Machine Learning Reference Sheet

By Danny Liu

1 Basic Knowledge

Assume $(x^{(i)}, y^{(i)}) \sim p(x, y)$ is generated iid such that $f(x^{(i)}) = y^{(i)}$. ML tries to approximate f
Parametric models assume $D \sim p(D; \theta)$ and so D is no longer needed after θ has been estimated
Non-parametric (memory-based) models store D and interpolate them later for prediction
Regression, linear svm, generative models (parametric) and d-trees, kNN, rbf svm (nonparametric)
kNN: find k nearest neighbors of x and predict majority label. If k too small, overfitting to noise
Curse of Dimensionality: Volume increases exponentially with dimensions so data becomes sparse

1.1 Generative Learning Algorithms

Discriminative algorithms (d-trees, softmax, svms, kNN, k-means, etc..) try to learn $p(y|x)$
Generative algorithms model $p(y)$ and $p(x|y)$ and then make predictions by maximizing the posterior probability $\underset{y}{\text{argmax}}\, p(y|x) = \underset{y}{\text{argmax}}\, \frac{p(x|y)p(y)}{p(x)} = \underset{y}{\text{argmax}}\, p(x|y)p(y)$

1.2 Gaussian Discriminant Analysis

Assume $y \sim \text{Bernoulli}(\phi)$, $x|y = 0 \sim \mathcal{N}(\mu_0, \Sigma)$, $x|y = 1 \sim \mathcal{N}(\mu_1, \Sigma)$, then solving joint likelihood $\ell(\phi, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^{n} p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^{n} p(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi)$, results in
$\phi = \frac{1}{n} \sum_{i=1}^{n} 1\{y^{(i)} = 1\}$, $\mu_j = \frac{\sum_{i=1}^{n} 1\{y^{(i)}=j\} x^{(i)}}{\sum_{i=1}^{n} 1\{y^{(i)}=j\}}$, $\Sigma = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T$
By viewing $p(y = 1|x) = f(x)$, we can write $p(y = 1|x) = \frac{1}{1+\exp(-\theta^T x)}$ where $\theta = g(\phi, \Sigma, \mu_0, \mu_1)$
If $p(x|y)$ is Gaussian, then $p(y|x)$ is logistic. However, the assumption $x|y = j \sim \text{Poisson}(\lambda_j)$ still leads to a logistic posterior. Hence GDA makes a stronger assumption that data is Gaussian
Note that while there are two mean vectors (μ_0, μ_1), GDA typically shares the covariance matrix Σ

1.3 Naive Bayes

Assume $y \sim \text{Multinoulli}(\boldsymbol{\phi})$ and $p(x|y) = \prod_i p(x_i|y)$, then our prediction rule is simply
$\underset{y}{\text{argmax}}\, p(y|x) = \underset{y}{\text{argmax}}\, p(x|y)p(y) = \underset{y}{\text{argmax}} \prod_i p(x_i|y)p(y)$

2 Decision Trees

Leaf nodes predict Y or $p(Y|X \in \text{leaf})$. Training time $O(dn \log n)$
Internal nodes test one feature X_i against threshold and each branch selects value for X_i
Decision trees are rule based classifiers that divide feature space into axis-parallel (hyper-)rectangles
Random forests are a bag of decision tress with the additional property that only a subset $k < d$ features are considered at each split. Can determine feature importances by mean impurity decrease

2.1 Entropy

$H(X) = -\sum_{i=1}^{n} p(X=i)\log_2 p(X=i)$ measures impurity/uncertainty of random variable
$H(Y|X) = \sum_{v\in X} p(X=v)H(Y|X=v)$ gives conditional entropy of Y given X
$I(Y,X) = H(Y) - H(Y|X) = I(X,Y)$ gives info-gain/mutual information of Y and X
Assuming sender and receiver know distribution, how many bits on average to transmit one symbol?
$p(X=1) = 1 \implies H(X) = 0$, $p(X=1) = 0.5 \implies H(X) = 1$, $p(X=1) = 0.9 \implies H(X) = 0.136$

2.2 ID3 Algorithm

Starting at root, if entropy is 0 or all feature values are equal, create leaf nodes
Else, choose feature X_i and threshold c based on largest info-gain / minimum conditional entropy
For each value of X_i, create new descendent, sort training examples to descendent nodes and recurse

3 Ensemble Methods

Bootstrap sampling: given S_n with n examples, create S'_n by sampling with replacement n times
On average, a bootstrap sample contains $1 - e^{-1} \approx 0.63$ of the original sample
Bagging reduces variance without increasing bias by averaging results of m independent classifiers
To get independent classifiers, train distinct classifier on each of m bootstrapped samples $S_n^{(1)}, \ldots, S_n^{(m)}$

