6.891 Machine learning and neural networks

Tommi Jaakkola
MIT AI LAB
tommi@ai.mit.edu

LECTURE 6 (September 26, 2000)

Topics

- Feature selection
- Bayesian estimation/inference
- Combining multiple methods
- Boosting

Feature selection

- Various objectives
  - Reduction of computations
  - Noise reduction
  - Regularization
  - Relevance detection
- There are roughly two main types of feature selection methods
  1. Filter method
  2. Wrapper method

  We have to make approximations almost always
- We can also do feature *weighting* rather than *selection*

Feature selection: example

- Reducing the number of useless word detectors

$$\begin{aligned} \phi_k &= 0,1 \quad \text{whether } k^{th} \text{ word is present in a document} \\ y &= 0,1 \quad \text{document label} \end{aligned} \tag{1}$$

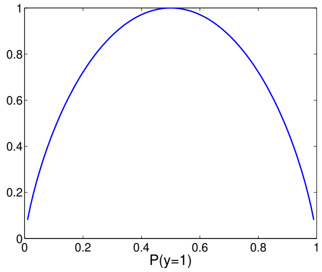
  We get $P(\phi_k|y)$, $P(y)$, and $P(\phi_k) = \sum_{y=0,1} P(\phi_k|y)P(y)$ from the parameter estimation algorithm (maximum likelihood or other)
- We pick only features that provide substantial information about the labels, i.e., those with high *mutual information* with the labels:

$$I(\phi_k; y) = \sum_{\phi,y=0,1} P(\phi_k, y) \log_2 \frac{P(\phi_k, y)}{P(\phi_k)P(y)} \tag{2}$$

Background

- Entropy (uncertainty) of a binary random variable y

$$H(y) \;=\; -\sum_{y=0,1} P(y)\log_2 P(y) \tag{3}$$



Why Shannon entropy?
10101101010100011101101000110101...

Background cont'd

- Properties of mutual information:

$$I(\phi_k; y) = \sum_{\phi, y=0,1} P(\phi_k, y)\log_2 \frac{P(\phi_k, y)}{P(\phi_k)P(y)} \tag{4}$$

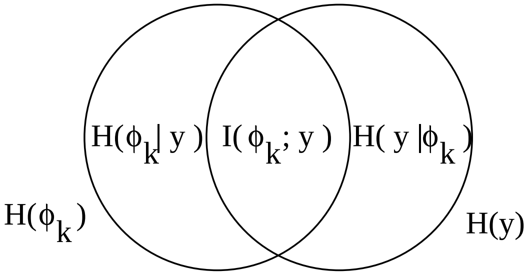
1. $I(\phi_k; y) = I(y; \phi_k)$ (symmetry)
2. If ϕ_k and y are independent, $I(\phi_k; y) = 0$
3. $I(\phi_k; y) \le H(y)$, $I(\phi_k; y) \le H(\phi_k)$
4. $I(\phi_k; y) = H(y) - H(y|\phi_k) = H(\phi_k) - H(\phi_k|y)$

where the conditional entropy $H(y|\phi_k)$ is defined as

$$H(y|\phi_k) \;=\; \sum_{\phi_k=0,1} P(\phi_k)\left[-\sum_{y=0,1} P(y|\phi_k)\log_2 P(y|\phi_k)\right] \tag{5}$$

Background cont'd

- Venn diagram



$$I(\phi_k; y) = H(y) - H(y|\phi_k) = H(\phi_k) - H(\phi_k|y)$$

Feature selection: example

- Reducing the number of useless word detectors

$$\begin{aligned} \phi_k &= 0,1 \quad \text{whether } k^{th} \text{ word is present in a document} \\ y &= 0,1 \quad \text{document label} \end{aligned} \tag{6}$$

We get $P(\phi_k|y)$, $P(y)$, and $P(\phi_k) = \sum_{y=0,1} P(\phi_k|y)P(y)$ from the parameter estimation algorithm (maximum likelihood or other)

- We pick only features that provide substantial information about the labels, i.e., those with high *mutual information* with the labels:

$$I(\phi_k; y) = \sum_{\phi, y=0,1} P(\phi_k, y)\log_2 \frac{P(\phi_k, y)}{P(\phi_k)P(y)} \tag{7}$$

- What approximations are we making here?

