CS 276 Course Review

Intro/Patterns

- In pattern recognition, a pattern can be anything perceived.
- **Generative Model -** model for randomly generating observable data values. A generative model is a full probablisitc model of all varaibles
- **Discriminative Model -** model dependence of an unobserved variable y on an observed variable x. That is, given an observed x, it predicts an associated y.
- Differences between discriminative and generative models:
    - Generative is a full probablistic model of all variables; discriminative models provide a model for only the target variable(s) conditional on the observed data.
    - Generative models can be used to simulate values of any variable in the model; discriminative models only allow sampling the target variables conditional on the observed quantities.
    - Generative models capture the joint distribution between x and y, but discriminative models can perform better when the joint distribution is not required.
    - Most discriminative models are supervised (since they try to determine y based on x).

1) Discriminative methods:
    - Models find the discriminative target without understanding the underlying pattern (structure) or develop a full mathematical description.
    - Aim to minimize a utility function (e.g. classification error)
    - Rely on large, label training data
    - Usefull when all we need is classification (do not need to generalize the algorithm)
2) Generative methods:
    - Bayesian school, pattern theory

    1) Define patterns and regularities
    2) Specify likelihood model for how signals are generated
    3) Learning probability models from collections of signals
    4) Interfences

    - Basic idea: build models for underlying patterns, and can learned, adapted, and generalized with data.

Bayesian Decision Theory

- The basic framework for pattern recognition
- y represents a belief about the current situation (e.g. a belief about what an observable X is). We have a set of features, $x, which describe X. We then take an α which is a decision based on our belief and the observable X.
- For the fish example: X = image of the fish, x = (brightness, length, # of fin, etc), y is the belief what type of fish X is ($\Omega^c = \{$"sea bass", "salmon", "trout"$\}$), and α is the decision we make for the fish (in this case $\Omega^c = \Omega^\alpha$).

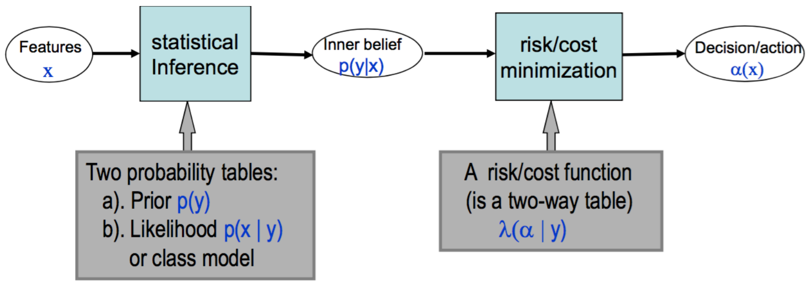


Figure 1: Bayesian decision process

- The **inner belief** (Bayes' Rule) is given by:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

  where $p(y)$ is the **prior probability**, $p(x|y)$ is the **likelihood**.

- **Joint probability** is given by

$$p(x, y) = p(y|x)p(x)$$

- The **expected risk** is computed with:

$$R(\alpha_i|x) = \sum_{j=1}^{k} \lambda(\alpha_i|y = j)p(y = j)|x)$$

  where $\lambda(\alpha_i|y = j)$ is the **penalty** (usually a matrix) of taking action α_i when the true outcome is $y = j$ the $p(y = j|x)$ is the likelihood the true outcome is $y = j$ given data x.

- The optimal action minimizes conditional risk
- A simple example: you a betting on which side a tossed coin will fall on. You win \$1 if you are right and lose \$1 if you are wrong. Let x be whtever evidence that we observe. Together with some prior information about the coin, you believe the head has a 60% chance and tail 40%.

$$y \in H, T, \alpha \in H, T, \lambda(\alpha(x)|y) = \alpha \binom{-1+1}{+1-1}$$

- λ here represents a loss table (so if we guess correctly we have a negative loss)

$$R(\alpha_i|x) = \alpha \binom{-1+1}{+1-1} \binom{0.6}{0.4}$$
$$R(\alpha_i|x) = \alpha \binom{-0.2}{0.2}$$