3.1 Adaboost

Boosting reduces both bias and variance by combining simple weak learners into a strong ensemble
Set $\widetilde{W}_0^{(i)} = \frac{1}{n}$ for $i = 1, \ldots, n$, then for $t = 1, \ldots, m$
Fit $h(x;\hat{\theta}_t)$ to weighted training set $\widetilde{W}_{t-1}^{(i)}$ (for example decision stump $h(x,\theta) = \text{sgn}(\theta_1 x_k + \theta_0)$)
Compute weighted classification error $\hat{\epsilon}_t = \sum_{i=1}^{n} \widetilde{W}_{t-1}^{(i)} 1\{y^{(i)} \neq h(x^{(i)},\hat{\theta}_t)\}$ and score $\hat{\alpha}_t = \frac{1}{2}\ln\frac{1-\hat{\epsilon}_t}{\hat{\epsilon}_t}$
Update weights on training examples $\widetilde{W}_t^{(i)} = c_t \widetilde{W}_{t-1}^{(i)} \exp(-y^{(i)}\hat{\alpha}_t h(x^{(i)};\hat{\theta}_t))$
Return $h_m(x) = \text{sgn}\big(\sum_{i=1}^{m} \hat{\alpha}_t h(x;\hat{\theta}_t)\big)$, where incorrect classifiers will get negative weights
Boosting increases margin aggressively since it concentrates on the hardest examples

4 Principal Component Analysis

Assume data approximately lies on a lower dimensional subspace $\mathcal{Z}$ of $\mathcal{X}$. Then PCA finds a basis for $\mathcal{Z}$ consisting of the top $k < d$ principal components (directions of maximal variance) in $\mathcal{X}$
Normalize data to have 0 mean and unit variance so all features are on the same scale
Use svd to find the top k orthonormal eigenvectors $\{v_1, \ldots, v_k\}$ of $\Sigma = \frac{1}{n}X^T X$ (works since $\Sigma \succeq 0$)
Apply transformation $x^{(i)} \to z^{(i)} = (v_1^T x^{(i)}, \ldots, v_k^T x^{(i)})^T$ and reconstruct $\hat{x}^{(i)} = \sum_{j=1}^{k} z_j^{(i)} v_j$
$J_k = \frac{1}{n}\sum_{i=1}^{n} \|x^{(i)} - \hat{x}^{(i)}\|^2 = \frac{1}{n}\sum_{i=1}^{n} (x^{(i)})^T x^{(i)} - \sum_{j=1}^{k} \lambda_j = \sum_{j=k+1}^{d} \lambda_j$ is the mean squared projection error
Typically choose k such that 99% of the variance is retained, equivalently $\frac{nJ_k}{\sum_{i=1}^{n}\|x^{(i)}\|^2} \leq 0.01$
The variance along unit u is $\frac{1}{n}\sum_{i=1}^{n}(x^{(i)})^T u)^2 = \frac{1}{n}\sum_{i=1}^{n} u^T x^{(i)}(x^{(i)})^T u = u^T \Sigma u$ (assuming 0 mean)
Maximizing the variance, $\mathcal{L}(u,\lambda) = u^T\Sigma u + \lambda(1 - u^T u) \implies \Sigma u = \lambda u \implies u^T \Sigma u = \lambda$
Hence directions of maximal variance are eigenvectors of Σ with variance as the eigenvalue

5 Linear Regression

Assume $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$, $y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$, then $p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$, so

$\ell(\theta) = \log \prod_{i=1}^{n} p(y^{(i)}|x^{(i)}; \theta) = \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \propto -\sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2$

$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{2}(X\theta - y)^T (X\theta - y)$, $\nabla_\theta J(\theta) = X^T X\theta - X^T y$

Note that $\operatorname{rank} X = \operatorname{rank} X^T = \operatorname{rank} XX^T = \operatorname{rank} X^T X$, so θ always has a solution

If $X^T X$ is singular, then θ could have many solutions. Cost of inverting $X^T X$ is $O(d^3)$

6 Logistic Regression

Assume $y|x; \theta \sim \text{Bernoulli}(\phi)$, $\log \frac{\phi}{1-\phi} = \theta^T x$, then $\phi = \frac{1}{1+e^{-\theta^T x}} = h_\theta(x) = \sigma(\theta^T x)$, so

$\ell(\theta) = \log \prod_{i=1}^{n} \phi^{y^{(i)}} (1-\phi)^{1-y^{(i)}} = \sum_{i=1}^{n} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$

$J(\theta) = -y^T \log \sigma(X\theta) - (1 - y^T) \log(1 - \sigma(X\theta))$, $\nabla_\theta J(\theta) = X^T(\sigma(X\theta) - y)$