Filtering approach: general view

- We find a subset of features that preserve most of the information about the labels

  Let $\Phi = \{\phi_{i_1}, \ldots, \phi_{i_m}\}$ be our current feature set. We can remove any feature ϕ_k for which

  $$\hat{P}(y|\Phi \setminus \phi_k) \approx \hat{P}(y|\Phi), \quad \text{for all settings of the variables } y, \Phi$$

  We have to estimate all the probabilities $\hat{P}(\cdot|\cdot)$ from the available data
- The selection method is generic in the sense that it is independent of the classification method used after feature selection
- Advantage: modularity, speed (?)
- Disadvantage: classification accuracy

Wrapper approach: example

- Let's look at the document classification task again

  We have m possible binary word detectors $\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x})$
- Suppose we wish to use a logistic regression model as the classifier

  $$P(y = 1|\mathbf{x}, \mathbf{w}) = g(\, w_0 + \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})\,) \tag{8}$$
- We wish to construct a subset of features that still retains good predictive accuracy
- We need a search algorithm such as
  1. Greedily add/prune features
  2. Regularization

Wrapper example: greedy approach

1. Find k for which

   $$P(y = 1|\mathbf{x}, \mathbf{w}) = g(\, w_0 + w_k \phi_k(\mathbf{x})\,) \tag{9}$$

   yields the best classifier
2. Find k' for which

   $$P(y = 1|\mathbf{x}, \mathbf{w}) = g(\, w_0 + w_k \phi_k(\mathbf{x}) + w_{k'} \phi_{k'}(\mathbf{x})\,) \tag{10}$$

   yields the best classifier. w_k and $w_{k'}$ are optimized jointly.

- When do we stop?
- Cross-validation

Wrapper example: regularization

$$P(y = 1|\mathbf{x}, \mathbf{w}) = g(\, w_0 + \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})\,) \tag{11}$$

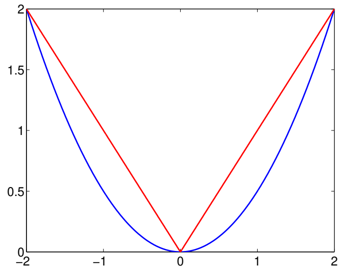
- We can introduce a regularization penalty that tries to set the weights to zero unless they are "useful"

  $$J(\mathbf{w}; C) = \sum_{t=1}^{n} \log P(y_t|\mathbf{x}_t, \mathbf{w}) - C \sum_i |w_i| \tag{12}$$

  where $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ is our training set.
- Note that the selection of the non-zero weights is carried out as a set, not individually
- Why should this regularization penalty work?
- Redundant features?

Wrapper example: regularization

- The effect of the regularization penalty on feature selection depends on its derivative at $w \approx 0$



$w^2/2$ (blue) versus $|w|$ (red)

Wrapper approach: general view

- We find a subset of features that leads to the best classification accuracy GIVEN a specific classification method

  Example: let $\Phi = \{\phi_{i_1}, \ldots, \phi_{i_m}\}$ be our current feature set. We add a feature ϕ_k for which

  $$\hat{P}(\text{ error }|\Phi, \mathbf{w}) > \hat{P}(\text{ error }|\Phi, \phi_k, \mathbf{w}') \quad (13)$$

  where $\mathbf{w}$ and $\mathbf{w}'$ are parameters optimized with the corresponding set of features in the classifier. The probabilities $\hat{P}(\cdot|\cdot)$ need to be estimated from available data.

