Lecture 1 - Maximum Likelihood Tom Zollo - **ColumbiaX ML 102 Study Guide**

- Maximum Likelihood equation
  - The analytic criterion for this maximum likelihood estimator is:
    - $$\nabla_\theta \prod_{i=1}^{n} p(x_i|\theta) = 0.$$
    - Find parameters μ, Σ that most likely generated data
  - So if we have data $x_1, \ldots, x_n$ in R^d that we hypothesize is i.i.d. Gaussian, the maximum likelihood values of the mean and covariance matrix are:
    - $$\hat{\mu}_{\text{ML}} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \hat{\Sigma}_{\text{ML}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu}_{\text{ML}})(x_i - \hat{\mu}_{\text{ML}})^T$$

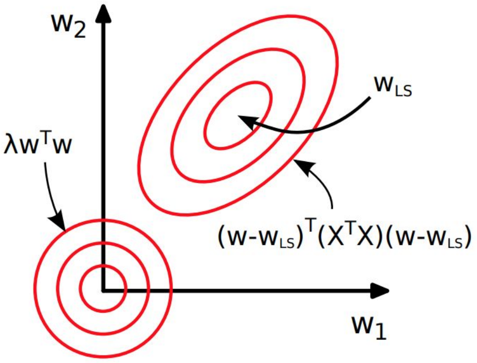
Lecture 2 - Linear Regression and Least Squares

- Goal: Find a function $f : R^d \rightarrow R$ such that $y = f(x; w)$ for the data pair (x, y). $f(x; w)$ is called a regression function. Its free parameters are w.
- A regression method is called linear if the prediction f is a linear function of the unknown parameters w.
  - $$y_i \approx f(x_i; w) = w_0 + \sum_{j=1}^{d} x_{ij} w_j$$
- Some notation (adding 1's to account for w_0)
  - $$x_i = \begin{bmatrix} 1 \\ x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1d} \\ 1 & x_{21} & \dots & x_{2d} \\ \vdots & & \vdots & \\ 1 & x_{n1} & \dots & x_{nd} \end{bmatrix} = \begin{bmatrix} 1 - & x_1^T & - \\ 1 - & x_2^T & - \\ & \vdots & \\ 1 - & x_n^T & - \end{bmatrix}$$
- Least squares objective: (minimize sum of squared errors)
  - $$w_{\text{LS}} = \arg\min_w \sum_{i=1}^{n}(y_i - f(x_i; w))^2 \equiv \arg\min_w \mathcal{L}$$
- Vector version
  - $$\mathcal{L} = \textstyle\sum_{i=1}^{n}(y_i - x_i^T w)^2 \qquad w_{\text{LS}} = \Big(\sum_{i=1}^{n} x_i x_i^T\Big)^{-1}\Big(\sum_{i=1}^{n} y_i x_i\Big)$$
- Matrix Version
  - $$\arg\min_w \|y - Xw\|^2 \quad \Rightarrow \quad w_{\text{LS}} = (X^T X)^{-1} X^T y.$$
  - Minimizes the squared Euclidean distance between output vector y and prediction vector Xw
  - Matrix version only works if n>d
  - $X^T X$ must be full rank
    - Full rank: at least as many linearly independent rows or columns as lesser of the number of rows and columns
- Given x_{new}, the least squares prediction for y_{new} is:
  - $$y_{\text{new}} \approx x_{\text{new}}^T w_{\text{LS}}$$
- Polynomial Regression

    - $y = w_0 + w_1 x + w_2 x^2$
    - Still linear in w, same solution only the preprocessing is different
- Geometry of Least Squares
    - $$\arg\min_{w} \|y - Xw\|^2 \quad \Rightarrow \quad w_{\text{LS}} = (X^T X)^{-1} X^T y.$$
    - The columns of X define a d + 1-dimensional subspace in the higher dimensional R^n.
    - The closest point to y in that subspace is the orthonormal projection of y into the column space of X.

