

# CS 189 Final Note Sheet

## Bayesian Decision Theory

Bayes Rule:  $P(\omega|x) = \frac{P(x|\omega)P(\omega)}{P(x)}$ ,  $P(x) = \sum_i P(x|\omega_i)P(\omega_i)$   
 $P(x, \omega) = P(x|\omega)P(\omega) = P(\omega|x)P(x)$   
 $P(\text{error}) = \int_{-\infty}^{\infty} P(\text{error}|x)P(x)dx$

$$P(\text{error}|x) = \begin{cases} P(\omega_1|x) & \text{if we decide } \omega_2 \\ P(\omega_2|x) & \text{if we decide } \omega_1 \end{cases}$$

$$0\text{-1 Loss: } \lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j \text{ (correct)} \\ 1 & i \neq j \text{ (mismatch)} \end{cases}$$

Expected Loss (Risk):  $R(\alpha_i|x) = \sum_{j=1}^c \lambda(\alpha_i|\omega_j)P(\omega_j|x)$

0-1 Risk:  $R(\alpha_i|x) = \sum_{j \neq i}^c P(\omega_j|x) = 1 - P(\omega_i|x)$

## Probabilistic Motivation for Least Squares

$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$  with noise  $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$

Note: The intercept term  $x_0 = 1$  is accounted for in  $\theta$

$$\implies p(y^{(i)}|x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

$$\implies L(\theta) = \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

$$\implies l(\theta) = m \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{i=1}^m (y^{(i)} - \theta^T x^{(i)})^2$$

$$\implies \max_{\theta} l(\theta) \equiv \min_{\theta} \sum_{i=1}^m (y^{(i)} - h_{\theta}(x))_+^2$$

Gaussian noise in our data set  $\{x^{(i)}, y^{(i)}\}_{i=1}^m$  gives us least squares  
 $\min_{\theta} \|X\theta - y\|_2^2 \equiv \min_{\theta} \theta^T X^T X \theta - 2\theta^T X^T y + y^T Y$

$$\nabla_{\theta} l(\theta) = X^T X \theta - X^T y = 0 \implies \boxed{\theta^* = (X^T X)^{-1} X^T y}$$

Gradient Descent:  $\theta_{t+1} = \theta_t + \alpha(y_t^{(i)} - h(x_t^{(i)}))x_t^{(i)}$ ,  $h_{\theta}(x) = \theta^T x$

## Least Squares Solution

$\min_x \|Ax - y\|_2^2 \implies x^* = A^{\dagger} y$  min norm sol'n

Sol'n set:  $x_0 + N(A) = x^* + N(A)$

$$A^{\dagger} = \begin{cases} (A^T A)^{-1} A^T & A \text{ full column rank} \\ A^T (A A^T)^{-1} & A \text{ full row rank} \\ V \Sigma^{\dagger} U^T & \text{any } A \end{cases}$$

L2 Reg:  $\min_x \|Ax - y\|_2^2 + \lambda \|x\|_2^2 \implies x^* = (A^T A + \lambda I)^{-1} X^T y$

The above variant is used when A contains a null space. L2 Reg falls out of the MLE when we add a Gaussian prior on x with  $\Sigma = cI$ . We get L1 Reg when x has a Laplace prior.

## Logistic Regression

Classify  $y \in \{0, 1\} \implies$  Model  $p(y = 1|x) = \frac{1}{1 + e^{-\theta^T x}} = h_{\theta}(x)$

$$\frac{dh_{\theta}}{d\theta} = \left(\frac{1}{1 + e^{\theta^T x}}\right)^2 e^{-\theta^T x} = \frac{1}{1 + e^{\theta^T x}} \left(1 - \frac{1}{1 + e^{\theta^T x}}\right) = h_{\theta}(1 - h_{\theta})$$

$$p(y|x; \theta) = (h_{\theta}(x))^y (1 - h_{\theta}(x))^{1-y} \implies$$

$$L(\theta) = \prod_{i=1}^m (h_{\theta}(x^{(i)}))^{y^{(i)}} (1 - h_{\theta}(x^{(i)}))^{1-y^{(i)}} \implies$$

