Plug-in Classifiers or Generative Classifiers

1. Assume a parametric form for the densities (e.g., Gaussian conditioned on each class), estimate the parameters and plug them in the Bayesian classifier
   a. QDA: $g_i(\underline{x}) = -\frac{1}{2}ln|\hat{\Sigma}_i| - \frac{1}{2}(\underline{x}-\underline{\hat{\mu}}_i)^T\hat{\Sigma}_i^{-1}(\underline{x}-\underline{\mu}_i) + ln\,\hat{\pi}_i = \underline{x}^T A\underline{x} + \underline{b}^T\underline{x} + c$
   b. LDA: Hyper ellipsoid case: $g_i(\underline{x}) = \underline{\hat{\mu}}_i^T\hat{\Sigma}^{-1}\underline{x} - [\frac{1}{2}\underline{\hat{\mu}}_i^T\hat{\Sigma}^{-1}\underline{\hat{\mu}}_i - \hat{\pi}_i] = \underline{w}^T\underline{x} - w_0$
   c. Simplified LDA: $g_i(\underline{x}) = [\frac{1}{\hat{\sigma}^2}\underline{\hat{\mu}}_i^T\underline{x} - (\frac{1}{2\hat{\sigma}^2}\underline{\hat{\mu}}_i^T\underline{\hat{\mu}}_i - \hat{\pi}_i)] = \underline{w}^T\underline{x} - w_0$
2. Estimate densities (Non-parametric methods)
   a. Hypercubes: Parzen
   b. Gaussian windows: PNN: sum of multivariate Gaussian distributions centered at each training sample for each class.

      $$\hat{p}(\underline{x}\,|\,z=i) = \frac{1}{(2\pi)^{p/2}\sigma_i^p}\frac{1}{n_i}\sum_{j=1}^{n_i}\exp\left\{-\frac{(\underline{x}-\underline{x}_i^j)^T(\underline{x}-\underline{x}_i^j)}{2\sigma_i^2}\right\}; i=1,2,..,C$$

      Use these to compute $P(z=i|\underline{x})$ via Bayes rule. Nice thing is, you can use costs of misclassifications, etc. and make Bayesian decisions. σ is a hyper-parameter. Discuss how you tune hyper-parameters.
      If you group (cluster) the data of each class prior to approximation (e.g., K-means, GMM, LVQ), you can reduce the storage and you get RBF networks.
   c. kNN classifier: easy to implement and the one you should always try! *k* is a hyper-parameter. *k* =1 is called the nearest neighbor classifier. Relation to computational geometry.
      i. Store data (features and labels)
      ii. Given $\underline{x}$, find *k* nearest neighbors
      iii. Pick class with the largest number among the k neighbors

      Discuss what you mean by nearest neighbor: Euclidean, Hamming, Holder's p-norm, Cosine (used in text processing, independent of magnitude),

      $$\|\underline{x}-\underline{x}^i\|_p = \left[\sum_{j=1}^{n}(x_j - x_j^i)^p\right]^{1/p}$$

      $$c(\underline{x},\underline{x}^i) = \frac{\underline{x}^T\underline{x}^i}{\|\underline{x}\|_2\|\underline{x}^i\|_2}$$

      Euclidean: normalize the data first, not good for high-dimensions! The ratio between the nearest and farthest points approaches 1, i.e., the points essentially become uniformly distant from each other. For higher dimensions use lower p! Important to reduce dimensions via PCA, for example.
      Hamming distance is the number of values that are different between two vectors (used to compare strings of equal length).
      Jaccard index: to compare sets d(A,B)= 1-[|(A AND B)|/|(A OR B)|]; Sorensen-Dice Index: d(A,B)= 1-[2|(A AND B)|/(|A| +| B|];

Haversine and Vincenty Distances: Distance between two points on a sphere or ellipsoid, respectively

Discriminative Classifiers: Why not estimate the parameters directly?

