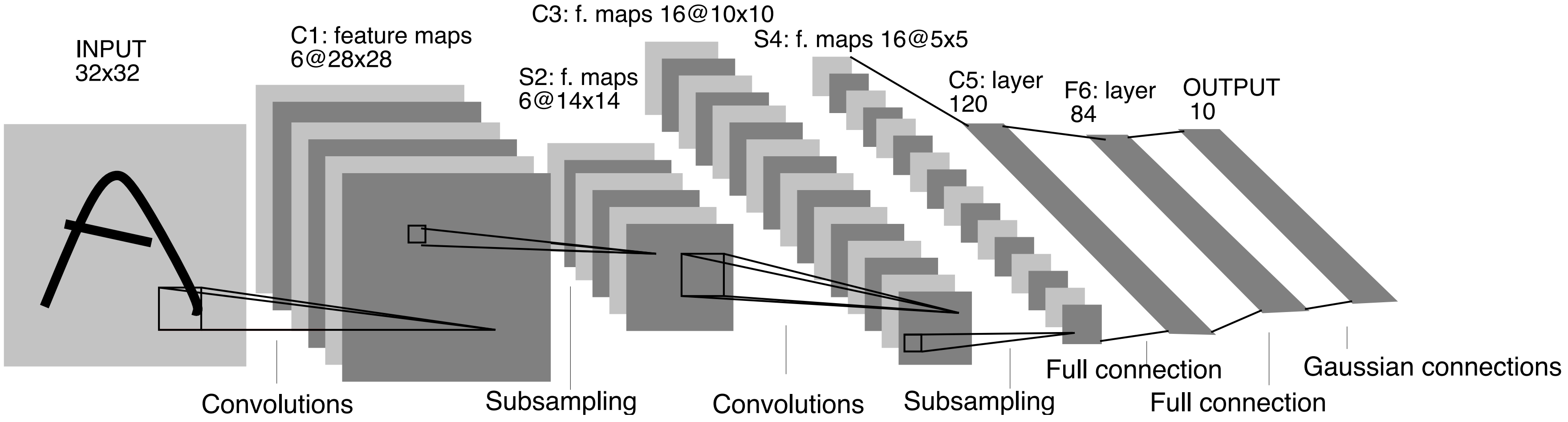
– 1989: isolated handwritten digit recognition

– 1991: face recognition, sonar image analysis

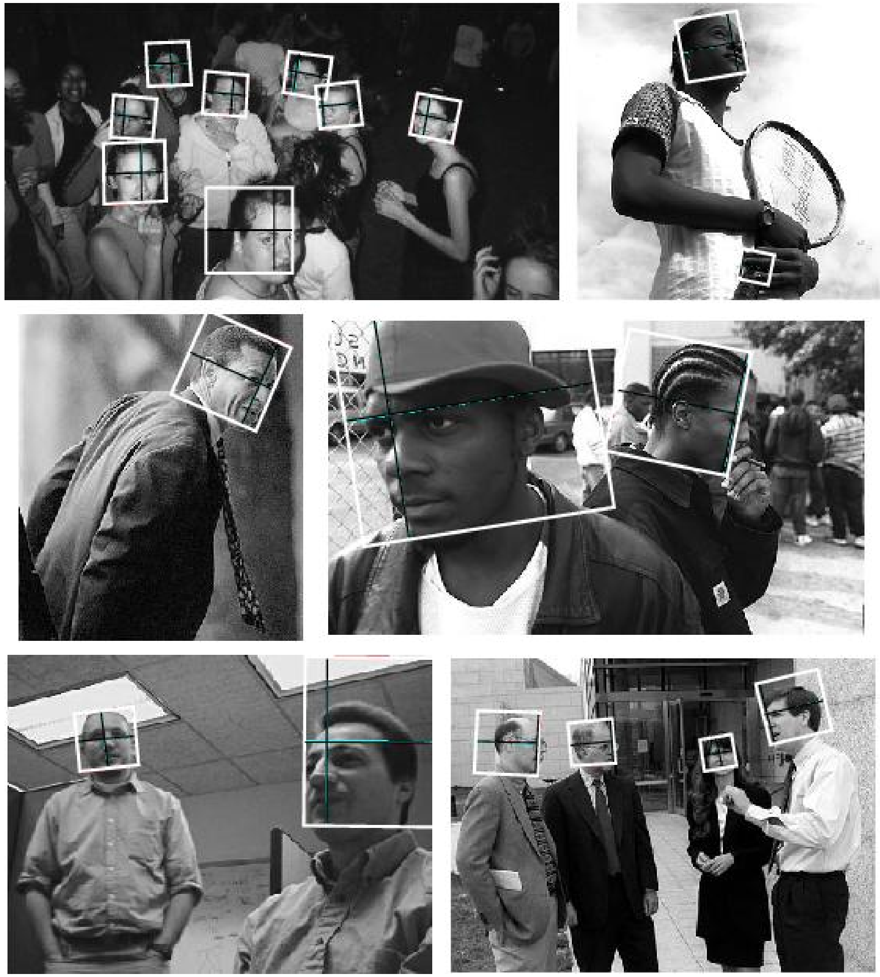
– 1993: vehicle recognition

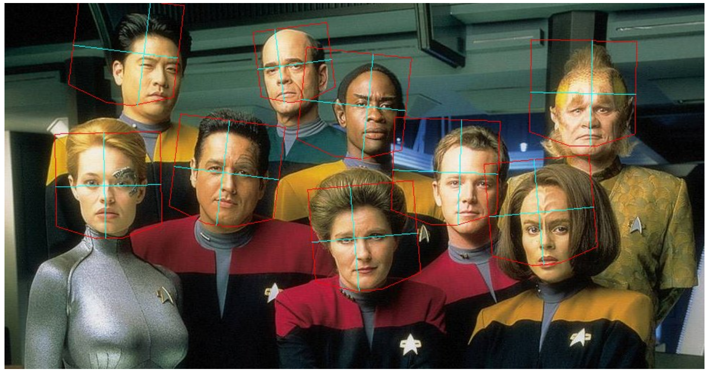
– 1994: zip code recognition

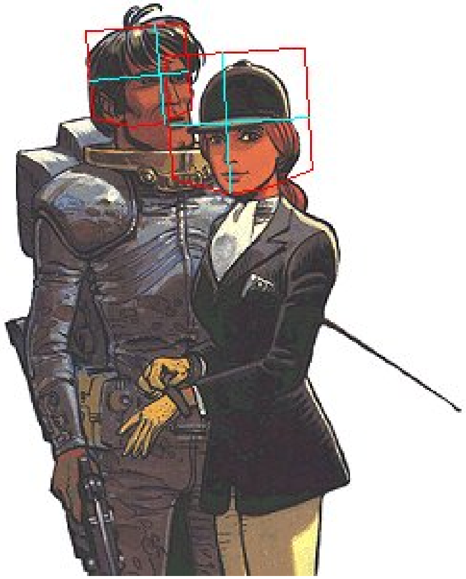
– 1996: check reading



Feature learning for face recognition







Note: more powerful but slower than Viola-Jones

Feature learning revisited

Handcrafted features

- Result from knowledge acquired by the feature designer.
- This knowledge was acquired on multiple datasets associated with related tasks.

Multilayer features

- Trained on a single dataset (e.g. CNNs).
- Requires lots of training data.
- Interesting training data is expensive

Multitask/multilayer features

- In the vicinity of an interesting task with costly labels there are related tasks with abundant labels.
- Example: face recognition $\leftrightarrow$ face comparison.
- More during the next lecture!