$$l(\theta) = \sum_{i=1}^m y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \implies$$

$$\nabla_{\theta} l = \sum_i (y^{(i)} - h_{\theta}(x^{(i)})) x^{(i)} = X^T (y - h_{\theta}(X)), \text{ (want max } l(\theta))$$

$$\text{Stoch: } \boxed{\theta_{t+1} = \theta_t + \alpha(y_t^{(j)} - h_{\theta}(x_t^{(j)}))x_t^{(j)}}$$

$$\text{Batch: } \boxed{\theta_{t+1} = \theta_t + \alpha X^T (y - h_{\theta}(X))}$$

## Multivariate Gaussian $X \sim \mathcal{N}(\mu, \Sigma)$

$$f(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

$$\Sigma = E[(X - \mu)(X - \mu)^T] = E[XX^T] - \mu\mu^T$$

$\Sigma$  is PSD  $\implies x^T \Sigma x \geq 0$ , if inverse exists  $\Sigma$  must be PD

If  $X \sim \mathcal{N}(\mu, \Sigma)$ , then  $AX + b \sim \mathcal{N}(A\mu + b, A\Sigma A^T)$

$$\implies \Sigma^{-\frac{1}{2}}(X - \mu) \sim \mathcal{N}(0, I), \text{ where } \Sigma^{-\frac{1}{2}} = U\Lambda^{-\frac{1}{2}}$$

The distribution is the result of a linear transformation of a vector of univariate Gaussians  $Z \sim \mathcal{N}(0, I)$  such that  $X = AZ + \mu$  where we have  $\Sigma = AA^T$ . From the pdf, we see that the level curves of the distribution decrease proportionally with  $x^T \Sigma^{-1} x$  (assume  $\mu = 0$ )  $\implies$

$$c\text{-level set of } f \propto \{x : x^T \Sigma^{-1} x = c\}$$

$$x^T \Sigma^{-1} x = c \equiv x^T U \Lambda^{-1} U^T x = c \implies$$

$$\underbrace{\lambda_1^{-1} (u_1^T x)^2}_{\text{axis length: } \sqrt{\lambda_1}} + \dots + \underbrace{\lambda_n^{-1} (u_n^T x)^2}_{\text{axis length: } \sqrt{\lambda_n}} = c$$

Thus we have that the level curves form an ellipsoid with axis lengths equal to the square root of the eigenvalues of the covariance matrix.

## LDA and QDA

Classify  $y \in \{0, 1\}$ , Model  $p(y) = \phi^y \phi^{1-y}$  and

$$p(x|y = 1; \mu_1) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_1)^T \Sigma^{-1} (x - \mu_1)\right)$$

$l(\theta, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^m p(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \Phi)$  gives us

$$\phi_{MLE} = \frac{1}{m} \sum_{i=1}^m 1\{y^{(i)} = 1\}, \mu_{MLE} = \text{avg of } x^{(i)} \text{ classified as } k,$$

$\Sigma_{MLE} = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu_{y(i)})(x^{(i)} - \mu_{y(i)})^T$ . Notice the covariance matrix is the same for all classes in LDA.

If  $p(x|y)$  multivariate gaussian (w/ shared  $\Sigma$ ), then  $p(y|x)$  is logistic function. The converse is NOT true. LDA makes stronger assumptions about data than does logistic regression.

$$h(x) = \arg \max_k -\frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k) + \log(\pi_k)$$

where  $\pi_k = p(y = k)$

For QDA, the model is the same as LDA except that each class has a unique covariance matrix.

$$h(x) = \arg \max_k -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log(\pi_k)$$

## Optimization

Newtons Method:  $\theta_{t+1} = \theta_t - [\nabla_{\theta}^2 f(\theta_t)]^{-1} \nabla_{\theta} f(\theta_t)$

Gradient Decent:  $\theta_{t+1} = \theta_t - \alpha \nabla_{\theta} f(\theta_t)$ , for minimizing

