Machine learning: lecture 10

Tommi S. Jaakkola
MIT CSAIL
tommi@csail.mit.edu

Topics

- Feature selection cont'd
  - greedy selection, regularization
- Combination of methods
  - regression: forward fitting
  - classification: decision stumps, reweighting, adaboost

Review: feature selection

- The goal here is document classification based on feature vectors (word indicators) $\Phi = [\phi_1, \ldots, \phi_m]^T$
- To improve classification performance we include only features that have high *mutual information* with the labels:

$$\hat{I}(\phi_i; y) = \sum_{\phi_i=0,1} \sum_{y=0,1} \hat{P}(\phi_i, y) \log_2 \left[\frac{\hat{P}(\phi_i, y)}{\hat{P}(\phi_i)\hat{P}(y)} \right]$$

- Many questions remain:
  - how many features do we include?
  - what about redundant features?
  - coordination among features?
  - which classifier does this type of selection benefit?

Alternative methods

- We can also try to solve the document classification task with a logistic regression model (or other discriminative classifier)

  Our prediction based on m binary word features $\Phi = [\phi_1, \ldots, \phi_m]^T$ is given by

$$P(y = 1|\Phi, \mathbf{w}) = g(\, w_0 + w_1\phi_1 + \ldots w_m\phi_m \,)$$

  where $g(\cdot)$ is the logistic function.

- There many alternative feature selection methods
  1. greedy selection (adding/pruning) of features
  2. find relevant features using regularization

Greedy selection of features

1. Find k for which

$$P(y = 1|\Phi, w_0, w_k) = g(\, w_0 + w_k \phi_k \,)$$

   leads to the best classifier

2. Find k' for which

$$P(y = 1|\Phi, w_0, w_k, w_{k'}) = g(\, w_0 + w_k \phi_k + w_{k'} \phi_{k'} \,)$$

   yields the best classifier. Here all the parameters w_0, w_k and $w_{k'}$ should be reoptimized when trying to add each k'

3. etc.

   (we need a stopping criterion)

Feature selection via regularization

$$P(y = 1|\Phi, \mathbf{w}) = g(\, w_0 + w_1\phi_1 + \ldots + w_m\phi_m \,)$$

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

$$J(\mathbf{w}; C) = \sum_{t=1}^{n} \log P(y_t|\Phi_t, \mathbf{w}) - C \sum_{i=1}^{m} |w_i|$$

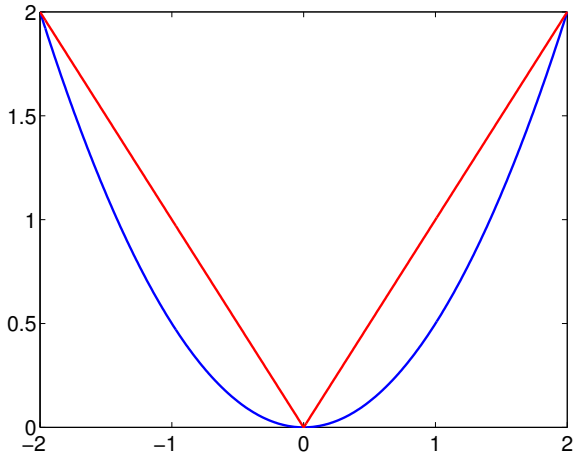
  Note that w_0 is not penalized.

- The selection of non-zero weights here is carried out *jointly*, not individually

- Why should this regularization penalty lead to a sparse solution?

Feature selection via regularization cont'd

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



$w^2/2$ versus $|w|$

$$J(\mathbf{w}; C) = \sum_{t=1}^{n} \log P(y_t|\Phi_t, \mathbf{w}) - C \sum_{i=1}^{m} |w_i|$$

- How do we deal with redundant features?

