(1-12-2016)

Statistical Learning IV

1. Discriminative vs Generative Classification
   - Logistic regression vs Naive Bayes
2. Stochastic Optimization (see handout)
3. K-means
4. EM for Gaussian mixtures

} unsupervised learning

(Studying 4. is optional.)

3. Discriminative vs Generative Methods

Probabilistic model: $\mathcal{P} = \{ P_\beta(x,y) \mid \beta \in R^p \}$

Generative

(imagine data generated by first choosing class y, then features x given y.)
e.g. spam

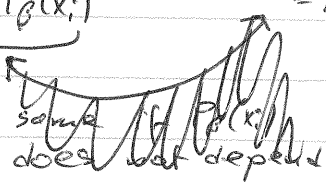
estimate $P(x,y)$

LDA, Naive Bayes, ...

overkill for classification

$$\underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} -\log P_\beta(x_i, y_i)$$

$$= \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} -\log P_\beta(y_i \mid x_i) + \sum_{i=1}^{N} -\log P_\beta(x_i)$$



acts as regularizer, so reduces variance $\Rightarrow$ better for small N

Discriminative

estimate $P(y \mid x)$

Logistic regression, ...

less overkill, closer to class boundary, which is only thing we are interested in

$$\underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} -\log P_\beta(y_i \mid x_i)$$

$$= \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} -\log P_\beta(y_i \mid x_i) + \sum_{i=1}^{N} -\log P^*(x_i)$$

P^*: true distribution of x

better fit of true distribution for large N if $P^* \notin \mathcal{P}$

Two methods are generative-discriminative pair if one estimates $P_\beta(x,y)$ and other $P_\beta(y \mid x)$ in same model $\mathcal{P}$.

E.g.: LDA - logistic regression for continuous features x_j;
Naive Bayes - logistic regression for binary features

Logistic Regression vs Naive Bayes

LR: $\log \frac{\hat{P}(y=1|X)}{\hat{P}(y=-1|X)} = X^T\hat{\beta} + \hat{\beta}_0$ (writing intercept separately)

$$\Updownarrow$$

$$\hat{P}(y=1|X) = \frac{e^{X^T\hat{\beta}+\hat{\beta}_0}}{1+e^{X^T\hat{\beta}+\hat{\beta}_0}}$$

Will show that

(*) $\log \frac{\hat{P}(y=1|X)}{\hat{P}(y=-1|X)} = X^T\hat{\alpha} + \hat{\alpha}_0$ for Naive Bayes

($\hat{\alpha}$, $\hat{\alpha}_0$ determined by NB parameters $\hat{\theta}_{1j}, \hat{\theta}_{-1j}, \hat{\pi}_1, \hat{\pi}_{-1}$)

Hence Naive Bayes is working with a probabilistic model $\mathcal{P} = \{P_\alpha(X,y) \mid \alpha \in \mathbb{R}^p\}$ such that $P_\alpha(y|X)$ is the same as in the model logistic regression is using.

So NB-LR form a generative-discriminative pair

Therefore expect NB better for smaller N, LR better for larger N.

NG, Jordan, NIPS 2001 make this precise:

* Asymptotic performance as $N \to \infty$: LR always better than NB
* But LR needs $N \geq O(p)$ to reach asymptotic performance

whereas (under some technical assumptions) NB needs $N \geq O(\log p)$ to reach asymptotic performance

To show (*):

$$\log \frac{\hat{P}(y=1|x)}{\hat{P}(y=-1|x)} = \log \frac{\hat{P}(x|y=1)\hat{P}(y=1)}{\hat{P}(x|y=-1)\hat{P}(y=-1)} = \log \frac{\hat{f}_1(x)\hat{\pi}_1}{\hat{f}_{-1}(x)\hat{\pi}_{-1}}$$

$$x = \begin{pmatrix} 0 \leftarrow \text{UNIVERSITY} \\ \vdots \\ 1 \leftarrow \text{BIG} \\ 2 \leftarrow \text{MONEY} \end{pmatrix} = \begin{pmatrix} n_1 \\ \vdots \\ n_p \end{pmatrix} \qquad f_k(x) = \prod_{j=1}^{p} \theta_{k,j}^{n_j}$$

$$\log \frac{\hat{f}_1(x)\hat{\pi}_1}{\hat{f}_{-1}(x)\hat{\pi}_{-1}} = \log \frac{\prod_{j=1}^{p} \hat{\theta}_{1,j}^{n_j}}{\prod_{j=1}^{p} \hat{\theta}_{-1,j}^{n_j}} + \log \frac{\hat{\pi}_1}{\hat{\pi}_{-1}}$$

$$= \log \prod_{j=1}^{p} \left(\frac{\hat{\theta}_{1,j}}{\hat{\theta}_{-1,j}} \right)^{n_j} + \log \frac{\hat{\pi}_1}{\hat{\pi}_{-1}}$$

$$= \sum_{j=1}^{p} n_j \log \frac{\hat{\theta}_{1,j}}{\hat{\theta}_{-1,j}} + \log \frac{\hat{\pi}_1}{\hat{\pi}_{-1}}$$

$$= x^T \hat{\alpha} + \hat{\alpha}_0 \quad \text{for } \hat{\alpha} = \begin{pmatrix} \log \frac{\hat{\theta}_{1,1}}{\hat{\theta}_{-1,1}} \\ \vdots \\ \log \frac{\hat{\theta}_{1,p}}{\hat{\theta}_{-1,p}} \end{pmatrix} \qquad \hat{\alpha}_0 = \log \frac{\hat{\pi}_1}{\hat{\pi}_{-1}}$$

