6.891 Machine learning and neural networks

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

Note: Friday's recitation will be Monday (9/25) at 1pm in 36-153

Topics

- Text classification example
- Feature selection
- Bayesian estimation

Example problem

- Text classification (information retrieval)

  A large number of documents in a database

  Only a few labeled documents $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
- We wish to build a classifier on the basis of the training set (labeled documents).

  For simplicity we assume that the labels are binary (1/0)
- We need several steps
  1. Feature transformation (why?)
  2. Model specification
  3. Model estimation (and feature/model selection)

Feature transformation

- The presence/absence of specific words in a document carries information about what the document is about
- We can construct m (about 10,000) indicator features $\{\phi_k(\mathbf{x})\}$ for whether a word appears in the document

  $\phi_k(\mathbf{x}) = 1$, if word k appears in the document; otherwise zero

  $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}) \ldots \phi_m(\mathbf{x})]^T$ is the resulting feature vector
- Could we extract better features?

Model specification

- We can treat each word detector $\phi_i(\mathbf{x})$ as an independent expert
- We combine these "expert opinions" by modeling their decisions given the labels:

$$P(\phi(\mathbf{x})|y,\theta) = \left[\prod_{k=1}^{m} P(\phi_k(\mathbf{x})|y,\theta_k)\right] \quad (1)$$

where $P(\phi_k(\mathbf{x})|y,\theta_k)$ is the conditional probability that the k^{th} word appears in a document labeled y. θ_k are the parameters associated with this conditional probability.

- Classification via Bayes rule:

$$P(y|\phi(\mathbf{x})) = \frac{P(\phi(\mathbf{x})|y)P(y)}{P(\phi(\mathbf{x}))} = \frac{P(\phi(\mathbf{x})|y)P(y)}{\sum_{y'=0}^{1} P(\phi(\mathbf{x})|y')P(y')} \quad (2)$$

Model estimation

- We can write the conditional probabilities of a single feature as

$$P(\phi_k(\mathbf{x})|y,\theta_k) = \theta_{k|y}^{\phi_k(\mathbf{x})}(1-\theta_{k|y})^{1-\phi_k(\mathbf{x})} \quad (3)$$

where $\theta_{k|y}$ is the probability that the word k appears in a document labeled y and $\theta_k = \{\theta_{k|1}, \theta_{k|0}\}$.

- Maximum likelihood estimation (here for a single feature)

$$\begin{aligned} J_n(\theta_k) &= \sum_{i=1}^{n} \log P(\phi_k(\mathbf{x}_i)|y_i,\theta_k) && (4)\\ &= \sum_{i=1}^{n} \left[\phi_k(\mathbf{x}_i)\log(\theta_{k|y_i}) + (1-\phi_k(\mathbf{x}_i))\log(1-\theta_{k|y_i})\right] \\ &= \sum_{y=0}^{1} \left[N_{ky}\log(\theta_{k|y}) + (N_y - N_{ky})\log(1-\theta_{k|y})\right] && (5) \end{aligned}$$

N_{ky} = # of times word k appears in documents labeled y and N_y = # of times label y appears

Model estimation cont'd

- We get closed form maximum likelihood estimates

$$\begin{aligned} J_n(\theta_k) &= \sum_{y=0}^{1} \left[N_{ky}\log(\theta_{k|y}) + (N_y - N_{ky})\log(1-\theta_{k|y})\right] \\ \frac{\partial}{\partial\theta_{k|1}} J_n(\theta_k) &= \frac{N_{k1}}{\theta_{k|1}} - \frac{N_1 - N_{k1}}{1-\theta_{k|1}} = 0 && (6) \end{aligned}$$

implying that $\hat{\theta}_{k|1} = N_{k1}/N_1$ (interpretation?).

- **BUT**: we have very few documents and some words are rare; these estimates are unlikely to be good
- We need regularization but what prior should we use?

Priors and fictitious data

- How do we construct a prior distribution for the parameter $\theta_{k|1}$ that gives the probability of a feature given that $y = 1$?
- Answer: we generate fictitious data.