Combination of methods

- Similarly to feature selection we can select simple "weak" classification or regression methods and combine them into a single "strong" method
- Example techniques
  - forward fitting (regression)
  - boosting (classification)

Combination of regression methods

- We want to combine multiple "weak" regression methods into a single "strong" method

$$f(\mathbf{x}) = f(\mathbf{x};\theta_1) + \ldots + f(\mathbf{x};\theta_m)$$

- Suppose we are given a family simple regression methods

$$f(\mathbf{x};\theta) = w\,\phi_k(\mathbf{x})$$

  where $\theta = \{k, w\}$ specifies the identity of the basis function as well as the associated weight.

- *Forward-fitting*: sequentially introduce new simple regression methods to reduce the remaining prediction error

Forward fitting cont'd

Simple family: $f(\mathbf{x};\theta) = w\phi_k(\mathbf{x}), \;\; \theta = \{k, w\}$

- We can fit each new component to reduce the prediction error; in each iteration we solve the same type of estimation problem

$$\text{Step 1:} \qquad \hat{\theta}_1 \leftarrow \arg\min_{\theta} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i;\theta))^2$$

Forward fitting cont'd

Simple family: $f(\mathbf{x};\theta) = w\phi_k(\mathbf{x}), \;\; \theta = \{k, w\}$

- We can fit each new component to reduce the prediction error; in each iteration we solve the same type of estimation problem

Step 1: $$\hat{\theta}_1 \leftarrow \arg\min_{\theta} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i;\theta))^2$$

Step 2: $$\hat{\theta}_2 \leftarrow \arg\min_{\theta} \sum_{i=1}^{n} (\underbrace{y_i - f(\mathbf{x}_i;\hat{\theta}_1)}_{\text{error}} - f(\mathbf{x}_i;\theta))^2$$

Forward fitting cont'd

Simple family: $f(\mathbf{x};\theta) = w\phi_k(\mathbf{x}), \quad \theta = \{k, w\}$

- We can fit each new component to reduce the prediction error; in each iteration we solve the same type of estimation problem

  Step 1: $$\hat{\theta}_1 \leftarrow \arg\min_{\theta} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i;\theta))^2$$

  Step 2: $$\hat{\theta}_2 \leftarrow \arg\min_{\theta} \sum_{i=1}^{n} (\underbrace{y_i - f(\mathbf{x}_i;\hat{\theta}_1)}_{\text{error}} - f(\mathbf{x}_i;\theta))^2$$

  Step 3: $\ldots$

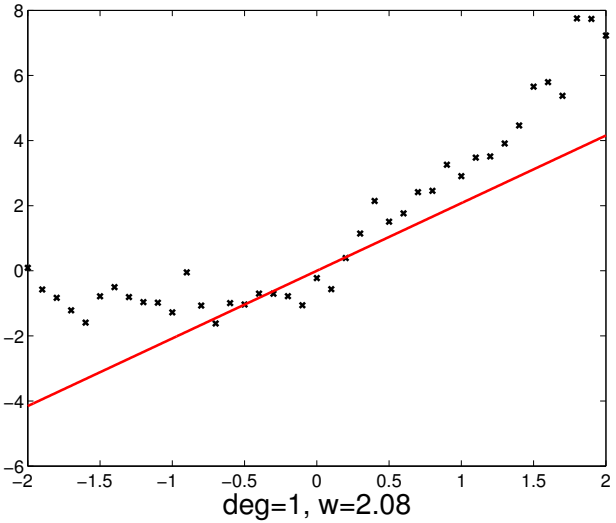
- The resulting combined regression method