- Advantage: accuracy with the classifier at hand
- Disadvantage: speed (sometimes)

Where we are

- Feature selection
- *Bayesian estimation/inference*
- Combining multiple methods
- Boosting

Bayesian estimation

- So far we have sought a single parameter setting, a single subset of features
- If there are many nearly optimal settings, why should we settle for a single one?
- The "optimal" or posterior weighting of different settings comes directly from probability calculus:

  $$\overbrace{P(\mathbf{w}|\text{data})}^{\text{Posterior}} = c \cdot \overbrace{P(\text{data}|\mathbf{w})}^{\text{Likelihood}} \overbrace{P(\mathbf{w})}^{\text{Prior}} \quad (14)$$

  where

  $$c = \int P(\text{data}|\mathbf{w})P(\mathbf{w}) = P(\text{data}) \quad (15)$$

  is the normalization constant.

Bayesian estimation: example

- n random bits (0/1), $\{y_1, \ldots, y_n\}$, parameter to be estimated $\theta = P(y=1)$.

$$\overbrace{P(\theta|y_1,\ldots,y_n)}^{\text{Posterior}} = c \cdot \overbrace{\prod_{i=1}^{n} \theta^{y_i}(1-\theta)^{1-y_i}}^{\text{Likelihood}} \cdot \overbrace{1}^{\text{Prior}} \quad (16)$$

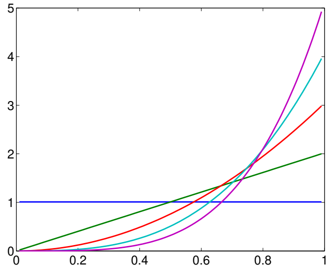
$$= c \cdot \theta^{N_1}(1-\theta)^{n-N_1} \cdot 1 \quad (17)$$

where N_1 is the number of 1's in the observed sequence.

If we observe only 1's, then

$P(\theta|y_1,\ldots,y_n) = c \cdot \theta^n$

What would the maximum likelihood answer be?



Bayesian estimation: properties

- Sequential estimation

$$P(\theta) \quad (18)$$

$$P(\theta|y_1) = c \cdot P(y_1|\theta)\, P(\theta) \quad (19)$$

$$P(\theta|y_1,\ldots,y_i) = c' \cdot P(y_i|\theta)\, P(\theta|y_1,\ldots,y_{i-1}) \quad (20)$$

The answer $P(\theta|y_1,\ldots,y_n)$ does not depend on the order in which we incorporate the observations!

- Prediction: ensemble average

$$P(y|y_1,\ldots,y_n) = \int P(y|\theta)\, P(\theta|y_1,\ldots,y_n)\, d\theta \quad (21)$$

- Overfitting?
- Problems?

Combination of multiple methods

- We want to combine multiple "weak" classifiers/regression methods into a single "strong" method
- Basic (forward fitting) idea: given a regression training set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$

  Step 1: $\hat{f}_1(\mathbf{x}) \leftarrow \text{argmin}_f \sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2/2$

  Step 2: $\hat{f}_2(\mathbf{x}) \leftarrow \text{argmin}_f \sum_{i=1}^{n}(y_i - \hat{f}_1(\mathbf{x}_i) - f(\mathbf{x}_i))^2/2$

  Step 3: ...

  The resulting combined regression method

$$\hat{f}_{comb}(\mathbf{x}) = \hat{f}_1(\mathbf{x}) + \ldots + \hat{f}_m(\mathbf{x}) \quad (22)$$

  has much lower (training) error.

- Boosting is an analogous modular and sequential way of combining weak prediction methods into a strong predictor

Boosting

- A Boosting algorithm sequentially estimates and combines classifiers by reweighting training sets (concentrating on the harder examples)
- AdaBoost preliminaries:

  a) Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ with binary ± 1 labels y_i.

  b) A set of possible "weak" binary (± 1) classifiers $h(\mathbf{x}; \mathbf{w})$, $\mathbf{w} \in \mathcal{W}$ such as *decision stumps*.