The sigmoid function is $\sigma(z) = \frac{1}{1+e^{-z}}$ and $\sigma'(z) = \sigma(z)(1 - \sigma(z))$

If $y \in \{-1, +1\}$, then we use logistic loss $L(y, \hat{y}) = \log(1 + e^{-y\hat{y}})$ instead of cross-entropy loss

7 Softmax Regression

Assume $y|x; \boldsymbol{\theta} \sim \text{Multinoulli}(\boldsymbol{\phi})$, $\log \frac{\phi_j}{\phi_k} = \theta_j^T x$, then $\phi_j = \frac{e^{\theta_j^T x}}{\sum_{l=1}^{k} e^{\theta_l^T x}} = h_{\theta_j}(x)$, so

$\ell(\boldsymbol{\theta}) = \log \prod_{i=1}^{n} \prod_{j=1}^{k} \phi_j^{1\{y_j^{(i)}=1\}} = \sum_{i=1}^{n} \sum_{j=1}^{k} 1\{y^{(i)} = j\} \log p(y^{(i)} = j|x^{(i)}; \theta)$

$J(\boldsymbol{\theta}) = -\sum_{j=1}^{k} y_j^T \log \sigma(X\theta_j)$, $\nabla_{\theta_j} J(\boldsymbol{\theta}) = -\sum_{i=1}^{n} x^{(i)}(1\{y^{(i)} = j\} - p(y^{(i)} = j|x^{(i)}; \theta))$

Similar to logistic regression, we see $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = X^T(h_{\boldsymbol{\theta}}(X) - y)$ which is now a $d \times k$ matrix

Note $\sigma(\boldsymbol{z})_j = \frac{\exp(z_j)}{\sum_k \exp(z_k)}$, $\frac{\partial \sigma(\boldsymbol{z})_j}{\partial z_i} = \sigma(\boldsymbol{z})_i(1 - \sigma(\boldsymbol{z})_i)$ if $i = j$ and $\frac{\partial \sigma(\boldsymbol{z})_j}{\partial z_i} = -\sigma(\boldsymbol{z})_i \sigma(\boldsymbol{z})_j$ if $i \neq j$

8 Regularization

Assuming the prior $\theta \sim \mathcal{N}(0, \lambda^{-1} I)$ is equivalent ℓ_2 regularization in the MAP estimate

$$\begin{aligned}
\operatorname*{argmax}_\theta \log p(\theta|D) &= \operatorname*{argmax}_\theta \log p(D|\theta)p(\theta) = \operatorname*{argmax}_\theta \log p(D|\theta) + \log p(\theta) \\
&= \operatorname*{argmax}_\theta \log p(D|\theta) + \log\left(\frac{1}{(2\pi)^{d/2}|\lambda^{-1}I|^{1/2}} \exp\left(-\tfrac{1}{2}\theta^T(\lambda^{-1}I)^{-1}\theta\right)\right) \\
&= \operatorname*{argmax}_\theta \log p(D|\theta) - \tfrac{\lambda}{2}\|\theta_{1:d}\|_2^2
\end{aligned}$$

Assuming the prior $\theta_i \sim \text{Lap}(0, \frac{1}{\lambda})$ is equivalent to ℓ_1 regularization in the MAP estimate

$$\begin{aligned}
\operatorname*{argmax}_\theta \log p(\theta|D) &= \operatorname*{argmax}_\theta \log p(D|\theta) + \log \prod_{i=1}^{d} \tfrac{1}{2(1/\lambda)} \exp(-\tfrac{|\theta_i|}{1/\lambda}) \\
&= \operatorname*{argmax}_\theta \log p(D|\theta) - \lambda\|\theta_{1:d}\|_1
\end{aligned}$$

ℓ_1 regularization results in sparser solutions as it's more likely for the loss function to touch the axis aligned spikes first, but is not differentiable. Laplace prior also places more mass near $\theta_i = 0$

Intuitively prefer smaller weights as large weights are sensitive to small changes in feature

9 Perceptron

Assume data is linearly separable by a hyperplane with normal vector θ, then $h_\theta(x) = \text{sgn}(\theta^T x)$
Recall distance from (x_0, y_0, z_0) to plane is $D = \frac{ax_0+by_0+cz_0+d}{\sqrt{a^2+b^2+c^2}}$ where $\vec{n} = (a, b, c)$
Repeat until convergence for $i = 1$ to n, if $y^{(i)}\theta^T x^{(i)} \leq 0$, then $\theta \leftarrow \theta + y^{(i)}x^{(i)}$
Intuition is that θ tilts more towards $x^{(i)}$ if it was positive and more away if negative
$J(\theta) = \sum_{i=1}^{n} \max(0, -y^{(i)}\theta^T x^{(i)})$, then for single example $-\nabla_\theta J_i(\theta) = y^{(i)}x^{(i)}$
Geometric margin of ith example $\gamma^{(i)} = \frac{y^{(i)}(\theta^T x+b)}{\|\theta\|} = \frac{\hat{\gamma}^{(i)}}{\|\theta\|}$ is signed distance to hyperplane