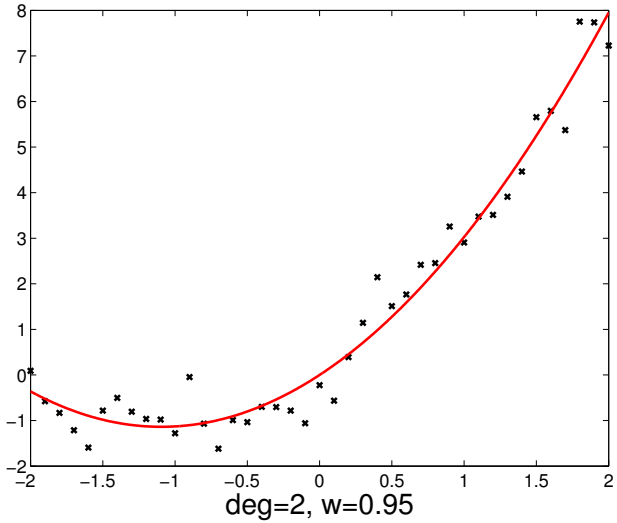
$$\hat{f}(\mathbf{x}) = f(\mathbf{x};\hat{\theta}_1) + \ldots + f(\mathbf{x};\hat{\theta}_m)$$

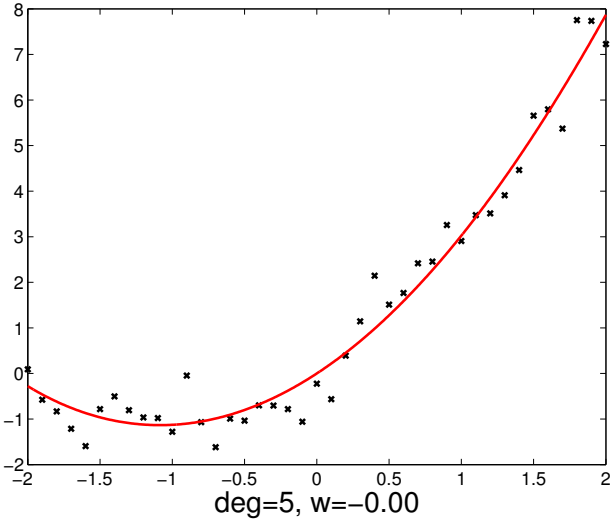
has much lower (training) error.

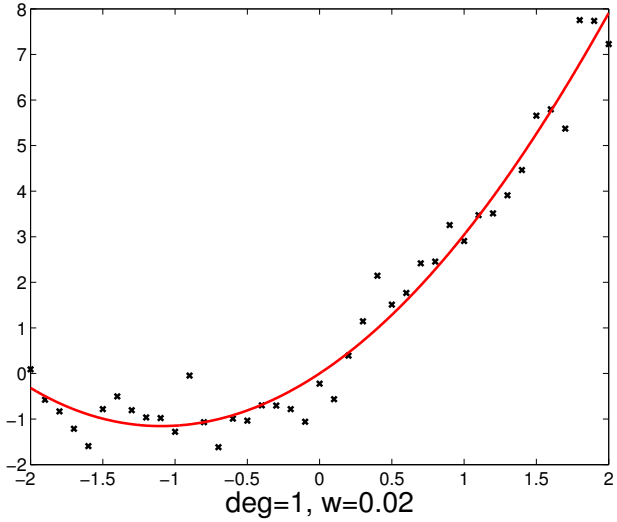
Forward fitting: example

$f(x;\theta) = wx^k$, where $\theta = \{w, k\}$.









Combination of classifiers

- We can combine simple "weak" classifiers to produce a single "strong" classifier in a manner similar to the regression models:

$$h_m(\mathbf{x}) = h(\mathbf{x};\theta_1) + \ldots + h(\mathbf{x};\theta_m)$$

  where the predicted label for $\mathbf{x}$ is the sign of $h_m(\mathbf{x})$.