- so we choose H, which corresponds to a loss of -0.2
- A **decision rule** is a mapping function from the feature space to a set of actions $(\alpha(x) : \Omega^d \rightarrow \Omega^\alpha)$
  - A decision is made to minimize the average (expected) cost (risk):

$$\alpha(x) = \underset{\Omega^\alpha}{\text{argmin}} R(\alpha|x)$$

- Special Case: 0-1 loss
  - Let $\lambda(\alpha|y) = 0$ if $\alpha = y$
  - Let $\lambda(\alpha|y) = 1$ if $\alpha \neq y$
  - The risk for classifying x to class $\alpha = i$ is the probability of misclassification, which is given by:

$$R(\alpha = i|x) = \sum_{y \neq i} p(y|x) = 1 - p(y = i|x) = \lambda(\alpha|y = i)$$

  - The optimal decision to make is the class that has the maximum posterior probability

$$\alpha(x) = \underset{\Omega^\alpha}{argmin}(1 - p(\alpha|x)) = \underset{\Omega^\alpha}{argmax}(p(\alpha|x))$$

Discriminant Functions/Decision Boundaries

- **Discriminant Functions** are functions which describe a mathematical description of the optimal decision to make. That is, they divide a decision space into optimal decisions based on some empirical criteria (min loss, min risk, etc).
- Discriminant functions partition the feature space through a set of functions $\{g_1(x), ...g_k(x)\}$
- The optimal decision is $\alpha(x) = \text{argmax}\{g_1(x), ...g_k(x)\}$
- For Guassian distributions (see HW 1 problem 1), we have following proof:
  Proof Let $p(x|w_i) \approx \mathcal{N}(\mu_i, \sigma^2 I)$. Then

$$\begin{aligned} g_i(x) &= \log p(x|y=i) + \log p(y=i) \\ &= \frac{-1}{2\sigma^2}(-2\mu_i^T x + \mu_i^T \mu_i) + \log p(y=i) \\ &= \frac{1}{\sigma^2}\mu_i^T x + \frac{-1}{2\sigma^2}\mu_i^T \mu_i + \log p(y=i) \end{aligned}$$

Bayes Risk, Bayes Error

- **Bayes Risk** is the risk of the optimal decision (α):

$$R(\alpha) = \int R(\alpha(x)|x)p(x)dx = \int \sum_{y=1}^{k} \lambda(\alpha(x)|y)p(y|x)p(x)dx = \int \sum_{y=1}^{k} \lambda(\alpha(x)|y)p(y,x)dx$$

- We can estimate the Bayes risk (known as **empirical risk**) by summing all the risks (wrong decisions over the testing set)

$$\text{Given } D = \{(x_i, y_i); i = 1, 2, ..., m\} \text{ Bayes risk} = R(\alpha) = \sum_{j=1}^{m} \lambda(\alpha(x_j)|y-j)$$

- The **Bayesian Decision Policy** minimizes loss $\pi = \text{argmin } R_\pi$ (always picks the action with the lowest risk).
  - For the case of 0/1 loss, the Bayesian decision policy will always choose the class with the highest posterior probability $p(y=i|x)$. Accordingly, the risk of this decision is $1 - p(y=i|x)$

ROC and PR Curves

- **ROC curves** plot of comparing the true positive rate (TPR, correctly labeled positive samples) against the false positive rate (FPR, incorrectly labeled negative samples)

$$TPR = \frac{\text{\# true positives}}{\text{\# positives}}$$

$$FPR = \frac{\text{\# false positives}}{\text{\# negatives}}$$

- The threshold is slide across the entire range of the distribution. A point on the curve is plotting using the FPR vs TPR (x,y) at a particular value of the threshold.
- **PR Curves** (precision-recall) plot the precision (measure of correctness) against a recall (measure of completeness).

$$\text{Precision} = \frac{\text{\# true positive}}{\text{\# true positive + \# false positive}}$$

$$\text{Recall} = \frac{\text{\# true positive}}{\text{\# true positive + \# false negative}}$$

- The threshold is slide across the distribution the same was as in ROC curves, just computing different values