9.1 Perceptron Convergence Theorem

If data is linearly separable with margin $\gamma = \min_i \gamma^{(i)}$ by unit norm θ^* and $\|x^{(i)}\| \leq R$ for all i,
then the perceptron converges after at most $\frac{R^2}{\gamma^2}$ updates.

10 Support Vector Machines

$\min_{\theta,b,\xi} \frac{1}{2}\|\theta\|^2 + C\sum_{i=1}^{n} \xi_i$ where $\xi_i = \max(0, 1 - y^{(i)}(\theta^T x^{(i)} + b))$ is the primal and the dual is
$\max_\alpha \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y^{(i)}y^{(j)}\langle x^{(i)}, x^{(j)}\rangle$ such that $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{n} \alpha_i y^{(i)} = 0$
From KKT conditions $\alpha_i = 0 \implies y^{(i)}(\theta^T x^{(i)} + b) > 1$, $\alpha_i = C \implies y^{(i)}(\theta^T x^{(i)} + b) < 1$ and
$0 < \alpha_i < C \implies y^{(i)}(\theta^T x^{(i)}+b) = 1$, where an upper bound of C limits α_i to allow misclassifications
$J(\theta) = \sum_{i=1}^{n} \xi_i = \sum_{i=1}^{n} \max(0, 1 - y^{(i)}(\theta^T x^{(i)} + b))$ hinge loss is encoded in slack variables
$\theta = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)}$, $\hat{y} = \sum_{i=1}^{n} \alpha_i y^{(i)}\langle x^{(i)}, x\rangle + b$, where $b = y^{(i)}(1 - \xi_i) - \theta^T x^{(i)}$ for $0 < \alpha_i < C$
ξ_i is zero only when classifier is correct with confidence $y^{(i)}(\theta^T x^{(i)} + b) \geq 1 = \hat{\gamma} = \gamma\|\theta\|$
Dual learns weights to s.vs and predicts test example by a weighted similarity to s.vs
$C \propto \frac{1}{\lambda}$. High C means less misclassifications, lower margin, more s.vs, higher model complexity
In practice, prefer low C so training is faster. Train time roughly $O(n^2 d)$ rbf and $O(nd)$ linear

10.1 Kernels

Since prediction and optimization only depend on inner products, lets map $x^{(i)} \to \phi(x^{(i)})$
Computing $\phi(x^{(i)})$ is inefficient, thus find implicit mapping (kernel) where $k(x^{(i)}, x^{(j)}) = \langle\phi(x^{(i)}), \phi(x^{(j)})\rangle$
Kernels redefine inner product as if we had found the corresponding mapping to a higher dimension
$k(x, z) = (1 + \langle x, z\rangle)^2 \implies \phi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2)^T$ "maps" x to quadratic space
If $k_1(x, z)$, $k_2(x, z)$ are valid kernels, then $k(x, z) = f(x)k_1(x, z)k_2(x, z) + k_2(x, z)$ is a valid kernel
$k(x, z) = \exp(-\gamma\|x - z\|^2)$, $\gamma > 0$ places a Gaussian around every support vector, hence $\gamma \propto \frac{1}{\lambda}$
$\exp(-\frac{1}{2}\|x - z\|^2) = \exp(-\frac{1}{2}x^T x)\exp(x^T z)\exp(-\frac{1}{2}z^T z)$, $\exp(x^T z) = 1 + \frac{x^T z}{1!} + \frac{(x^T z)^2}{2!} + \ldots$
$k(x, z) = \tanh(\gamma\langle x, z\rangle + r)$ is sigmoid kernel and $k(x, z) = (\gamma\langle x, z\rangle + r)^d$ is polynomial kernel

10.2 Mercer's Theorem

k is a valid kernel $\iff$ $K \succeq 0$ for any $\{x^{(i)}\}_{i=1}^n$, where $K_{ij} = k(x^{(i)}, x^{(j)}) = \langle\phi(x^{(i)}), \phi(x^{(j)})\rangle$
$K \succeq 0 \implies K_{ij} = e_i^T PDP^T e_j = (e_i^T PD^{\frac{1}{2}})(D^{\frac{1}{2}}P^T e_j) = (D^{\frac{1}{2}}P^T e_i)^T(D^{\frac{1}{2}}P^T e_j) = \langle\phi(x^{(i)}, \phi(x^{(j)})\rangle$
$K_{ij} = \langle\phi(x^{(i)}, \phi(x^{(j)})\rangle \implies x^T Kx = \sum_{ij} x_i x_j K_{ij} = \langle\sum_i x_i\phi(x^{(i)}), \sum_j x_j\phi(x^{(j)})\rangle \geq 0$