  Imagine having observed N_1' documents with labeled $y = 1$ and word k appeared in N_{k1}' of them.

  The likelihood of this data given a specific parameter value $\theta_{k|1}$ is

$$P(\text{data}|\theta_{k|1}) = \theta_{k|1}^{N_{k1}'}(1-\theta_{k|1})^{N_1' - N_{k|1}} \quad (7)$$

- We can incorporate this "prior" by treating these fictitious observations as if they were additional real observations

Prior distribution

- We can interprete the additional likelihood as a prior distribution

$$
\begin{aligned}
J_n'(\theta_k) &= \log P(\text{ real data } |\theta_k) + \log P(\text{ data } |\theta_k) && (8)\\
&= \log P(\text{ real data } |\theta_k) + \log P(\,\theta_k\,) + \text{const.} && (9)
\end{aligned}
$$

where the prior distribution $P(\,\theta_k\,)$ is obtained by normalizing the likelihood of the fictitious data

$$
P(\,\theta_k\,) \propto P(\text{ data } |\theta_k) \tag{10}
$$

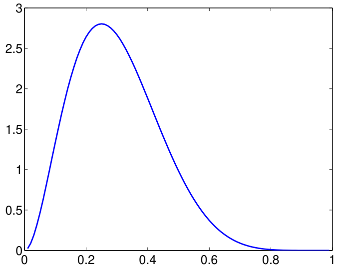
Beta distribution

- Beta distribution as a prior

$$
P(\theta_{k1}) = c_{k1} \cdot \theta_{k1}^{N'_{k1}} (1-\theta_{k1})^{N'_1 - N'_{k1}} \tag{11}
$$

where c_{k1} is the normalization constant. The parameters N'_{k1}, N'_1 of the prior are known as *hyper-parameters* (typically denoted as α_1 and α_0)

- Equivalent sample size interpretation



$N'_1 = 6$, $N'_{k1} = 2$

Model estimation cont'd

- Combining the log-likelihood with the log-prior probability we get

$$
\hat{\theta}_{k|1} = \frac{N_{k1} + N'_{k1}}{N_1 + N'_1} \tag{12}
$$

(you should be able to derive these as the maximum penalized likelihood estimates)

Similarly we get all the remaining parameters $\hat{\theta}_k = \{\hat{\theta}_{k|1}, \hat{\theta}_{k|0}\}$, $k = 1, \ldots, m$, for our probability model

$$
P(\phi(\mathbf{x})|y,\theta) = \left[\prod_{k=1}^{m} P(\phi_k(\mathbf{x})|y,\theta_k) \right] \tag{13}
$$

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

$$\begin{aligned} \phi_k &= 0,1 \quad \text{whether } k^{th} \text{ word is present in a document} \\ y &= 0,1 \quad \text{document label} \end{aligned} \tag{14}$$

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)

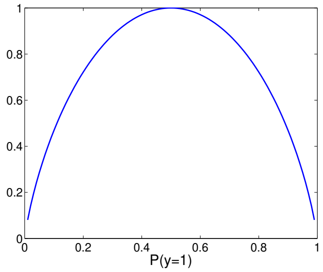
- We pick only features that provide substantial information about the labels, i.e., those with high *mutual information* with the labels:

$$I(\phi_k; y) = \sum_{\phi,y=0,1} P(\phi_k, y) \log_2 \frac{P(\phi_k, y)}{P(\phi_k)P(y)} \tag{15}$$

Background

- Entropy (uncertainty) of a binary random variable y

$$H(y) = -\sum_{y=0,1} P(y) \log_2 P(y) \tag{16}$$



Why Shannon entropy?
1010110101010001110110100011010101...