Recall: you can set one of the discriminants to any arbitrary value, say 0!

$$P(z=k\,|\,\underline{x}) = \frac{e^{y(\underline{x},\underline{w}_k)}}{\sum_{i=1}^{C} e^{y(\underline{x},\underline{w}_i)}} = \frac{e^{y(\underline{x},\underline{w}_k)-y(\underline{x},\underline{w}_C)}}{1+\sum_{i=1}^{C-1} e^{y(\underline{x},\underline{w}_i)-y(\underline{x},\underline{w}_C)}};\; y(\underline{x},\underline{w}_k) = g_k(\underline{x})$$

$$Binary\ case, LDA: P(z=1\,|\,\underline{x}) = \frac{1}{1+e^{-[y(\underline{x},\underline{w}_1)-y(\underline{x},\underline{w}_0)]}} = \frac{1}{1+\exp(-\underline{w}^T\underline{x}_a)} = \frac{1}{1+\exp(-y)} = \sigma(y);\; y=\underline{w}^T\underline{x}_a$$

Leads to a popular classifier called logistic regression!

$$P(z\,|\,\underline{x},\underline{w}) = \left(\frac{1}{1+e^{-y}}\right)^{z}\left(\frac{1}{1+e^{y}}\right)^{1-z} = \frac{1}{1+e^{-(2z-1)y}} = e^{yz}\sigma(-y)$$

$$\ln P(z\,|\,\underline{x},\underline{w}) = yz + \ln\sigma(-y) = yz - \xi(y) = yz - \ln(1+\exp(y))$$

$$MAP: D = \left\{\{\underline{x}^1, z^1\}\ \{\underline{x}^2, z^2\},\cdots\cdots,\{\underline{x}^N, z^N\}:\ z^n\ \in\{0,1\}\right\}$$

$$L(\underline{w}) = P(\{z^n\}_{n=1}^N\,|\,\{\underline{x}^n\}_{n=1}^N,\underline{w}) = \prod_{n=1}^{N} P(z^n\,|\underline{x}^n,\underline{w}) = \prod_{n=1}^{N}(\sigma(y_n))^{z^n}(1-\sigma(y_n))^{1-z^n}$$

$$J(\underline{w}) = -\ln L(\underline{w}) = NLL(\underline{w}) = -\sum_{n=1}^{N}[\underbrace{z^n\ln\sigma(y_n)+(1-z^n)\ln(1-\sigma(y_n))]}_{J_n}\ .....Cross\ Entropy$$

$$=\sum_{n=1}^{N}[z^n\varsigma(-y_n)+(1-z^n)\varsigma(y_n)] = \sum_{n=1}^{N}[z^n\ln(1+\exp(-y_n))+(1-z^n)\ln(1+\exp(y_n))]$$

$$\nabla_{\underline{w}}J = -\sum_{n=1}^{N}\underbrace{\frac{\partial J_n}{\partial\sigma(y_n)}}_{\frac{z^n-\sigma(y_n)}{\sigma(y_n)(1-\sigma(y_n))}}\ \underbrace{\frac{\partial\sigma(y_n)}{\partial y_n}}_{\sigma(y_n)(1-\sigma(y_n))}\ \underbrace{\nabla_{\underline{w}}y_n}_{\underline{x}^n} = \sum_{n=1}^{N}\left(\sigma(y_n)-z^n\right)\underline{x}^n = \underbrace{X^T}_{(p+1)\mathrm{x}N}\ \underbrace{(\underline{\sigma}-\underline{z})}_{N\mathrm{x}1}$$

$$\nabla^2_{\underline{w}}J = X^TDX > 0\ where\ D = Diag[\sigma(y_n)(1-\sigma(y_n))]$$

$$\mathrm{Re}\,gularize\ by\ adding\ \lambda\underline{w}^T\underline{w}\ or\ \lambda\|\underline{w}\|_1\ to\ J(\underline{w})$$

$$\begin{aligned}
\underline{w}_{i+1} &= \underline{w}_i - \left(\nabla^2_{\underline{w}_i}J\right)^{-1}\nabla_{\underline{w}_i}J \\
&= \underline{w}_i + \left(X^TD_iX\right)^{-1}X^T(\underline{z}-\underline{\sigma}_i) \\
&= \left(X^TD_iX\right)^{-1}\left[X^TD_iX\,\underline{w}_i + X^T(\underline{z}-\underline{\sigma}_i)\right] \\
&= \left(X^TD_iX\right)^{-1}X^TD_i\underbrace{\left[X\,\underline{w}_i + D_i^{-1}(\underline{z}-\underline{\sigma}_i)\right]}_{\underline{y}_{new,i}}
\end{aligned}$$