- If each component classifier returns only ± 1 it is beneficial to allow some of them to have more "votes" than others:

$$h_m(\mathbf{x}) = \alpha_1\, h(\mathbf{x};\theta_1) + \ldots + \alpha_m\, h(\mathbf{x};\theta_m)$$

  where the (non-negative) votes α_i can be used to emphasize components that are more reliable than others

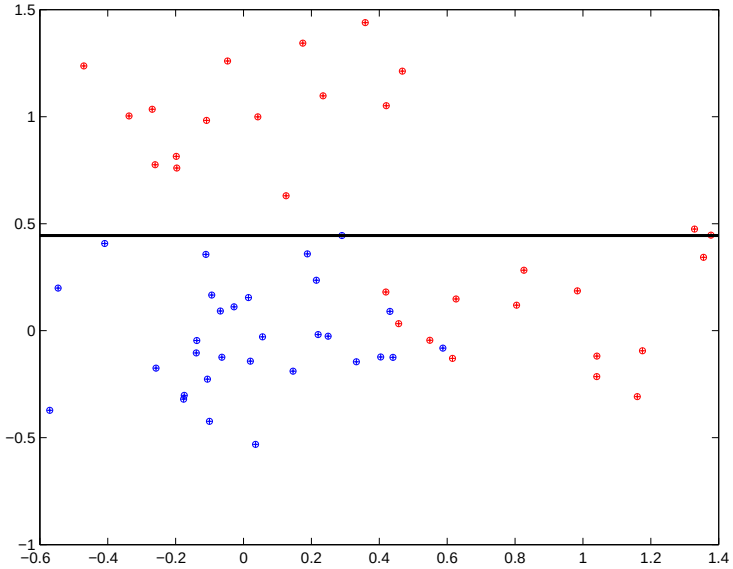
Components: decision stumps

- Consider the following simple family of component classifiers generating ± 1 labels:

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

  where $\theta = \{k, w_1, w_0\}$. These are called *decision stumps*.

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



Combination of classifiers cont'd

- We need to define a convenient loss function so as to be able to train each new component $h(\mathbf{x};\theta)$ (and the corresponding votes) in a simple modular fashion

$$h_m(\mathbf{x}) = \alpha_1 h(\mathbf{x};\theta_1) + \ldots + \alpha_m h(\mathbf{x};\theta_m)$$

- There are many options:
  - logistic (negative log-probability)

    $$-\log P(y|\mathbf{x},\theta_1,\ldots,\theta_m,\alpha_1,\ldots,\alpha_m) = -\log g(\, y\, h_m(\mathbf{x})\,)$$

    where $g(\cdot)$ is the logistic function.
  - simple exponential

    $$\exp\{\, -y\, h_m(\mathbf{x})\,\}$$

Modularity and loss

- The empirical exponential loss is "modular":

$$\sum_{i=1}^{n} \exp\{ -y_i h_m(\mathbf{x}_i) \}$$

$$= \sum_{i=1}^{n} \exp\{ -y_i h_{m-1}(\mathbf{x}_i) - y_i \alpha_m h(\mathbf{x}_i; \theta_m) \}$$

Modularity and loss

- The empirical exponential loss is "modular":

$$\sum_{i=1}^{n} \exp\{ -y_i h_m(\mathbf{x}_i) \}$$

$$= \sum_{i=1}^{n} \exp\{ -y_i h_{m-1}(\mathbf{x}_i) - y_i \alpha_m h(\mathbf{x}_i; \theta_m) \}$$

$$= \sum_{i=1}^{n} \underbrace{\exp\{ -y_i h_{m-1}(\mathbf{x}_i)\}}_{\text{fixed at stage } m} \exp\{-y_i \alpha_m h(\mathbf{x}_i; \theta_m) \}$$

Modularity and loss

- The empirical exponential loss is "modular":

$$\sum_{i=1}^{n} \exp\{ -y_i h_m(\mathbf{x}_i) \}$$

$$= \sum_{i=1}^{n} \exp\{ -y_i h_{m-1}(\mathbf{x}_i) - y_i \alpha_m h(\mathbf{x}_i; \theta_m) \}$$

$$= \sum_{i=1}^{n} \underbrace{\exp\{ -y_i h_{m-1}(\mathbf{x}_i)\}}_{\text{fixed at stage } m} \exp\{-y_i \alpha_m h(\mathbf{x}_i; \theta_m) \}$$

$$= \sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i; \theta_m) \}$$

The combined classifier based on $m-1$ iterations defines a weighted loss criterion for the next simple classifier to add