Lecture 3 - Least Squares and Ridge Regression

- Probabilistic view of least squares
    - We are making an independent Gaussian noise assumption about the error, $E_i = y_i - x^T_i w$
        - Error = Measured - Predicted
    - Other ways of saying this
        - $$1)\ \ y_i = x_i^T w + \epsilon_i, \quad \epsilon_i \overset{iid}{\sim} N(0, \sigma^2), \quad \text{for } i = 1, \ldots, n,$$
          $$2)\ \ y_i \overset{ind}{\sim} N(x_i^T w, \sigma^2), \quad \text{for } i = 1, \ldots, n,$$
          $$3)\ \ y \sim N(Xw, \sigma^2 I), \ \text{as on the previous slides.}$$
    - Under the Gaussian assumption y ~ N(Xw, σ^2I):
        - $E[w_{ML}] = w$
            - w_{ML} is an unbiased estimate of w: $E[w_{ML}] = w$
        - $Var[w_{ML}] = \sigma^2(X^TX)^{-1}$
            - If $Var[w_{ML}]$ is large, predictions are sensitive to measured data (bad generalization)
- Regularized Least Squares
    - $w_{OPT} = \text{argmin}_w \, ||y - Xw||^2 + \lambda\, g(w)$
        - λ > 0 : a regularization parameter
        - g(w) > 0 : a penalty function that encourages desired properties about w.
- Ridge Regression
    - $w_{RR} = \text{argmin}_w \, ||y - Xw||^2 + \lambda\, ||w||^2$
        - The term $g(w) = ||w||^2$ penalizes large values in w
        - $$\text{Case } \lambda \to 0 \quad : \quad w_{\text{RR}} \to w_{\text{LS}}$$
          $$\text{Case } \lambda \to \infty \ : \ w_{\text{RR}} \to \vec{0}$$
    - $w_{RR} = (\lambda I + X^TX)^{-1}X^Ty$
        - Assume pre-processing (so no 1's):
            - Subtract mean off y
            - Standardize X (subtract mean, divide by stdev)



- Ridge Regression vs. Least Squares
    - $w_{LS} = (X^TX)^{-1}X^Ty \Leftrightarrow wRR = (\lambda I + X^TX)^{-1}X^Ty$
    - $||w_{RR}||_2 \leq ||w_{LS}||_2$
        - Because λ pulls w towards origin

Lecture 4 - Bias/Variance, Bayes Rule and MAP Inference

- Least Squares (λ = 0) - low bias, high variance
- Ridge regression (λ > 0) - high bias, low variance
- Generalization error = measurement noise + bias + variance

- K-fold cross-validation:
    - 1. Randomly split the data into K roughly equal groups.
    - 2. Learn the model on K − 1 groups and predict the held-out Kth group.
    - 3. Do this K times, holding out each group once.
    - 4. Evaluate performance using the cumulative set of predictions.
    - Can be used to tune parameters like λ
    - The data you test the model on should never be used to train the model!
- Bayes Rule
    - $\underbrace{P(B|A)}_{posterior} = \underbrace{P(A|B)}_{likelihood} \underbrace{P(B)}_{prior} \;/\; \underbrace{P(A)}_{marginal}$
    - Bayes rule generalizes to continuous-valued random variables as follows; instead of probabilities we work with densities.
        - Let θ be a continuous-valued model parameter.
        - Let X be data we possess. Then by Bayes rule,
            - $$p(\theta|X) \;=\; \frac{p(X|\theta)p(\theta)}{\int p(X|\theta)p(\theta)d\theta} \;=\; \frac{p(X|\theta)p(\theta)}{p(X)}$$
                - Able to integrate out because area under a probability curve is 1
- Maximum A Priori estimation
    - $w_{\text{MAP}} \;=\; \arg\max_{w} \; \ln p(w|y, X)$
        - Maximizes likelihood of parameters given data
    - The solution is $w_{\text{MAP}} = (\lambda\sigma^2 I + X^T X)^{-1} X^T y$
        - Notice that $w_{MAP} = w_{RR}$ (modulo a switch from λ to $\lambda\sigma^2$)
        - RR maximizes the posterior p(w|y,X) (parameters)
        - LS maximizes the likelihood p(y|X,w) (labels)