Background cont'd

- Properties of mutual information:

$$I(\phi_k; y) = \sum_{\phi,y=0,1} P(\phi_k, y) \log_2 \frac{P(\phi_k, y)}{P(\phi_k)P(y)} \tag{17}$$

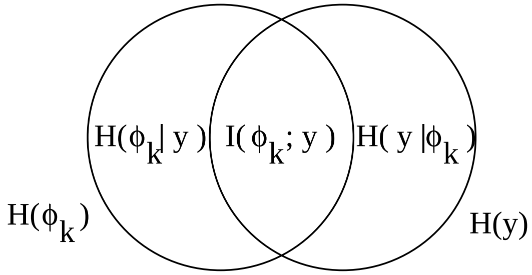
1. $I(\phi_k; y) = I(y; \phi_k)$ (symmetry)
2. If ϕ_k and y are independent, $I(\phi_k; y) = 0$
3. $I(\phi_k; y) \le H(y)$, $I(\phi_k; y) \le H(\phi_k)$
4. $I(\phi_k; y) = H(y) - H(y|\phi_k) = H(\phi_k) - H(\phi_k|y)$

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

$$H(y|\phi_k) = \sum_{\phi_k=0,1} P(\phi_k) \left[-\sum_{y=0,1} P(y|\phi_k) \log_2 P(y|\phi_k) \right] \tag{18}$$

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

$$\begin{aligned} \phi_k &= 0,1 \quad \text{whether } k^{th} \text{ word is present in a document} \\ y &= 0,1 \quad \text{document label} \end{aligned} \tag{19}$$

  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:

$$I(\phi_k; y) = \sum_{\phi, y=0,1} P(\phi_k, y) \log_2 \frac{P(\phi_k, y)}{P(\phi_k)P(y)} \tag{20}$$

- What approximations are we making here?

Filtering approach

- 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

- 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

$$\hat{P}(\,\text{error}\,|\Phi, \mathbf{w}) > \hat{P}(\,\text{error}\,|\Phi, \phi_k, \mathbf{w}') \tag{21}$$

  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
- Disadvantage: speed

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

$$P(y=1|\mathbf{x}, \mathbf{w}) = g(\, w_0 + \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})\,) \tag{22}$$

- 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

$$P(y = 1|\mathbf{x}, \mathbf{w}) = g(\, w_0 + w_k \phi_k(\mathbf{x})\,) \tag{23}$$

yields the best classifier

2. Find k' for which

$$P(y = 1|\mathbf{x}, \mathbf{w}) = g(\, w_0 + w_k \phi_k(\mathbf{x}) + w_{k'} \phi_{k'}(\mathbf{x})\,) \tag{24}$$

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

- When do we stop?
- Cross-validation

Wrapper example: regularization

$$P(y = 1|\mathbf{x}, \mathbf{w}) = g(\, w_0 + \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})\,) \tag{25}$$

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

$$J(\mathbf{w}; C) = \sum_{t=1}^{n} \log P(y_t|\mathbf{x}_t, \mathbf{w}) - C \sum_i |w_i| \tag{26}$$

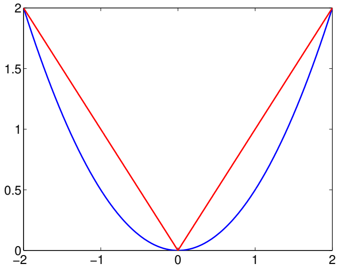
where $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ is our training set.

- 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)

Where we are

- Text classification example
- Feature selection
- *Bayesian estimation*

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" weighting of different settings comes directly from probability calculus:

$$P(\mathbf{w}|\text{data}) \propto P(\text{data}|\mathbf{w})P(\mathbf{w}) \tag{27}$$

  where, for example, in the logistic regression case

$$P(\text{data}|\mathbf{w}) = \prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w}), \quad P(\mathbf{w}) = N(\mathbf{w}; \mu, \Sigma) \tag{28}$$

Bayesian estimation

- Given the *posterior* weighting of the parameters

$$P(\mathbf{w}|\text{data}) \propto P(\text{data}|\mathbf{w})P(\mathbf{w}) \tag{29}$$

  we would classify any new example on the basis of

$$P(y|\mathbf{x}, \text{data}) = \int P(y|\mathbf{x}, \mathbf{w})\, P(\mathbf{w}|\text{data})\, d\mathbf{w} \tag{30}$$

- Overfitting?
- Problems?