0. Big Picture

Log(arithmic) loss:

* my predictions are probability distributions P for Y

* $L(Y, P) = -\log P(Y) \in [0, \infty]$
  - small if I gave large probability to Y
  - large if I gave small probability to Y.

Maximum Likelihood = ERM for log loss

$$\text{ML: } \operatorname*{argmax}_\beta \prod_{i=1}^N P_\beta(Y_i | X_i) = \operatorname*{argmin}_\beta \sum_{i=1}^N -\log P_\beta(Y_i | X_i)$$

$$= \operatorname*{argmin}_\beta \sum_{i=1}^N L(Y_i, P_\beta(\cdot | X_i))$$

↑ loss when learning is log loss in big picture.

Statistical Learning V (26-11-2015)

0. Big Picture
1. Surrogate losses
2. Logistic regression
3. Discriminative vs generative methods
4. Unsupervised learning
5. Clustering: K-means, EM with Gaussian mixtures

1. Surrogate losses

Classification: find $\hat{f}$ with small expected 0/1-loss

$$EPE(\hat{f}) = \underset{X,Y}{\mathbb{E}}[L(Y, \hat{f}(X))]$$

L: 0 if $\hat{f}(X)$ predicts Y correctly
1 if $\hat{f}(X)$ predicts Y incorrectly

Empirical Risk Minimization:

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i))$$

Suppose $\mathcal{F}$ set of all linear functions: $\{f(x) = x^T\beta \mid \beta \in \mathbb{R}^p\}$

$y \in \{-1, +1\}$ $\quad L(y, f(x)) = \begin{cases} 0 & \text{if } \operatorname{sign}(f(x)) = y \\ 1 & \text{if } \operatorname{sign}(f(x)) \neq y \end{cases}$

Difficulties:
- ERM solution not unique for linearly separable data
- computational problems for non-separable data

Surrogate/proxy loss L':
Care about loss L, but train classifier with loss L'
Want: L large $\Leftrightarrow$ L' large
L small $\Leftarrow$ L' small

Usually: surrogate L' convex in parameters, so can optimize efficiently

Two most important surrogates: logistic loss, hinge loss
↑ today ↑ next week

Predict correctly iff $\text{sign}(x^T\hat{\beta}) = y \Rightarrow y \cdot x^T\beta$ larger is better

(see Figure of surrogate losses)

2. Logistic Regression

(How can we use the power of linear regression for classification?)

Assume 2 classes for simplicity.

Bad idea: $\hat{P}(y=1|x) = x^T\hat{\beta}$ ← least squares for classification

E.g. y_i is going to the beach on day i
x_{i3} is temperature in °C
$\hat{\beta}_3 = \frac{1}{5}$
Then $+10\,°C \Rightarrow x^T\hat{\beta} + 2 \Rightarrow \hat{P}(y=1|x) + 2 > 1$ ↯

Probabilities do not behave linearly.

Better: $\log P_\beta(Y=1|x) = x^T\beta \iff P_\beta(y=1|x) = e^{x^T\beta}$

$x^T\hat{\beta}+2 \Rightarrow P_{\hat{\beta}}(y=1|x)$ increases by factor e^2, but can still go outside $[0,1]$

Logistic Regression:

log odds is linear

$$\log \frac{P_\beta(Y=1|x)}{P_\beta(Y=-1|x)} = x^T\beta \iff \begin{cases} P_\beta(y=1|x) = \frac{e^{x^T\beta}}{1+e^{x^T\beta}} = \frac{1}{1+e^{-x^T\beta}} \\ P_\beta(y=-1|x) = \frac{1}{1+e^{x^T\beta}} \end{cases}$$

$$P_\beta(y|x) = \frac{1}{1+e^{-yx^T\beta}}$$

$P_{\hat{\beta}}(Y=1|x) > \frac{1}{2} \iff x^T\hat{\beta} > 0 \Rightarrow$ decision boundary: x s.t. $x^T\hat{\beta} = 0$ is linear

logistic loss = log loss for logistic regression:

$$L(y,x^T\beta) = -\log P_\beta(y|x) = \log(1+e^{-yx^T\beta})$$

Train with maximum likelihood = ERM for logistic loss

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{N} \log(1+e^{-y_i x_i^T\beta})$$

No formula for minimum, but can find it by convex optimization

Important extensions:

$$\text{minimize} \sum_{i=1}^{N} \log(1+e^{-y_i x_i^T\beta}) + \lambda \text{pen}(\beta)$$

L_2-penalty: $\text{pen}(\beta) = \sum_{j=2}^{p} \beta_j^2$ (like ridge regression)

L_1-penalty: $\text{pen}(\beta) = \sum_{j=2}^{p} |\beta_j|$ (like lasso)

Considerations for penalization similar as for squared error.