11 Evaluating Performance

k-fold cross validation: partition training set into k folds, choose each fold in turn as validation set and train model on remaining $k-1$ folds. Leave-one-out cv special case when $k=n$
In practice, can do multiple trials of cv and bootstrap test set to compute statistics using t-test
Precision $= \frac{\text{TP}}{\text{TP + FP}} = \mathbb{P}(y^{(i)} = 1 | h(x^{(i)}) = 1)$, recall $= \frac{\text{TP}}{\text{TP + FN}} = \mathbb{P}(h(x^{(i)}) = 1 | y^{(i)} = 1)$
TPR $= \mathbb{P}(h(x^{(i)}) = 1 | y^{(i)} = 1)$, FPR $= \mathbb{P}(h(x^{(i)}) = 0 | y^{(i)} = 1)$, TNR $= \mathbb{P}(h(x^{(i)}) = 0 | y^{(i)} = 0)$
AUROC: area under ROC curve (plot of TPR vs FPR). $F_\beta = \frac{(1+\beta^2) \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision + recall}}$
Given classifier that outputs confidence, generate ROC or PR curve by varying threshold

11.1 Algorithm Analysis

Learning Curves: plot J_{train} and J_{test} for different training set sizes to determine high bias/variance
Error analysis: plug-in ground truth for each component and see how error changes
Ablative analysis: remove components from system to see how it breaks

11.2 Imbalanced and Multiclass

Subsampling: randomly pick k majority class examples and include all k minority class examples
Oversampling: include all m majority class examples and sample minority class with replacement
Give minority class examples larger weight (classifier must be able to handle sample weights)
One-vs-all: train m classifiers and pick most confident positive or $\hat{y} = \text{argmin}_y \sum_{j=1}^{k} \text{Loss}(R(y, j) h_j(x))$
One-vs-one: train $\binom{m}{2}$ binary classifiers and pick class with largest number of positive votes

12 Optimization

Batch gradient descent does one step using n examples: $\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$
Stochastic gradient descent does one step *per* example: $\theta \leftarrow \theta - \alpha \nabla_\theta J_i(\theta)$
Learning rate $\alpha_k = \frac{1}{1+k}$ where $k =$ outer loop iteration is typically a good choice

12.1 Newton's Method

Iteratively find better approximations for roots of differentiable function by $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
In optimization, find critical points of twice differentiable function by $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma \boldsymbol{H} f(\boldsymbol{x}_n)^{-1} \nabla f(\boldsymbol{x}_n)$
Often converges in fewer steps than gradient descent, but computing $\boldsymbol{H}_f^{-1}$ is costly

12.2 Expectation Maximization

Given a set of observable X and latent Z, directly solving $\ell(\theta) = \sum_z \log p(x, z; \theta)$ is often intractable
Rewite $\ell(\theta) = \sum_z \log p(x, z; \theta) = \sum_z \log q(z|x; \theta) \frac{p(x,z;\theta)}{p(z|x;\theta)} \geq \sum_z q(z|x; \theta) \log \frac{p(x,z;\theta)}{q(z|x;\theta)} \equiv J(q, \theta)$
Applied Jensen's inequality since log is concave. Furthermore, let $q(z|x; \theta) = p(z|x; \theta)$
Instead of maximizing $\ell(\theta)$ directly, EM maximizes the lower bound $J(q, \theta)$ via coordinate ascent
Since $\ell(\theta) \geq J(q, \theta) = \sum_z p(z|x; \theta) \log \frac{p(x,z;\theta)}{p(z|x;\theta)} = \sum_z p(z|x; \theta) \log p(x, z; \theta) + H(p(z|x; \theta))$
Maximizing $J(q, \theta)$ is equivalent to maximizing the expected complete log-likelihood, thus
[E-step]: Compute $Q(\theta|\theta^{(t)}) = E_{p(z|x;\theta^{(t)})}[\log p(x, z; \theta)] = \sum_z p(z|x; \theta) \log p(x, z; \theta)$
[M-step]: Set $\theta^{(t+1)} = \underset{\theta}{\text{argmax}}\, Q(\theta|\theta^{(t)})$. Guaranteed to increase likelihood every iteration