Empirical exponential loss cont'd

- We can further simplify the estimation criterion for the new component classifiers

  When $\alpha_m \approx 0$ (low confidence votes)

$$\exp\{-y_i \alpha_m h(\mathbf{x}_i; \theta_m)\} \approx 1 - y_i \alpha_m h(\mathbf{x}_i; \theta_m)$$

  and our empirical loss criterion reduces to

$$\approx \sum_{i=1}^{n} W_i^{(m-1)} \left(1 - y_i \alpha_m h(\mathbf{x}_i; \theta_m)\right) =$$

$$= \sum_{i=1}^{n} W_i^{(m-1)} - \alpha_m \left(\sum_{i=1}^{n} W_i^{(m-1)} y_i h(\mathbf{x}_i; \theta_m) \right)$$

  So we can choose each new component classifier to optimize a weighted classification error

Boosting

- A Boosting algorithm sequentially estimates and combines classifiers by reweighting training examples (concentrating on the harder examples)
  - each component classifier is presented with a slightly different problem depending on the weights
- Some preliminaries:
  - a set of "weak" binary (± 1) classifiers $h(\mathbf{x}; \theta)$ such as *decision stumps*
  - normalized weights $\tilde{W}_i^{(0)}$ on the training examples, initially set to uniform $(\tilde{W}_i^{(0)} = 1/n)$

The AdaBoost algorithm

1) At the k^{th} iteration we find (any) classifier $h(\mathbf{x}; \hat{\theta}_k)$ for which the *weighted classification error* ϵ_k

$$\epsilon_k = 0.5 - \frac{1}{2}\left(\sum_{i=1}^{n} \tilde{W}_i^{(k-1)} y_i h(\mathbf{x}_i; \hat{\theta}_k)\right)$$

is better than chance.

The AdaBoost algorithm

1) At the k^{th} iteration we find (any) classifier $h(\mathbf{x}; \hat{\theta}_k)$ for which the *weighted classification error* ϵ_k

$$\epsilon_k = 0.5 - \frac{1}{2}\left(\sum_{i=1}^{n} \tilde{W}_i^{(k-1)}\, y_i h(\mathbf{x}_i; \hat{\theta}_k)\right)$$

is better than chance.

2) The new component classifier is assigned "votes" based on its performance

$$\hat{\alpha}_k = 0.5\, \log(\, (1-\epsilon_k)/\epsilon_k\,)$$

where $\hat{\alpha}_k$ is the minimizer of the weighted loss

$$\sum_{i=1}^{n} \tilde{W}_i^{(k-1)}\, \exp\{\, -y_i \alpha_k h(\mathbf{x}_i; \hat{\theta}_k)\,\}$$

The AdaBoost algorithm

1) At the k^{th} iteration we find (any) classifier $h(\mathbf{x}; \hat{\theta}_k)$ for which the *weighted classification error* ϵ_k

$$\epsilon_k = 0.5 - \frac{1}{2}\left(\sum_{i=1}^{n} \tilde{W}_i^{(k-1)}\, y_i h(\mathbf{x}_i; \hat{\theta}_k)\right)$$

is better than chance.

2) The new component classifier is assigned "votes" based on its performance

$$\hat{\alpha}_k = 0.5\, \log(\, (1-\epsilon_k)/\epsilon_k\,)$$

3) The weights on the training examples are updated according to (c is chosen so that the new weights $\tilde{W}_i^{(k)}$ sum to one):

$$\tilde{W}_i^{(k)} = c \cdot \tilde{W}_i^{(k-1)} \cdot \exp\{\, -y_i \hat{\alpha}_k h(\mathbf{x}_i; \hat{\theta}_k)\, \}$$

Boosting: example