Lecture 5 - Bayesian Linear Regression

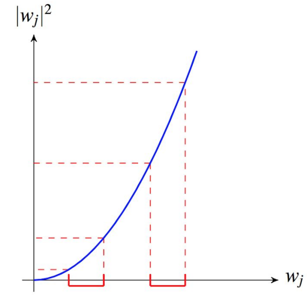
- Bayesian Linear Regression
    - Have a vector $y \in R^n$ and covariates matrix $X \in R^{n \times d}$. The ith row of y and X correspond to the ith observation (y_i, x_i).
        - Note X is not a list of measured data like normal linear regression
    - In a Bayesian setting, we model this data as:
        - $$\begin{aligned} \textbf{Likelihood}: &\quad y \sim N(Xw, \sigma^2 I) \\ \textbf{Prior}: &\quad w \sim N(0, \lambda^{-1} I) \end{aligned}$$
    - The unknown model variable is $w \in R^d$
    - MAP inference returns the maximum of the log joint likelihood:
        - Joint likelihood: p(y,w|X) = p(y|w,X)p(w)
    - Also maximizes posterior of w:
        - $w_{MAP} = \text{argmax}_w$ ln p(w|y, X)
- Point Estimates
    - w_{MAP} and w_{ML} are referred to as point estimates of the model parameters.
    - They find a specific value (point) of the vector w that maximizes an objective function — the posterior (MAP) or likelihood (ML)
        - ML: Only considers the data model: p(y|w, X)
        - MAP: Takes into account model prior: p(y, w|X) = p(y|w, X)p(w)

    - Bayesian inference characterizes uncertainty about w
- Bayesian Inference for Linear Regression
    - The posterior distribution of w is:
        - p(w|y, X) = N(w|μ, Σ)
        - $$\begin{aligned} \Sigma &= (\lambda I + \sigma^{-2} X^T X)^{-1}, \\ \mu &= (\lambda \sigma^2 I + X^T X)^{-1} X^T y \quad \Leftarrow \quad w_{\text{MAP}} \end{aligned}$$
    - The posterior p(w|y, X) is perhaps most useful for predicting new data
        - Predictive distribution:
            - $$p(y_0|x_0, y, X) = \int_{\mathbb{R}^d} \underbrace{p(y_0|x_0, w)}_{likelihood} \underbrace{p(w|y, X)}_{posterior} \, dw$$
            - Prediction ≈ likelihood of y_0 x posterior on w
        - We know from the model and Bayes rule that
            - Model: $p(y_0|x_0, w) = N(y_0|x^T_0 w, \sigma^2)$
            - Bayes rule: p(w|y, X) = N(w|μ, Σ)
        - The predictive distribution can be calculated exactly with these distributions. Again we get a Gaussian distribution:
            - $$\begin{aligned} p(y_0|x_0, y, X) &= N(y_0|\mu_0, \sigma_0^2) \\ \mu_0 &= x_0^T \mu, \\ \sigma_0^2 &= \sigma^2 + x_0^T \Sigma x_0. \end{aligned}$$
- Active Learning
    - Prior => Posterior => Prior
    - $p(w|y_0, x_0, y, X) \propto p(y_0|w, x_0) p(w|y, X)$
        - The posterior after (y, X) has become the prior for (y_0, x_0)
    - $$p(y_0|x_0, y, X) = N(y_0|\mu_0, \sigma_0^2)$$
        - Pick x_0 for which σ^2 is largest and measure y, maximize uncertainty reduction
    - Entropy (i.e., uncertainty) minimization
    - Differential entropy: $H(p) = -\int p(z) \ln p(z)\, dz$
        - This is a measure of the spread of the distribution. More positive values correspond to a more “uncertain” distribution (larger variance)
- Selecting Lambda
    - Evidence maximization:
        - $$\hat{\lambda} = \arg\max_{\lambda} \; \ln p(y|X, \lambda)$$