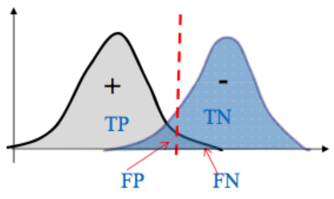


Figure 2: Sample of threshold sliding across distributions

Dimension Reduction Techniques

- **Dimension Reduction** reduces the complication of classification, but can result in worse classification
- Dimension reduction is only applicable to signals that can afford to lose data (since reducing the dimensionality is literally throwing away data)
  - white noise cannot be reduced - it is full of unique data (we need data which will separate after dimension reduction)
- Dimension reduction techniques can be classified in three axes:
  1) Generative (PCA) vs discriminative (Fisher's linear discriminant)
  2) Linear (PCA, Fisher) vs. Non-linear (MDS)
  3) Global (porjection, e.g. PCA) vs Local (nearest neighbor, e.g. LLE)

Principle Component Analysis

- A generative method. Basic idea: reduces the data down to dimensions whose primary axis is in the direction of the greatest amount of variance (minimizing the variance when the data is projected onto this axis), then generate the data using these reduced dimensions.

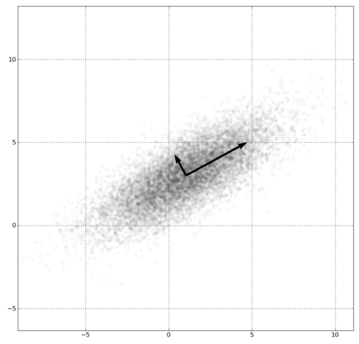


Figure 3: Principle axis on a multivariate distribution

- Variance minimization: find $\boldsymbol{u}$ that minimizes the projected variance

$$\text{variance} = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{u}^T\boldsymbol{x_n} + \boldsymbol{u}^T\bar{\boldsymbol{x}}\right) \quad = \boldsymbol{u}^T S\boldsymbol{u}$$

where $S = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{x_n} - \bar{\boldsymbol{x}}\right)^T\left(\boldsymbol{x_n} - \bar{\boldsymbol{x}}\right)$. S is called the scatter (covariance) matrix.

- **Goal**: find minimize variance ($\boldsymbol{u}^T S\boldsymbol{u}$) under the constraint $\boldsymbol{u}$ is a unit vector ($||\boldsymbol{u}|| = 1 \implies g(\boldsymbol{u}) = \boldsymbol{u}^T\boldsymbol{u} - 1$)
- We can use a Lagrange multiplier to solve for this constraint.

$$\begin{aligned} L(\boldsymbol{u}, \lambda) &= f(\boldsymbol{u}) - \lambda g(\boldsymbol{u}) \\ &= \boldsymbol{u}^T S\boldsymbol{u} - \lambda(\boldsymbol{u}^T\boldsymbol{u} - 1) \\ \frac{\partial L}{\partial \boldsymbol{u}} &= S\boldsymbol{u} - \lambda\boldsymbol{u} = 0 \\ &\implies S\boldsymbol{u} = \lambda\boldsymbol{u} \end{aligned}$$

so $\boldsymbol{u}$ is an eigenvector of the covariance matrix S.

Fisher Linear Discriminant

- In a two-class classification problem, given: n samples $textx_1, ...\mathrm{x}_n$ in a d-dimensional feature space, n_1 in subset X_1 with label w_1 and n_2 in subset X_2 labeled w_2.
- **Goal**: find a vector w and project the n samples on this acis $y = w^T\mathbf{x} =< w, \mathbf{x} >$, so that the project samples are well separated.
  - Find a line to project data onto that class separation

$$\mathbf{y} = w^T\mathbf{x}$$

  - This projects $\mathbf{x}$ onto w.