Possible to choose $\hat{\theta}_{k,j}$, $\hat{\pi}_k$ such that $\hat{\alpha} = \hat{\beta}$, $\hat{\alpha}_0 = \hat{\beta}_0$, but NB does not do that $\forall_0$

Unsupervised Learning

Intro

Supervised: $T = \begin{pmatrix} y_1 \\ x_1 \end{pmatrix}, \ldots, \begin{pmatrix} y_N \\ x_N \end{pmatrix}$ ← can evaluate every method by EPE

Unsupervised: $T = x_1, \ldots, x_N$

What can we do?

Clustering: split data into groups of points (clusters) that are similar

1. What do you mean by "similar"?
2. What kind of clusters are you looking for?

K-means

(1) Squared Euclidean distance between $x_i, x_{i'}$:
$$d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

(2) Find K clusters s.t. distance to mean within each cluster is small.

$C(i) \in \{1, \dots, K\}$ is cluster assigned to x_i.

$$\text{minimize} \sum_{k=1}^{K} \sum_{i:C(i)=k} d(x_i, \mu_k) \qquad (*)$$

where μ_k = mean of points in cluster k

NB. Mistake in book! (multiplies by N_k in each cluster)

Algorithm:
1. Initialize cluster assignment C
2. Set μ_k to current mean in cluster k
3. Given means $\mu_1, \dots, \mu_K$, assign each x_i to nearest mean to get new C
4. repeat from 2 until no changes in C

- Converges to local minimum
- May choose K s.t. increasing K does not reduce (*) very much.

Gaussian Mixtures and EM

- like k-means, but with "soft" cluster assignments (each cluster is a Gaussian modelled by)
- for simplicity: explain for two clusters

cluster 1: $X \sim N(\mu_1, \sigma_1^2)$ w.p. $1-\pi$
cluster 2: $X \sim N(\mu_2, \sigma_1^2)$ w.p. π

probability density: $(1-\pi)\phi_{\mu_1,\sigma_1^2}(x) + \pi\phi_{\mu_2,\sigma_2}(x)$

To find parameters:

ML is difficult numerically

+ gives bad solution with $\hat{\sigma}_1 = 0$

$\hat{\mu}_1 = x_i$ for some i.

Solution: algorithm very similar to K-means

Hidden variables $\Delta_i = \begin{cases} 0 & \text{if } x_i \text{ in cluster 1} \\ 1 & \text{if } x_i \text{ in cluster 2} \end{cases}$

1. Initializes parameters $\underbrace{\hat{\pi}, \hat{\mu}_1, \hat{\sigma}_1, \hat{\mu}_2, \hat{\sigma}_2}_{\hat{\theta}}$, $(\hat{\sigma}_1 > 0.5, \hat{\sigma}_2 > 0.5)$

2. Soft cluster assignments:

$$\hat{\gamma}_i = \mathbb{E}[\Delta_i | \hat{\theta}, T] = \frac{\hat{\pi}\varphi_{\hat{\mu}_2, \hat{\sigma}_2}(x_i)}{\hat{\pi}\varphi_{\hat{\mu}_2, \hat{\sigma}_2}(x_i) + (1-\hat{\pi})\varphi_{\hat{\mu}_1, \hat{\sigma}_1}(x_i)}$$

3. Update parameter estimates for clusters:

$$\hat{\mu}_1 = \frac{\sum_{i=1}^{N}(1-\hat{\gamma}_i)x_i}{\sum_i (1-\hat{\gamma}_i)} \qquad \hat{\mu}_2 = \frac{\sum_{i=1}^{N}\hat{\gamma}_i x_i}{\sum_i \hat{\gamma}_i}$$

$$\hat{\sigma}_1^2 = \frac{\sum_{i=1}^{N}(1-\hat{\gamma}_i)(x_i - \hat{\mu}_1)^2}{\sum_i (1-\hat{\gamma}_i)} \qquad \hat{\sigma}_2^2 = \frac{\sum_{i=1}^{N}\hat{\gamma}_i(x_i - \hat{\mu}_2)^2}{\sum_i \hat{\gamma}_i}$$

$$\hat{\pi} = \frac{\sum_{i=1}^{N}\hat{\gamma}_i}{\sum_{i=1}^{N}\hat{\gamma}_i + \sum_{i=1}^{N}(1-\hat{\gamma}_i)} = \frac{\sum_{i=1}^{N}\hat{\gamma}_i}{N}$$

4. repeat from 2 until convergence

- converges to local minimum.