13 Clustering

Organize data into clusters such that there is high intra-cluster and low inter-cluster similarity
The ℓ_p norm $\|x\|_p = \big(\sum_{i=1}^{n} |x_i|^p\big)^{1/p}$ implicitly defines an inner product, where $\|x\|_2 = \sqrt{x^T x}$

13.1 k-Means

Initialize k cluster centroids randomly $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$, then repeating until convergence
for $i = 1, \ldots, n$, set $c^{(i)} = \underset{j}{\text{argmin}} \|x^{(i)} - \mu_j\|^2$ minimizes J w.r.t c
for $j = 1, \ldots, k$, set $\mu_j = \frac{\sum_{i=1}^{n} 1\{c^{(i)}=j\} x^{(i)}}{\sum_{i=1}^{n} 1\{c^{(i)}=j\}}$ minimizes J w.r.t μ
k-means is coordinate descent on the distortion function $J(c,u) = \sum_{i=1}^{n} \|x^{(i)} - \mu_{c^{(i)}}\|^2$
k-medians is coordinate descent on $J(c,u) = \sum_{i=1}^{n} \|x^{(i)} - \mu_{c^{(i)}}\|_1$ and is more robust to outliers
k-medoids updates cluster centroid to be $\mu_j = \text{argmin}_{y \in \{x^{(1)}, \ldots, x^{(n)}\}} \sum_{i=1}^{n} d(y, x^{(i)})$
Choose number of clusters by varying k and using elbow method on J
Since J is non-convex, repeat k-means multiple times and choose clustering with lowest distortion

13.2 Hierarchical Clustering

Agglomerative clustering: bottom-up, every observation starts in its own cluster, find best pair to merge into new cluster, repeat until all clusters are merged. Produces dendrogram of clusterings
Single linkage: use closest pair, results in potentially long and skinny clusters
Complete linkage: use farthest pair, results in tight (small, round) clusters
Average linkage uses average of all pairs and is robust against noise (linkage defines cluster distance)
If $\text{dist}(A, B) = 2$, then create parent node AB with two branches of distance 1. Runtime $O(n^3)$

13.3 Gaussian Mixture Models

Assume $z^{(i)} \sim \text{Categorial}(\phi)$, $\phi_j = p(z^{(i)} = j)$, $\sum_{j=1}^{k} \phi_j = 1$, $x^{(i)}|z^{(i)} = j \sim \mathcal{N}(\mu_j, \Sigma_j)$, then
$\ell(\phi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{n} \log p(x^{(i)}; \phi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{n} \log \sum_{z^{(i)}=1}^{k} p(x^{(i)}|z^{(i)}; \phi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, using EM
$w_j^{(i)} = p(z^{(i)} = j|x^{(i)}; \phi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{p(x^{(i)}|z^{(i)}=j; \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z^{(i)}=j; \phi)}{\sum_{l=1}^{k} p(x^{(i)}|z^{(i)}=l; \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z^{(i)}=l; \phi)}$ [E-step]
$\mu_j = \frac{\sum_{i=1}^{n} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{n} w_j^{(i)}}$, $\Sigma_j = \frac{\sum_{i=1}^{n} w_j^{(i)} (x^{(i)} - \mu_j)^T (x^{(i)} - \mu_j)}{\sum_{i=1}^{n} w_j^{(i)}}$, $\phi_j = \frac{1}{n} \sum_{i=1}^{n} w_j^{(i)}$ [M-step]
[E-step] viewed as soft clustering examples and [M-step] viewed as weighted update of cluster centers
Choose number of clusters k by minimizing Bayesian Information Criteria: $\ln(n)k - 2\ln(\hat{L})$

13.4 Cluster Metrics

External Validation: given labels, could find weighted purity or entropy in each cluster
Internal Validation: without labels, clusters should be stable to subsampling of data
For dataset X, define $L(X)$ such that $L_{ij} = 1$ if $x^{(i)}$ and $x^{(j)}$ are in the same cluster else 0
Compute $\text{sim}(L(X), L(X'))$ where X' is a subsample and comparison done over common examples
Cluster Cohesion: $\sum_{i=1}^{n} \|\boldsymbol{x^{(i)}} - \boldsymbol{\mu}_{c^{(i)}}\|^2$, Cluster Separation: $\sum_{j=1}^{k} |C_i| \|\boldsymbol{\mu} - \boldsymbol{\mu_j}\|^2$
Silhouette Coefficient: for individual $x^{(i)}$, let a = average distance of $x^{(i)}$ to points in its cluster and $b = \min$(average distance of $x^{(i)}$ to points in another cluster), want $s = 1 - a/b$ close to 1