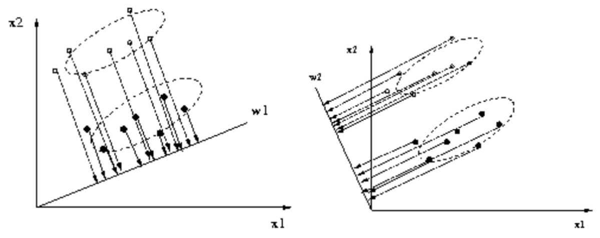


Figure 4: Desired FDA projection

- **Sample mean** for class w_i:

$$\mathbf{m}_i = \frac{1}{n_i}\sum_{x\in X_i}\mathbf{x}, \quad i = 1, 2$$

- **Scatter matrix** for class w_i:

$$\mathrm{S}_i = \sum_{x\in X_i}(\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T \quad i = 1, 2$$

- The **between class scatter matrix**:

$$\mathrm{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$$

- The **within-class scatter matrix**:
$$\mathrm{S}_W = \mathrm{S}_1 + \mathrm{S}_2$$
- The **sample mean** of the projected points in class w_i:
$$\tilde{m_i} = \frac{1}{n_i} \sum_{x \in X_i} w^T \mathbf{x} = w^T \mathbf{m}_i, \qquad i = 1, 2$$
- The **scatter** of the projecte points in class w_i:
$$\tilde{s_i} = \sum_{x \in X_i} (w^T \mathbf{x} - w^T \mathbf{m}_i)^2 = w^T \mathrm{S}_i w, \qquad i = 1, 2$$
- The sample mean and the scatter are 1-dimensional variables
- Fisher's linear discriminant $y = w^T \mathbf{x}$: choose w to maximize
$$J(w) = \frac{|\tilde{m_1} - \tilde{m_2}|^2}{\tilde{s_1}^2 + \tilde{s_2}^2} = \frac{w^T \mathrm{S}_B w}{w^T \mathrm{S}_W w}$$
i.e. the between class distance should be as large as possible, and the within class scatter should be as small as possible (known as the **Rayleigh quotient**).
- The Rayleigh coefficient means a vector w which minimizes J satisfies the generalized eigenvalue problem:
$$S_B w = \lambda S_W w$$
- Note that S_B is always in the direction of $\mathbf{m}_1 - \mathbf{m}_2$, the solution can be written as:
$$w = \mathrm{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

Multiple Discriminant Analysis

- For a c-class problem, we can generalize the scatter matrices.
- The within-class scatter matrix becomes:
$$\mathrm{S}_W = \mathrm{S}_1 + \mathrm{S}_2 + ... + \mathrm{S}_{c-1}$$
where S_i is the scatter matrix computed from samples inside class w_i.
- The between-class scatter matrix becomes:
$$\mathrm{S}_B = \mathrm{S}_{total} - \mathrm{S}_W = \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$
where S_{total} is the total scatter matrix computed from all samples.
- We want vectors $w_i, i = 1, 2, ..., c-1$, and project the samples from d-dimensional feature space $ x = (x_1, x_2, …, x_d)$ to the $c-1$ dimensional space $\mathbf{y} = (y_1, y_2, ..., y_{c-1})$.
$$\mathbf{y} = (w_1^T \mathbf{x}, ..., w_{c-1}^T \mathbf{x}) = W^T \mathbf{x}$$
where W is a $(c-1)xd$ matrix with w_i being the i-th column.
- The criterion for the optimal W is (Note: the bars indicate a determinant)
$$J(W) = \frac{|W^T \mathrm{S}_B W|}{|W^T \mathrm{S}_W W|}$$

Multi-dimenstional Scaling (MDS)

- Objective: project data points into 1, 2, or 3-dimensional spaces so that the spatial distance of these data points are preserved.
- MDS is used for two purposes:
  1) Visual the structures and properties of data, so that we may select proper models for them.
  2) Verify some distance (metric) measure on an unknown data set.
- With a good distance measure, data should correspond to a "meaningful" cluster.
- Given: a set of data points in d space $\{x_1, ..., x_n\}$ and a dissimilarity/distance measure/metric between two points $x_i,\ x_j:\ \delta_{ij}$.
- Objective: find points in 1, 2, or 3-space $\{y_1, ...y_n\}$ with usually Eclidean distances d_{ij} for two points y_i and y_j.
- One criterion: stress

$$Stress = \frac{\sum_{i,j}(d_{ij} - \delta_{ij})^2}{\sum_{i,j}\delta_{ij}^2}$$

  and minimize the total pair-wise stress between each point

Local Linear Embedding (LLE)