$$y_{new,i} = \underline{w}_i^T\underline{x}_a^n + \frac{z^n-\sigma(\underline{w}_i^T\underline{x}_a^n)}{\sigma(\underline{w}_i^T\underline{x}_a^n)(1-\sigma(\underline{w}_i^T\underline{x}_a^n))};\ z^n=1\Rightarrow increase\ by\ \frac{1}{\sigma(\underline{w}_i^T\underline{x}_a^n)};\ z^n=0\Rightarrow decrease\ by\ \frac{1}{(1-\sigma(\underline{w}_i^T\underline{x}_a^n))}$$

Iterative Reweighted Least Squares …. Can be implemented in a recursive way over data!

$$\min\ J_i = \frac{1}{2}\left(\underline{y}_{new,i} - X\,\underline{w}\right)^T D_i \left(\underline{y}_i - X\,\underline{w}\right)$$

$D_i =$ inverse of covraiance of "measurement error"

Multi-class case:

- Convert the problem to one versus rest problem (one-hot coding)
- Multiclass problems
    - M class problem: M two-class tasks or M(M-1)/2 two-class tasks
    - Obtain a separate classifier for each pair
    - What if more than one decision rule classifies? Majority rule
- Error Correcting Output Codes
    - Each class: an n bit pattern chosen so that Hamming distance is as large as possible among classes
    - Learn each of the n bits via binary classifiers
    - Select nearest class (i.e., one with the least Hamming distance)
- Work Directly with Multi-class formulation
    - Direct extension of the binary case … see notes
    - Upper and lower bounds on sigmoid function …. See notes

Laplace approximation of posterior to quantify uncertainty in predictions

Discuss using HW problem

$$p(x\,|\,\mu) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

$$p(\mu) = \lambda\exp(-\lambda\mu);\ \mu \ge 0;\ \lambda > 0$$

$$p(\mu\,|\,\{x_n\}_{n=1}^N) \propto p(\{x_n\}_{n=1}^N\,|\,\mu)\,p(\mu) = L(\mu)$$

$$\min_{\mu\ge 0}\ J = -\ln L(\mu) = \frac{1}{2\sigma^2}\left(\sum_{n=1}^N (x_n-\mu)^2\right) + \lambda\mu + constant$$

$$\frac{\partial J}{\partial\mu} = \lambda - \frac{1}{\sigma^2}\sum_{n=1}^N (x_n-\mu)$$

$$\Rightarrow \hat{\mu}_{MAP} = \max\left(\frac{\sum_{n=1}^N x_n - \sigma^2\lambda}{N}, 0\right)$$

$$\frac{\partial^2 J}{\partial\mu^2} = H = \frac{N}{\sigma^2}$$

$$so,\ Laplace\ approximation\ of\ \ posterior: p(\mu\,|\,\{x_n\}_{n=1}^N) \sim N\left(\hat{\mu}_{MAP}, \frac{\sigma^2}{N}\right)$$

$$P(z \mid \underline{x}, D) = \int_{\underline{w}} P(z \mid \underline{x}, \underline{w})\, p(\underline{w} \mid D)\, d\underline{w}$$

(i) Plug-in MAP estimate $\Rightarrow p(\underline{w} \mid D) = \delta(\underline{w} - \underline{w}^*)$: $P(z \mid \underline{x}, D) \approx P(z \mid \underline{x}, \underline{w}^*)$

(ii) Monte Carlo approximation: $P(z \mid \underline{x}, D) \approx \dfrac{1}{M} \sum_{m=1}^{M} P(z \mid \underline{x}, \underline{w}^m)$

$\underline{w}^m$ is drawn from the posterior distribution, $p(\underline{w} \mid D)$

(iii) Probit approximation: $P(z \mid \underline{x}, D) \approx \int_{\underline{w}} P(z \mid \underline{x}, \underline{w}) N(\underline{w}; \underline{w}^*, H^{-1})\, d\underline{w} \approx g\left(\dfrac{\mu_a}{\left(1 + \pi \sigma_a^2 / 8\right)^{1/2}}\right)$