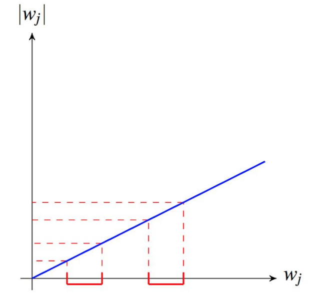
Lecture 6 - Sparse Linear Regression

- Consider the regression problem y = Xw where $X \in R^{n\times d}$ is “fat” (i.e., d>>n)
    - This is called an “underdetermined” problem
    - There are more dimensions than observations, w now has an infinite number of solutions
- Least Norm Solution
    - $w_{ln} = X^T(XX^T)^{-1}y \Rightarrow Xw_{ln} = XX^T(XX^T)^{-1}y = y$
        - We can construct another solution by adding to w_{ln} a vector $\delta \in R^d$ that is in the null space N of X
        - In fact, there are an infinite number of possible δ, because d > n
        - We can show that w_{ln} is the solution with smallest L2 norm

- Least squares and ridge regression treat all dimensions equally without favoring subsets of dimensions
    - The relevant dimensions are averaged with irrelevant ones
    - Poor generalization to new data, interpretability of results
- Regression with penalties
    - Cost = Goodness of fit + Penalty
    - Quadratic penalty: Reduction in cost depends on $|w_j|$
- Feature selection => encourage sparsity
    - Select a small subset of the d dimensions and switch off the rest
    - Quadratic v. Sparse Penalty on ||w||
        - Suppose w_k is large, all other w_j are very small but non-zero
            - Sparsity penalty: should keep w_k, and push other w_j to zero
            - Quadratic penalty: should favor entries w_j which all have similar size, and so it will push w_k towards small value
        - Overall, a quadratic penalty favors many small, but non-zero values



Reducing a large value w_j achieves a larger cost reduction.



Cost reduction does not depend on the magnitude of w_j.

- LASSO: Least Absolute Shrinkage and Selection Operator
    - With the LASSO, we replace the L2 penalty with an L1 penalty
    - $w_{\text{lasso}} = \arg\min_{w} \|y - Xw\|_2^2 + \lambda\|w\|_1$, where $\|w\|_1 = \sum_{j=1}^{d} |w_j|$
- LP
    - norms
        - $$\|w\|_p = \left(\sum_{j=1}^{d} |w_j|^p\right)^{\frac{1}{p}} \qquad \text{for } 0 < p \leq \infty$$
    - regression
        - $$w_{\ell_p} := \arg\min_{w} \|y - Xw\|_2^2 + \lambda\|w\|_p^p$$
    - P = 1 encourages sparsity
    - P < 1 sparse, non-convex

Lecture 7 - Nearest Neighbors and Bayes Classifiers

- Classification => maps input to a **discrete** output space (vs. regression's continuous output space)
- Nearest neighbors => return label y_i of point x_i closest to x_{new}
- Distance measures:
    - The default distance for data in $\mathbb{R}^d$ is the Euclidean one:

      $$\|u - v\|_2 = \left(\sum_{i=1}^{d}(u_i - v_i)^2\right)^{\frac{1}{2}} \quad \text{(line-of-sight distance)}$$

      But there are other options that may sometimes be better:

      ▸ ℓ_p for $p \in [1, \infty]$: $\|u - v\|_p = \left(\sum_{i=1}^{d} |u_i - v_i|^p\right)^{\frac{1}{p}}$.
- K-Nearest Neighbors:
    - Return K-closest points, label determined by majority vote
    - Smaller K leads to smaller training error
    - Larger K leads to more stable predictions due to voting
- Optimal classifiers:

- For any classifier f: X → Y, its prediction error is

  $$P(f(X) \neq Y) = \mathbb{E}[\mathbb{1}(f(X) \neq Y)] = \mathbb{E}[\underbrace{\mathbb{E}[\mathbb{1}(f(X) \neq Y)\,|X]}_{\text{a random variable}}] \qquad (\dagger)$$