- A generative, non-linear, and local technique for dimension reduction

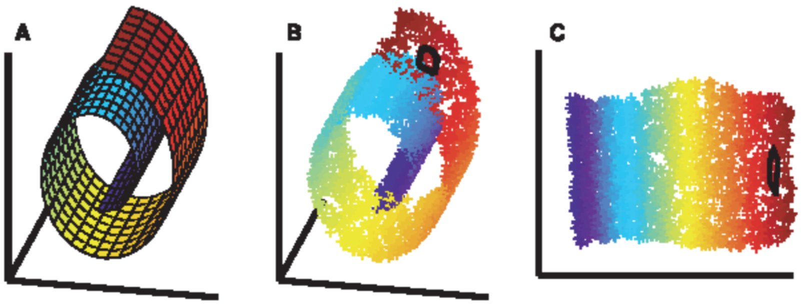


Figure 5: LLE manifold unfolding

- Objective: use the neighborhood of points to build a reconstructive basis
- LLE algorithm:
  1) Compute the neighbors of each data point, $\overrightarrow{X}_i$
  2) Compute weights W_{ij} that best reconstruct each data point $\overrightarrow{X}_i$ from its neighbors, minimizing the cost according to:

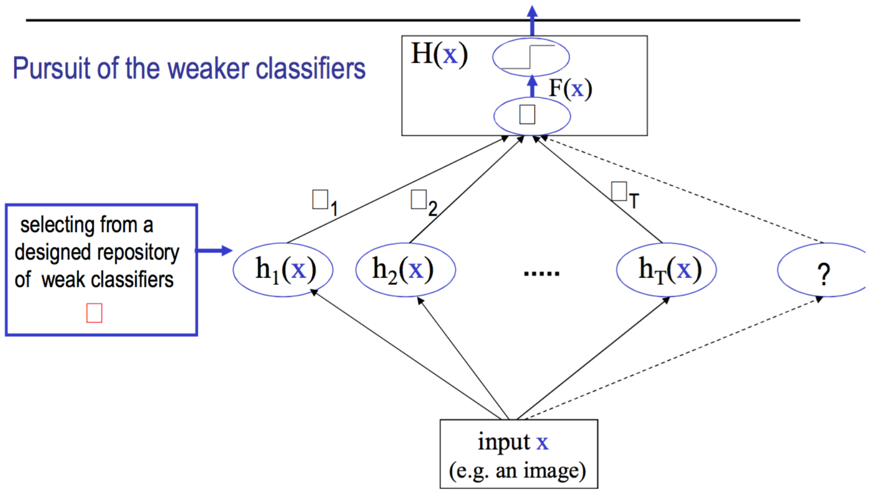
$$\epsilon(W) = \sum_i |X_i \sum_j W_{ij} X_j|^2$$

     Such that $\sum_{ij} = 1$. Thus ϵ gives you the squared difference between the data points and their reconstructions.
  3) Compute the vectors Y_i best reconstructed by the weights W_{ij}, minimizing the quadratic equation:

$$\phi(Y) = \sum_i |\overrightarrow{Y_i} - \sum_j W_{ij} \overrightarrow{Y_j}|^2$$

Boosting

- Technique for combining a number of “weak” classifiers to make a “strong” classifier.
- Several versions: Adaboost, RealBoost, LogitBoost - each optimizes a different design of loss functions.
- Basic idea: adds a hidden layer of nuerons one by one until the classification error on the training images goes to zero, by sequentially minimizing the upper-bound of the empirical error.
- For every type of boosting we have:
  1) A set of labeled training data from 2 classes:
  $$\Omega = \{(x_i, y_i) : x_i \in \Omega^d, y_i \in \{-1, 1\},\ i = 1, ..., m\}$$
  2) A set of weak classifiers, a pool constructed for a task
  $$\Delta = \{h_t(x) : t = 1, ..., T\}$$
  $$h_t(x) : \Omega^d \to \{-1, 1\}$$
- Basic idea pt. 2: Suppose we have two weak classifiers h_1 and h_2 which have 49% error each (slightly better than random chance), can we combine them to make a new classiifer so that the error becomes lower? If so, we can repeat this process to make the error zero.