Lagrange Multipliers:

Given,  $\min_x f(x)$  s.t.  $g_i(x) = 0$ ,  $h_i(x) \leq 0$  the corresponding

Lagrangian is:  $L(x, \alpha) = f(x) + \sum_{i=1}^k \alpha_i g_i(x) + \sum_{i=1}^l \beta_i h_i(x)$

We min over x and max over the Lagrange multipliers  $\alpha$  and  $\beta$

## Support Vector Machines

In the strictly separable case, the goal is to find a separating hyperplane (like logistic regression) except now we don't just want any hyperplane, but one with the largest margin.

$H = \{\omega^T x + b = 0\}$ , since scaling  $\omega$  and b in opposite directions doesn't change the hyperplane our optimization function should have scaling invariance built into it. Thus, we do it now and

define the closest points to the hyperplane  $x_{sv}$  (support vectors) to satisfy:  $|\omega^T x_{sv} + b| = 1$ . The distance from any support

vector to the hyper plane is now:  $\frac{1}{\|\omega\|_2}$ . Maximizing the distance to the hyperplane is the same as minimizing  $\|\omega\|_2$ .

The final optimization problem is:

$$\min_{\omega, b} \frac{1}{2} \|\omega\|_2^2 \text{ s.t. } y^{(i)} (\omega^T x^{(i)} + b) \geq 1, i = 1, \dots, m$$

Primal:  $L_p(\omega, b, \alpha) = \frac{1}{2} \|\omega\|_2^2 - \sum_{i=1}^m \alpha_i (y^{(i)} (\omega^T x^{(i)} + b) - 1)$

$$\frac{\partial L_p}{\partial \omega} = \omega - \sum \alpha_i y^{(i)} x^{(i)} = 0 \implies \omega = \sum \alpha_i y^{(i)} x^{(i)}$$

$\frac{\partial L_p}{\partial b} = -\sum \alpha_i y^{(i)} = 0$ , Note:  $\alpha_i \neq 0$  only for support vectors. Substitute the derivatives into the primal to get the dual.

$$\text{Dual: } L_d(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}$$

In the non-separable case we allow points to cross the marginal boundary by some amount  $\xi$  and penalize it.

$$\min_{\omega, b} \frac{1}{2} \|\omega\|_2^2 + C \sum_{i=1}^m \xi_i \text{ s.t. } y^{(i)} (\omega^T x^{(i)} + b) \geq 1 - \xi_i$$

The dual for non-separable doesn't change much except that each  $\alpha_i$  now has an upper bound of C  $\implies 0 \leq \alpha_i \leq C$

## Loss Functions

In general the loss function consists of two parts, the loss term and the regularization term.  $J(\omega) = \sum_i \text{Loss}_i + \lambda R(\omega)$

## Nearest Neighbor

Key Idea: Store all training examples  $\langle x_i, f(x_i) \rangle$

**NN:** Find closest training point using some distance metric and take its label.

**k-NN:** Find closest k training points and take on the most likely label based on some voting scheme (mean, median,...)

**Behavior at the limit:**  $1\text{NN } \lim_{N \rightarrow \infty} \epsilon^* \leq \epsilon_{NN} \leq 2\epsilon^*$

$\epsilon^*$  = error of optimal prediction,  $\epsilon_{nn}$  = error of 1NN classifier

KNN  $\lim_{N \rightarrow \infty, K \rightarrow \infty, \frac{K}{N} \rightarrow 0} \epsilon_{knn} = \epsilon^*$

**Curse of dimensionality:** As the number of dimensions increases, everything becomes farther apart. Our low dimension intuition falls apart. Consider the Hypersphere/Hypercube ratio, it's close to zero at d=10. Solutions:

1. Get more data to fill all of that empty space
2. Get better features, reducing the dimensionality and packing the data closer together. Ex: Bag-of-words, Histograms,...
3. Use a better distance metric.