14 Hidden Markov Models

Stochastic process: a collection or r.vs that evolve over time
Markov property: conditional distribution of future states depends only on the current state
A Markov chain is a stochastic process with the Markov property modeled by $\lambda = (S, \pi, A)$ and outputs observable state $O_t \in S$ such that $p(O|\lambda) = p(q_1, ..., q_T) = p(q_1)p(q_2|q_1)...p(q_T|q_{T-1})$
Hidden Markov Models assume there is an underlying markov chain that generates observations
$S = \{S_1, ..., S_N\}$ where the states are $q_t \in S$, $V = \{v_1, ..., v_m\}$ where the observations are $O_t \in V$
$\pi = \{\pi_i\}$ where $\pi_i = p(q_1 = S_i)$, $A = \{a_{ij}\}$ where $a_{ij} = p(q_t = S_j | q_{t-1} = S_i)$ and
$B = \{b_j(k)\}$ where $b_j(k) = p(v_k \text{ at t} | q_t = S_j)$ are the elements of a hidden markov model

14.1 Forward Algorithm

Find probability of seeing observation sequence $p(O|\lambda) = \sum_{q_T} p(q_T, O|\lambda)$
Define forward variable $\alpha_t(i) = p(O_1, O_2, ..., O_t, q_t = S_i|\lambda)$
Initialize $\alpha_1(i) = \pi_i b_i(O_1)$ for $1 \leq i \leq N$
Induct $\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1})$ for $1 \leq t \leq T-1$ and $1 \leq j \leq N$
Terminate $p(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$ with $O(N^2T)$ time and $O(NT)$ space complexity
Compare with brute force $p(O|\lambda) = \sum_q p(q_1)p(O_1|q_1)p(q_2|q_1)\dots p(O_T|q_T)$ which is $O(N^T T)$

14.2 Backward Algorithm

Define backward variable $\beta_t(i) = p(O_{t+1}, O_{t+2}, ..., O_T | q_t = S_i, \lambda)$
Initialize $\beta_T(i) = 1$ for $1 \leq i \leq N$
Induct $\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(O_{t+1})$ for $T-1 \geq t \geq 1$ and $1 \leq i \leq N$
Terminate $p(O|\lambda) = \sum_{i=1}^{N} \alpha_1(i)\beta_1(i) = \sum_{i=1}^{N} \pi_i b_i(O_1)\beta_1(i)$

14.3 Posterior Decoding

Find most likely state at every point in time. Want $q^* = \{q_t | q_t = \text{argmax}_{S_i} \, p(q_t = S_i | O, \lambda)\}$
Define $\gamma_t(i) = p(q_t = S_i|O, \lambda) = \frac{p(O, q_t = S_i|\lambda)}{p(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$

14.4 Viterbi Decoding

Find most likely path q^* given observation sequence. Want $q^* = \text{argmax}_q \, p(q|O, \lambda) = \text{argmax}_q \, p(q, O|\lambda)$
Define Viterbi probability $\delta_t(i) = \max_{q_1, ..., q_{t-1}} p(q_1, ..., q_{t-1}, O_1, ..., O_t, q_t = S_i|\lambda)$
Initialize $\delta_1(i) = \pi_i b_i(O_1)$ for $1 \leq i \leq N$
Induct $\delta_{t+1}(j) = \left[\max_{1 \leq i \leq N} \delta_t(i) a_{ij}\right] b_j(O_{t+1})$, $\psi_{t+1}(j) = \underset{1 \leq i \leq N}{\text{argmax}} \, \delta_t(i) a_{ij}$ until $t = T-1$, $1 \leq j \leq N$
Terminate $p^* = \max_{1 \leq i \leq N} \delta_T(i)$, $q_T^* = \underset{1 \leq i \leq N}{\text{argmax}} \, \delta_T(i)$ with backtracking $q_t^* = \psi_{t+1}(q_{t+1}^*)$ until $t = 1$

14.5 Learning HMMs

Adjust model parameters $\lambda = (A, B, \pi)$ to maximize $p(O|\lambda)$
Given complete data (we know the underlying states). Use MLE
$\hat{\pi}_i = \frac{\text{Count}(q_i)}{\sum_{j=1}^{N} \text{Count}(q_j)}$, $\hat{a}_{ij} = \frac{\text{Count}(S_i \to S_j)}{\text{Count}(S_i)}$, $\hat{b}_j(k) = \frac{\text{Count}(S_j \to v_k)}{\text{Count}(S_j)}$
Add pseudocounts to represent prior belief. Large pseudocounts $\to$ large regularization