Pursuit of the weaker classifiers
H(x)
F(x)
selecting from a designed repository of weak classifiers
h1(x)
h2(x)
.....
hT(x)
?
input x (e.g. an image)

Figure 6: The boost classifier

- The error rate of a weak classifier h is calculated with:

$$\epsilon(h) = \frac{1}{m} \sum_{i=1}^{m} 1(h(x_i) \neq y_i) < \frac{1}{2}$$

where 1() is an indicator function (equals 1 if the condition is true, 0 otherwise).

- A **weak classifier** works better than random chance. For binary classifiers, it has less than 50% errors. If the error is larger than 50%, we flip the sign of the classifier to make the error less than 50%. We just cannot have classifiers with exactly 50% error (they provide no insight/additional information).
- The **strong classifier** is a linear combination of the weak classifiers:

$$H(x) = sign\{\alpha_1 h_1(x) + ... + \alpha_T h_T(x)\} = sign\{F(x)\}$$

- Objective: choose $h = (h_1, ...h_t)$ and $\alpha = (\alpha_1, ..., \alpha_T)$ to minimize the empirical error:

$$Err(H) = \frac{1}{m}\sum_{i=1}^{m} 1(H(x_i) \neq y_i)$$

Adaboost

- Characteristics of Adaboost:
  1) AdaBoost is a *sequential algorithm* that minimizes an upper bound of the classification error $Err(H)$ by selecting *weak classifiers* h and their *weights* α one-by-one (a pursuit). Each time, a weak classifier is selected to maximally reduce the upper bound of error.
  2) AdaBoost assigns *weights to the data samples.* The weights are summed to one (that is, they are normalized with each round of boosting). These weights are updated when a new weak classifier h_t is added. Data samples that are misclassified by h_t are given more weight so that they recieve more "attention" in selecting the next weak classifier (because the weights are normalized to 1, this means the correctly classified data samples are given less "attention" in selecting the next weak classifier).
  3) Each step, a new weak classifier is selected to minimize the weighted error, so pays greater attention to misclassified samples.
  4) The empirical error will converge to zero at an exponential rate.
- The basic Adaboost algorithm:
  0) Initialize the data with uniform weight $\rightarrow D_0 = \frac{1}{m},\ \forall X_i \in \Omega$. Set t = 0.
  1) At step t, compute the weighted error for each weak classifier

$$\epsilon(h) = \sum_{i=1}^{m} D_t(x_i) 1(h(x_i) \neq y_i),\ \forall h \in \Delta$$

  2) Choose a new weak classifier which has the least weighted error:

$$h_t = \operatorname*{argmin}_{h \in \Delta} \epsilon_t(h)$$

  3) Assign a weight for the new classifier:

$$\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)}$$

Max Likelihood Estimation

- Setup: Given data $D = (x_1, ..., x_n)$ $(x_i \in \mathbb{R}^d)$. Assume a set of distributions $\{p_\theta | \theta \in \Theta\}$. Asumme D is a sample from $X_1, X_2, ..., X_N$ p_θ $(p_\theta(x) = p(x|\theta))$ iid (independent and identically distributed random variables) for some $\theta \in \Theta$.
- Goal: Estimate the true θ the D comes from
- Def: θ_{MLE} is a MLE for θ if $\theta_{MLE} = \operatorname*{argmax}_{\theta \in \Theta} p(D|\theta)$
  - More precisely, $p(D|\theta_{MLE}) = \max_{\theta \in \Theta} p(D|\theta)$
  - $p(D|\theta)$ is the likelihood function, a function of θ
- Since we said D is iid, $p(D|\theta) = p(x_1, x_2, ..., x_n)|\theta) = \prod_{i=1}^{n} p(x_i|\theta) = \prod_{i=1}^{n} P(X_i = x_i|\theta)$
- Remarks:
  1. MLE might not be unique
  2. MLE may fail to exist -> max may not be achieved at any θ

- Pros
  1. Easy to compute
  2. Interpretable
  3. Asymptotic properties:
     - Consistent - approaches true θ as N approaches ∞
     - Normal - distribution when N is very large is normal
- Cons:
  1. Point estimate - no representation of uncertainty