$\mu_a = \underline{x}^T \underline{w}^*; \sigma_a^2 = \underline{x}^T H^{-1} \underline{x}$

Plug-in underestimates uncertainty

MC: Posterior mean decision boundary

Laplace + probit is similar to MC

$$\Phi(a) \triangleq \int_{-\infty}^{a} N(x; 0, 1)\, dx \quad \textit{cumulative distribution function } (CDF); \Phi(-\infty) = 0; \Phi(\infty) = 1$$

Sigmoid function g(a) approximates probit $\Phi(\lambda a)$ well if slopes are matched at the origin. Note g(-∞)=0; g(∞)=1

$$g'(0) = g(0)[1 - g(0)] = \frac{1}{4}; \Phi'(0) = \frac{\lambda}{\sqrt{2\pi}} \Rightarrow \lambda = \sqrt{\frac{\pi}{8}}$$

$$P(z = 1 \mid \underline{x}, D) \approx \int_{\underline{w}} P(z = 1 \mid \underline{x}, \underline{w}) N(\underline{w}; \underline{w}^*, H^{-1})\, d\underline{w}$$

$$= \int_{\underline{w}} \left(\frac{1}{1 + e^{-\underline{w}^T \underline{x}}} \right) N(\underline{w}; \underline{w}^*, H^{-1})\, d\underline{w}$$

$$\textit{Let } a = \underline{w}^T \underline{x} = \underline{x}^T \underline{w} \Rightarrow a = N(a; \underbrace{\underline{x}^T \underline{w}^*}_{\mu_a}, \underbrace{\underline{x}^T H^{-1} \underline{x}}_{\sigma_a^2}). \textit{ So, } P(z = 1 \mid \underline{x}, D) \approx \int_a \underbrace{\left(\frac{1}{1 + e^{-a}} \right)}_{g(a)} N(a; \mu_a, \sigma_a^2)\, da$$

$$\approx \int_a \Phi(\sqrt{\pi / 8}\, a) N(a; \mu_a, \sigma_a^2)\, da = \Phi\left(\frac{\mu_a}{\left(\sigma_a^2 + 8/\pi\right)^{1/2}} \right) \approx g\left(\frac{\mu_a}{\left(1 + \pi \sigma_a^2 / 8\right)^{1/2}}\right)$$

$$Define\, x = z - \lambda a; z \sim N(z;0,1)$$
$$\Rightarrow p(x) = N(x; -\lambda\mu_a, 1+\lambda^2\sigma_a^2)$$
$$p(x \mid a) = N(x; -\lambda a, 1) \Rightarrow P(x \le 0 \mid a) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{0} e^{-(x+\lambda a)^2/2} dx = \Phi(\lambda a)$$
$$P(x \le 0) = \int_a P(x \le 0 \mid a) N(a; \mu_a, \sigma_a^2) da$$
$$= \int_a \Phi(\lambda a) N(a; \mu_a, \sigma_a^2) da = \Phi\left(\frac{\mu_a}{\left(\sigma_a^2 + 8/\pi\right)^{1/2}} \right)$$

$$AIC \triangleq -2\ln p(D \mid \underline{\hat{\theta}}_m) + 2dof(\underline{\hat{\theta}}_m)$$

$$BIC \triangleq -2\ln p(D \mid \underline{\hat{\theta}}_m) + dof(\hat{\theta}) \ln N$$

Training, Validation, Testing

Cross-validation: S-fold cross-validation, Leave-one-out CV, 5x2 Cross-Validation. Variation: 5 repetitions of 2-fold cross-validation on a randomized dataset, Bootstrap (sampling with replacement)

$$PE = 0.632 * PE_{training}(b) + 0.368 * PE_{val}(b)$$

$$P\{observation \notin bootstrap\ sample\} = (1 - \frac{1}{N})^N \to \frac{1}{e}\, as\ N \to \infty$$
$$\Rightarrow P\{observation \in validation\ samples\} = 0.368$$
$$\Rightarrow P\{observation \in training\ samples\} = 0.632$$

Discuss measures

Cobweb

ROC measures

Multi-class extensions

McNemar Test