Minkowski:  $Dis_p(x, y) = (\sum_{i=1}^d |x_i - y_i|^p)^{\frac{1}{p}} = \|x - y\|_p$

0-norm:  $Dis_0(x, y) = \sum_{i=1}^d 1|x_i \neq y_i|$

Mahalanobis:  $Dis_M(x, y|\Sigma) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}$

In high-d we get "Hubs" s.t most points identify the hubs as their NN. These hubs are usually near the means (Ex: dull gray images, sky and clouds). To avoid having everything classified as these hubs, we can use cosine similarity.

**K-d trees** increase the efficiency of nearest neighbor lookup.

## Gradients

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \triangleq \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \dots & \frac{\partial y_m}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \dots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}, \frac{\partial(A\mathbf{x})}{\partial \mathbf{x}} = A^T, \frac{\partial(\mathbf{x}^T A)}{\partial \mathbf{x}} = A,$$

$$\frac{\partial(\mathbf{x}^T \mathbf{x})}{\partial \mathbf{x}} = 2\mathbf{x}, \frac{\partial(\mathbf{x}^T A \mathbf{x})}{\partial \mathbf{x}} = (A + A^T)\mathbf{x}, \frac{\partial(\text{tr} BA)}{\partial A} = B^T$$

## Generative vs. Discriminative Model

**Generative:** Model class conditional density  $p(x|y)$  and find  $p(y|x) \propto p(x|y)p(y)$  or model joint density  $p(x, y)$  and marginalize to find  $p(y = k|x) = \int_x p(x, y = k)dx$

**Discriminative:** Model conditional  $p(y|x)$ .

## Decision Trees

Given a set of points and classes  $\{x_i, y_i\}_{i=1}^n$ , test features  $x_j$  and branch on the feature which “best” separates the data. In new space (space without features  $x_j$ ) repeat the process and continue splitting data until good classification accuracy is reached.

**Heuristics:**

1. Maximize Information Gain:  $\max_j \text{infogain}(D|X_j)$

$$\max_j H(D) - \sum_{x_j \in X_j} P(X_j = x_j) \cdot H(D|X_j = x_j)$$

where  $H(D) = -\sum_{c \in C} P(y = c) \log(p(y = c))$  is the entropy of the data set,  $C$  is the set of classes each data point can take, and  $P(y = c)$  is the fraction of data points with class  $c$ .

2. Minimize Gini Impurity
3. Minimize Misclassification Impurity

**Optimal Split:**

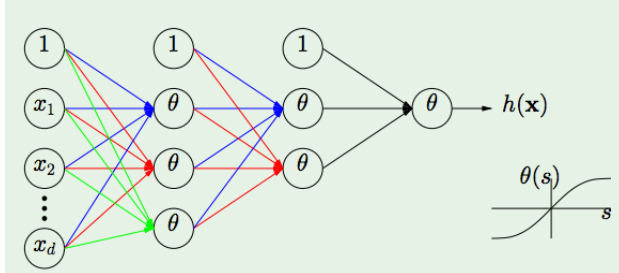
## Random Forests

**Problem:** Decision trees are unstable: small changes in the input data have large effect on tree structure  $\implies$  decision trees are high-variance estimators.

**Solution:** Random Forests train  $M$  different trees with randomly sampled subsets of the data (and sometimes with randomly sampled subsets of the features to decorrelate the trees). A new point is tested on all  $M$  trees and we take the majority as our output class (for regression we take the average of the output).

## Neural Networks

Neural Nets explore what you can do by combining perceptrons, each of which is a simple linear classifier. We use a soft threshold for each activation function  $\theta$  because it is twice differentiable.



**Activation Functions:**

$$\theta(s) = \tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}} \implies \theta'(s) = 1 - \theta^2(s)$$

$$\theta(s) = \sigma(s) \implies \theta'(s) = \sigma(s)(1 - \sigma(s))$$

**Notation:**  $w_{ij}^{(l)}$  is the weight from neuron  $i$  in layer  $l - 1$  to neuron  $j$  in layer  $l$ .