- For each x ∈ X ,

  $$\mathbb{E}[\mathbb{1}(f(X) \neq Y)\,|\,X = x] = \sum_{y \in \mathcal{Y}} P(Y = y\,|\,X = x) \cdot \mathbb{1}(f(x) \neq y), \qquad (\ddagger)$$

  - Sum of probabilities of all wrong answers
- The above quantity (‡) is minimized for this particular x ∈ X when

  $$f(x) = \arg\max_{y \in \mathcal{Y}} \; P(Y = y|X = x)$$

- The classifier f with this property for all x ∈ X is called the **Bayes classifier**, and it has the smallest prediction error (†) among all classifiers.

- The Bayes Classifier
  - Under the assumption (X, Y) iid ~ *P*, the optimal classifier is:

    $$f^{\star}(x) := \arg\max_{y \in \mathcal{Y}} \; P(Y = y|X = x)$$

  - From Bayes rule we equivalently have

    $$f^{\star}(x) = \arg\max_{y \in \mathcal{Y}} \; \underbrace{P(Y = y)}_{\textit{class prior}} \times \underbrace{P(X = x|Y = y)}_{\textit{data likelihood}\,|\,\textit{class}}$$

- Generative model: model x and y with distributions
- Discriminative model: plug x into distribution on y
- Example: Gaussian Class Conditional Densities

  Suppose $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \{0, 1\}$, and the distribution $\mathcal{P}$ of (X, Y) is as follows.

  ▸ **Class prior**: $P(Y = y) = \pi_y, \quad y \in \{0, 1\}$.

  ▸ **Class conditional density** for class $y \in \{0, 1\}$: $p_y(x) = N(x|\mu_y, \sigma_y^2)$.

  ▸ **Bayes classifier**:

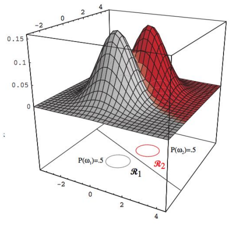
  $$\begin{aligned} f^{\star}(x) &= \operatorname*{argmax}_{y \in \{0,1\}} \; p(X = x|Y = y)P(Y = y) \\ &= \begin{cases} 1 & \text{if } \dfrac{\pi_1}{\sigma_1} \exp\left[-\dfrac{(x - \mu_1)^2}{2\sigma_1^2}\right] > \dfrac{\pi_0}{\sigma_0} \exp\left[-\dfrac{(x - \mu_0)^2}{2\sigma_0^2}\right] \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

- Given two sets with shared variance, decision boundary will be linear
- Given two sets with unique variance, decision boundary will be quadratic
- In general, Bayes classifier may be rather complicated
- Plug-In Classifier:
  - Approximate P(Y=y) and P(X=x|Y=y)
  - The result may no longer give the best results among all the classifiers we can choose from
  - Example:

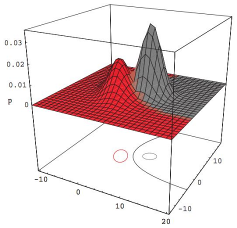
    Class priors: The MLE estimate of π_y is $\hat{\pi}_y = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y_i = y)$

    Class conditional density: Choose $p(x|Y = y) = N(x|\mu_y, \Sigma_y)$.

