Announcements
Review: feature selection, mutual information

- **Wrapper** methods: try to optimize the feature subset for a given supervised learning algorithm (e.g., for a given classifier).
  - Selection by mutual information: evaluate

\[
I(X; Y) = H(X) - H(X|Y) = D_{KL} (p(X, Y) \| p(X)p(Y))
\]

- **Wrapper** methods: optimize subset of features for a given classifier/regressor.
Plan for today

- Wrapper feature selection:
  - stepwise regression.

- Boosting
Combination of regressors

• Consider linear regression model

\[ y = f(x; \mathbf{w}) \underbrace{w_0 \phi_0(x) + w_1 \phi_1(x) + \ldots + w_d \phi_d(x)}_{\equiv 1}. \]

• We can see this as a combination of \( d + 1 \) simple regressors:

\[ y = \sum_{j=0}^{d} f_j(x; \mathbf{w}), \quad f_j(x; \mathbf{w}) \triangleq w_j \phi_j(x) \]
Forward stepwise regression

\[ y = \sum_{j=0}^{d} f_j(x; w), \quad f_j(x; w) = w_j \phi_j(x) \]

- We can build this combination greedily, one function at a time.

- Parametrize the set of functions: \( f(x; \theta), \theta = [w, j] \)

- Step 1: fit the first simple model

\[ \theta_1 = \arg\min_{\theta} \sum_{i=1}^{N} (y_i - f(x_i; \theta))^2 \]
Forward stepwise regression

- **Step 1:** fit the first simple model

  \[ \theta_1 = \arg\min_{\theta} \sum_{i=1}^{N} (y_i - f(x_i; \theta))^2 \]

- **Step 2:** fit second simple model to the residuals of the first:

  \[ \theta_2 = \arg\min_{\theta} \sum_{i=1}^{N} (y_i - f(x_i; \theta_1) - f(x_i; \theta))^2 \]

- . . . **Step n:** fit a simple model to the residuals of the previous step.

- **Stop when no significant improvement in training error.**

- **Final estimate after M steps:**

  \[ \hat{y}(x) = f(x; \theta_1) + \ldots + f(x; \theta_M) \]
Stepwise regression: example

Fit $\sum_{j=1}^{k} f(x; \theta_j)$

Residuals

$\theta_j$

d = 3, $w = -0.0512$

d = 0, $w = 1.1024$
Stepwise regression: example

Fit $\sum_{j=1}^{k} f(x; \theta_j)$

Residuals

$\theta_j$

$k = 3$

$d = 5, w = 0.0002$

$k = 4$

$d = 0, w = 0.0536$
Stepwise regression for classification

- Can perform stepwise selection for any classifier of the form

\[ \hat{y}(\mathbf{x}) = f \left( \sum_{j=0}^{d} w_j \phi_j(\mathbf{x}) \right) \]

- For instance, logistic regression:
  - Step 1: \( \hat{y}(\mathbf{x}) = \text{sign} \left( \sigma(w_1 x_{j_1}^{d_1}) - 1/2 \right) \)
  - Step 2: \( \hat{y}(\mathbf{x}) = \text{sign} \left( \sigma(w_1 x_{j_1}^{d_1} + w_2 x_{j_2}^{d_2}) - 1/2 \right) \)
Combining classifiers

- A similar idea: combine classifiers $h_1(x), \ldots, h_m(x)$

\[ H(x) = \alpha_1 h_1(x) + \ldots + \alpha_m h_m(x), \]

- $\alpha_j$ is the vote assigned to classifier $h_j$.
  - Votes should be higher for more reliable classifiers.

- Prediction:

\[ \hat{y}(x) = \text{sign } H(x). \]

- Classifiers $h_j$ can be simple (e.g., based on a single feature).
Greedy assembly of classifier combination

• Consider a family of classifiers $\mathcal{H}$ parametrized by $\theta$.

• Setting $\theta_1$:
Greedy assembly of classifier combination

- Consider a family of classifiers $\mathcal{H}$ parametrized by $\theta$.

- Setting $\theta_1$: minimize the training error

  \[
  \sum_{i=1}^{N} L_{0/1}(h(x_i; \theta_1), y_i).
  \]

- How do we set $\theta_2$?
Greedy assembly of classifier combination

- Consider a family of classifiers $\mathcal{H}$ parametrized by $\theta$.

- Setting $\theta_1$: minimize the training error

$$\sum_{i=1}^{N} L_{0/1}(h(x_i; \theta_1), y_i).$$

- How do we set $\theta_2$?

- We would like to minimize the training error of the combination,

$$\sum_{i=1}^{N} L_{0/1}(H(x_i), y_i),$$

where $H(x) = \text{sign}(\alpha_1 h(x; \theta_1) + \alpha_2 h(x; \theta_2)).$
Boosting: main ideas

- Boosting algorithms maintain weights $W_i$ on the training data.
- Initially, all weights are equal, $W_i^{(0)} = 1/N$.
- In $j$-th round of boosting, the weights are updated:
  - If $x_i$ is misclassified by $h_j$, $W_i^{(j)}$ goes up;
  - If $x_i$ is classified correctly, $W_i^{(j)}$ goes down.
- Fitting of $\theta_{j+1}$ is guided by $W^{(j)}$;
  - Forced to focus on examples misclassified so far.
Decision stumps

- Axis-parallel decision stump parametrized by $\theta = \{j, s, T\}$, where $j \in \{1, \ldots, d\}$, $T \in \mathbb{R}$, and $s = \pm 1$:

$$h(x; \theta) = \text{sign} (s \cdot x_j - T).$$

- Example:
  - $j = 2, T = 4.5, s = +1$
  - $j = 1, T = 7, s = -1$
AdaBoost: demo
AdaBoost: intuition

- **Adaptive boosting**: weights are updated based on classification so far.

- Votes: assigned based on the weighted error in $j$-th iteration

\[ \alpha_j = \frac{1}{2} \log \frac{1 - \epsilon_j}{\epsilon_j}. \]

- The only requirement: $\epsilon_j < 1/2$.

- Weights: updated and normalized, so that
  - Still sum to one;
  - Under the new weights, the error of $j$-th classifier is $1/2$. 
Next time

AdaBoost;
Mixtures of experts.