  Each decision stump pays attention to only a single component of the input vector

$$h(\mathbf{x}; \mathbf{w}) = \text{sign}(\, w_1 x_k - w_0 \,) \quad (23)$$

  where $\mathbf{w} = \{w_1, w_0, k\}$.

  c) Initially all examples have equal weights: $p(i) = 1/n$.

AdaBoost

1: Find (any) classifier $h(\mathbf{x}; \hat{\mathbf{w}}_k)$ such that the weighted error

$$\epsilon_k = \sum_{i=1}^{n} p(i) \, [y_i \neq h(\mathbf{x}_i; \hat{\mathbf{w}}_k)] \qquad (24)$$

is better than chance. (Here $[y \neq y'] = 1$ if $y \neq y'$ and zero otherwise).

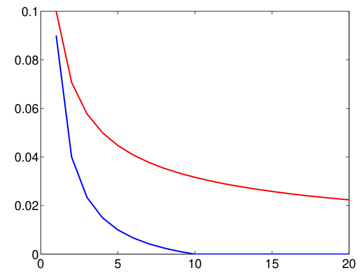
2: Set $\hat{\alpha}_k = \log(\,(1-\epsilon_k)/\epsilon_k\,)$

3: Update example weights: $p(i) \leftarrow c \cdot p(i)$ exp$(\,-\hat{\alpha}_k \, y_i \, h(\mathbf{x}_i; \hat{\mathbf{w}}_k)\,)$ (c is the normalization constant)

- The final hypothesis after m boosting iterations is given by

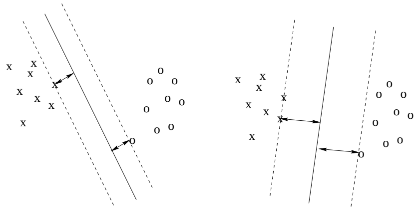
$$\hat{h}(\mathbf{x}) = \text{sign}(\,\hat{\alpha}_1 h(\mathbf{x}; \hat{\mathbf{w}}_1) + \ldots + \hat{\alpha}_m h(\mathbf{x}; \hat{\mathbf{w}}_m)\,) \qquad (25)$$

Boosting performance



- After training error reduces to zero, we can still be improving the test (generalization) error
- We can explain this in terms of the *margin*

Classification margin

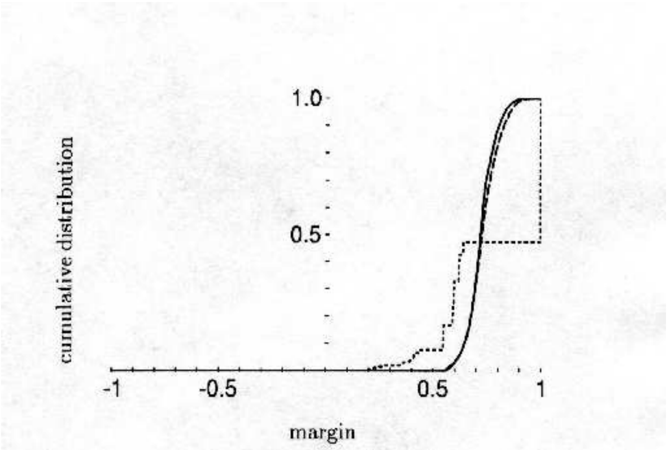


small margin large margin

- In both cases, the training error is zero
- In the linear case the margin is the orthogonal distance from the decision boundary to the closest example

- Why do we want a large margin?

Boosting and margin



$$\text{Margin}(\mathbf{x}_i) = y_i \frac{\hat{\alpha}_1 h(\mathbf{x}_i; \hat{\mathbf{w}}_1) + \ldots + \hat{\alpha}_m h(\mathbf{x}_i; \hat{\mathbf{w}}_m)}{\hat{\alpha}_1 + \ldots + \hat{\alpha}_m} \qquad (26)$$