14.5.1 Unsupervised Learning. Baum-Welch

Given number of states T, EM guaranteed to increase likelihood $p(O|\lambda)$ with complexity $O(N^2T)$
Define $\xi_t(i,j) = p(q_t = S_i, q_{t+1} = S_j|O,\lambda) = \frac{p(q_t=S_i,q_{t+1}=S_j,O|\lambda)}{p(O|\lambda)} = \frac{\alpha_t(S_i)a_{ij}b_j(O_{t+1})\beta_{t+1}(S_j)}{p(O|\lambda)}$
Compute $\alpha_t(i)$, $\beta_t(i)$, $p(O|\lambda)$ using forward-backward algorithm to get $\gamma_t(i)$ and $\xi_t(i,j)$ [E-step]
Update model parameters: $\hat{\pi}_i = \gamma_1(i)$, $\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}$, $\hat{b}_j(k) = \frac{\sum_{t=1 \text{ s.t } O_t=v_k}^{T}\gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(j)}$ [M-step]

15 Learning Theory

Let $R_n(h)$ be the training error and $R(h)$ be the expected error of $h \in \mathcal{H}$

15.1 Haussler's Theorem

Given finite hypothesis space $\mathcal{H}$, dataset $\mathcal{D}$ with n iid training and $0 \leq \epsilon \leq 1$, then for any learned hypothesis h that is consistent with the training data, $p(R(h) > \epsilon) \leq |\mathcal{H}|e^{-n\epsilon}$

15.2 Hoeffding Inequality

Let $Z_1, Z_2, ..., Z_n$ be n iid r.vs drawn Bernoulli(ϕ) and $\hat{\phi} = \frac{1}{n}\sum_{i=1}^{n} Z_i$ be the mean of the Z_i's
Then for any $\gamma > 0$, $p(|\phi - \hat{\phi}| > \gamma) \leq 2e^{-2n\gamma^2}$
Using the union bound we can derive $p(R(h) - R_n(h) > \epsilon) \leq |\mathcal{H}|e^{-2n\epsilon^2}$ for any $h \in \mathcal{H}$

15.3 PAC Bounds

Suppose we are willing to tolerate at most a δ probability of having $> \epsilon$ error, then
$p(R(h) > \epsilon) \leq |\mathcal{H}|e^{-n\epsilon} \leq \delta \iff p(R(h) \leq \epsilon) \geq 1 - |\mathcal{H}|e^{-n\epsilon} \geq 1 - \delta$, so for consistent h we need $n \geq \frac{1}{\epsilon}(\ln|\mathcal{H}| + \ln\frac{1}{\delta})$ to be probably (with probability $1-\delta$) and approximately correct (with error ϵ)
We can also bound the true error by letting $|\mathcal{H}|e^{-n\epsilon} = \delta$, in which case $\epsilon = \frac{1}{n}(\ln|\mathcal{H}| + \ln\frac{1}{\delta})$
For nonconsistent h we have, $n \geq \frac{1}{2\epsilon^2}\big(\ln|\mathcal{H}| + \ln(\frac{1}{\delta})\big)$ and $\epsilon = \sqrt{\frac{1}{2n}\big(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\big)}$

15.4 Bias-Variance Tradeoff

With probability at least $1-\delta$, $R(h) \leq R_n(h) + \sqrt{\frac{1}{2n}\big(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\big)}$. If we increase $|\mathcal{H}|$, then we can find a better $h \in \mathcal{H}$ to lower bias $R_n(h)$ but at the cost of higher variance ϵ

15.5 VC-Dimension

A set $S = \{x^{(1)}, ..., x^{(m)}\}$ of points $x^{(i)} \in \mathcal{X}$ can be shattered by hypothesis class $\mathcal{H}$ if and only if for any set of labels $\{y^{(1)}, ..., y^{(m)}\}$, there exists a consistent $h \in \mathcal{H}$
The VC-dimension over $\mathcal{X}$ is the size or the largest set of points in $\mathcal{X}$ that is shattered by $\mathcal{H}$
$\text{VC}(\mathcal{H}) = d + 1$ where $\mathcal{H}$ is the set of linear classifiers with bias in $\mathbb{R}^d$
If $|\mathcal{H}| = \infty$ but $\text{VC}(\mathcal{H}) = d$ in $\mathcal{X}$, then $R(h) \leq R_n(h) + \sqrt{\frac{1}{2n}\big[d\big(\ln\frac{2n}{d} + 1\big) + \ln\frac{4}{\delta}\big]}$

15.5.1 VC($\mathcal{H}$) for SVMs

Let $\mathcal{H}_\gamma$ be the set of classifiers in $\mathbb{R}^d$ with margin γ on $\mathcal{X}$ where each $x^{(i)} \in \mathcal{X}$ satisfies $\|x^{(i)}\| \leq R$
Then $\text{VC}(\mathcal{H}_\gamma) \leq \min\left\{d, \frac{4R^2}{\gamma^2}\right\}$