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

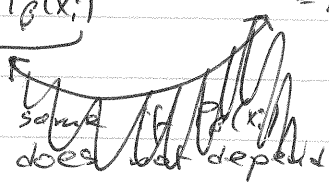
estimate $P(x, y)$

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 LDA

$x_{ij} \in \mathbb{R}$ (continuous features)

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

LDA: linear decision boundary $= \{x : \hat{P}(y=1|x) = \hat{P}(y=-1|x)\}$

$= \{x : \log \frac{\hat{P}(y=1|x)}{\hat{P}(y=0|x)} = 0\}$

in fact, $\log \frac{\hat{P}(y=1|x)}{\hat{P}(y=-1|x)} = x^T\hat{\alpha} + \hat{\alpha}_0$ is always linear in x, not just for x s.t. it is 0.

parameters $\hat{\alpha}, \hat{\alpha}_0$ are a function of LDA parameters $\hat{\mu}_k, \hat{\Sigma}, \hat{\pi}_k$ ← possible to choose s.t. $\hat{\alpha} = \hat{\beta}$, $\hat{\alpha}_0 = \hat{\beta}_0$, but LDA does not do that.

so $\hat{P}(y=1|x) = \frac{e^{x^T\hat{\alpha}+\hat{\alpha}_0}}{1+e^{x^T\hat{\alpha}+\hat{\alpha}_0}}$ ← has same form as for logistic regression.

LDA and LR are generative-discriminative pair

↳ expect LDA to be better for smaller N, LR for larger N. (unless LDA model is correct!)

Efron, 1975: if LDA model correct, then LR will perform as well as LDA if LR has 30% more data, so cannot be a lot worse.

Logistic Regression vs Naive Bayes

$x_{ij} \in \{0,1\}$ (binary features)
$j = 1, \dots, p$

↳ transform into $x_i = \begin{pmatrix} n_0 \\ n_1 \end{pmatrix}$ where n_{ce} is nr of a in i-th feature vector x_i

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 (NB)

($\hat{\alpha}$, $\hat{\alpha}_0$ determined by NB parameters for multinomial models $\hat{\theta}^k_1, \dots \hat{\theta}^k_m$ and by $\hat{\pi}_k$)

Hence NB - LR form generative-discriminative pair.

So expect NB better for small N, LR better for large N.

Ng, Jordan, NIPS 2001 make this precise

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

(under some technical assumptions)

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

f_k: multinomial model on binary features = Bernoulli model

simplify parametrization: $\hat{\theta}_k$ is prob $x_{ij}=1$ for class k
$1-\hat{\theta}_k$ is prob $x_{ij}=0$

$$\log \frac{\hat{f}_1(x)\hat{\pi}_1}{\hat{f}_{-1}(x)\hat{\pi}_{-1}} = \log \frac{\hat{\theta}_1^{n_1}(1-\hat{\theta}_1)^{n_0}}{\hat{\theta}_{-1}^{n_1}(1-\hat{\theta}_{-1})^{n_0}} + \log \frac{\hat{\pi}_1}{\hat{\pi}_{-1}}$$

$$= n_0 \cdot \log \frac{1-\hat{\theta}_1}{1-\hat{\theta}_{-1}} + n_1 \cdot \log \frac{\hat{\theta}_1}{\hat{\theta}_{-1}} + \log \frac{\hat{\pi}_1}{\hat{\pi}_{-1}}$$

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

possible to choose $\hat{\theta}_1, \hat{\theta}_{-1}, \hat{\pi}_1, \hat{\pi}_{-1}$ s.t. $\hat{\alpha} = \hat{\beta}$, $\hat{\alpha}_0 = \hat{\beta}_0$,

but NB does not do that.

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.