  - The MLE estimate of (μ_y, Σ_y) is



$\Sigma_0 = \Sigma_1$
Bayes classifier:
linear separator

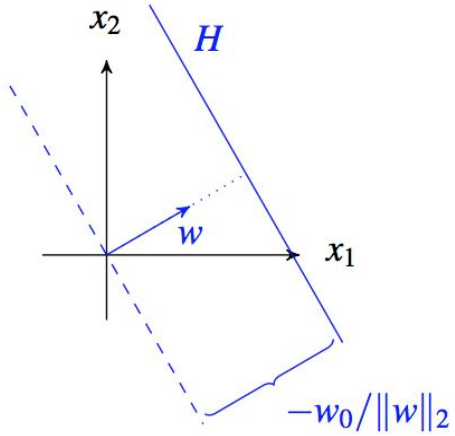


$\Sigma_0 \neq \Sigma_1$
Bayes classifier:
quadratic separator

    - $$\hat{\mu}_y = \frac{1}{n_y}\sum_{i=1}^{n} \mathbb{1}(y_i = y)x_i,$$ $$\hat{\Sigma}_y = \frac{1}{n_y}\sum_{i=1}^{n} \mathbb{1}(y_i = y)(x_i - \hat{\mu}_y)(x_i - \hat{\mu}_y)^T.$$
    - This is just the empirical mean and covariance of class y
    - **Plug-in classifier:**
    - $$\hat{f}(x) = \arg\max_{y \in \mathcal{Y}} \ \hat{\pi}_y |\hat{\Sigma}_y|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x - \hat{\mu}_y)^T \hat{\Sigma}_y^{-1}(x - \hat{\mu}_y)\right\}$$
    - Naive Bayes is a Bayes classifier that makes the assumption:
        - $$p(X = x|Y = y) = \prod_{j=1}^{d} p_j(x(j)|Y = y)$$ (treats X as conditionally independent given y => “Naive”)

Lecture 8 - Linear Classifiers and Perceptron

- Binary linear classifier: f(x) = sign(x^Tw + w_o)
    - Assume linear separability
    - w, w_o define affine hyperplane
- Hyperplane in R^d is linear subspace of dimension (d-1)
    - Vector representation: H = {x ∈ R^d | x^Tw = 0}
    - H = x^Tw + w_0 = 0
        - Setting w_0 > 0 moves H in opposite direction of w (magnitude -w_0 / $||w||_2$)



- Perceptron
    - y = f(x) = sign(x^Tw)
    - Seeks to minimize:
        - $$\mathcal{L} = -\sum_{i=1}^{n} (y_i \cdot x_i^T w)\mathbb{1}\{y_i \neq \text{sign}(x_i^T w)\}$$
            - Adds a negative number every time something is mislabeled
    - Because y ∈ {−1, +1}, $y_i \cdot x^T_i w$ is:
        - > 0 if y_i = sign($x^T_i w$)
        - < 0 if y_i != sign($x^T_i w$)
        - By minimizing L we’re trying to always predict the correct label.
    - Gradient Descent
        - Pick mislabeled example, update $w^{(t+1)} = w^{(t)} + \eta y_i x_i$
    - Drawbacks:
        - Only finds “some” hyperplane (not necessarily best/most generalizable)
        - Requires linear separability

Lecture 9 - Logistic Regression and LaPlace Approximation

- Logistic regression => finds linear boundary in R^d
    - $$P(y_i = +1|x_i, w) = \sigma(x_i^T w), \quad \sigma(x_i; w) = \frac{e^{x_i^T w}}{1 + e^{x_i^T w}}$$
    - Does not model X, thus is a discriminative model
        - Maximizes likelihood of of y => low bias, high variance
    - Joint Likelihood:

        - $$\begin{aligned} p(y_1,\ldots,y_n|x_1,\ldots,x_n,w) &= \prod_{i=1}^{n} p(y_i|x_i,w) \\ &= \prod_{i=1}^{n} \sigma_i(w)^{\mathbb{1}(y_i=+1)} \left(1-\sigma_i(w)\right)^{\mathbb{1}(y_i=-1)} \end{aligned}$$
        - $$p(y_1,\ldots,y_n|x_1,\ldots,x_n,w) = \prod_{i=1}^{n} \sigma_i(y_i \cdot w)$$
            - Want to maximize this over w
    - The maximum likelihood solution for w can be written:
        - $$\begin{aligned} w_{\text{ML}} &= \arg\max_{w} \sum_{i=1}^{n} \ln \sigma_i(y_i \cdot w) \\ &= \arg\max_{w} \mathcal{L} \end{aligned}$$ <= can't solve analytically
    - Steepest Ascent Algo: for step t = 1, 2, . . . do
        - $$\text{Update } w^{(t+1)} = w^{(t)} + \eta \sum_{i=1}^{n} \left(1-\sigma_i(y_i \cdot w)\right) y_i x_i$$
        - Modify plane using examples we got very wrong
- Bayesian Logistic Regression
    - $$w_{\text{MAP}} = \arg\max_w \textstyle\sum_{i=1}^{n} \ln \sigma_i(y_i \cdot w) - \lambda w^T w$$
    - Posterior Calculation:
        - Prior: w ~ N(0, λ^{-1}I)
        - Posterior:
            - $$p(w|x,y) = \frac{p(w)\prod_{i=1}^{n}\sigma_i(y_i \cdot w)}{\int p(w)\prod_{i=1}^{n}\sigma_i(y_i \cdot w)\,dw}$$
        - Can't calculate this, want to approximate
            - Strategy: Set p(w|x, y) ≈ Normal(μ, Σ), use LaPlace approximation to set μ and Σ
            - We can approximate f(w) = ln p(y,w|X) with a second order Taylor expansion
                - $$f(w) \approx f(z) + (w-z)^T \nabla f(z) + \frac{1}{2}(w-z)^T \left(\nabla^2 f(z)\right)(w-z)$$
            - The Laplace approximation defines z = w_{MAP}
    - Laplace approximation for logistic regression => Given labeled data (x_1, y_1),...,(x_n, y_n) and:
        - $$P(y_i|x_i,w) = \sigma(y_i x_i^T w), \quad w \sim N(0,\lambda^{-1}I), \qquad \sigma(y_i x_i^T w) = \frac{e^{y_i x_i^T w}}{1+e^{y_i x_i^T w}}$$, then:
        - 1. Find: $w_{\text{MAP}} = \arg\max_{w} \sum_{i=1}^{n} \ln \sigma(y_i x_i^T w_{\text{MAP}}) - \frac{\lambda}{2} w^T w$

          2. Set: $-\Sigma^{-1} = -\lambda I - \sum_{i=1}^{n} \sigma(y_i x_i^T w_{\text{MAP}})\left(1-\sigma(y_i x_i^T w_{\text{MAP}})\right) x_i x_i^T$

          3. Approximate: $p(w|x,y) = N(w_{\text{MAP}}, \Sigma)$.

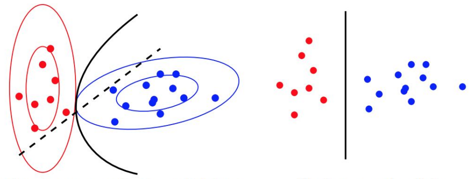
Lecture 10 - Kernel Methods and Gaussian Processes

- Feature expansion => map features to higher dimensional space
- Kernel => useful when we only have to work with dot products

    - A kernel K(·, ·) : $R^d \times R^d \rightarrow R$ is a symmetric function defined as follows:
        - For any set of n data points $x_1, \ldots, x_n \in R^d$, the n × n matrix K, where $K_{ij} = K(x_i, x_j)$, is positive semidefinite. Intuitively, this means K satisfies the properties of a covariance matrix.
    - Mercer's theorem: If the function K(·, ·) satisfies the above properties, then there exists a mapping φ : $R^d \rightarrow R^D$ such that:
        - $$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$
- Gaussian Kernel / Radial Basis Function (RBF) - most popular kernel
    - $$K(x, x') = a \exp\left\{-\frac{1}{b}\|x - x'\|^2\right\}$$
    - Takes into account proximity in R^d. x, x' close together have larger value (as defined by kernel width b) because *e* is raised to a less negative exponent.
    - The mapping φ(x) that produces the RBF kernel is infinite dimensional (it's a continuous function instead of a vector). Therefore:
        - $$K(x, x') = \int \phi_t(x)\phi_t(x')\, dt$$
- Certain functions of kernels can produce new kernels
- Kernelized Perceptron
    - $$\begin{aligned} y_0 &= \text{sign}\left(\sum_{i \in \mathcal{M}} y_i \phi(x_0)^T \phi(x_i)\right) \\ &= \text{sign}\left(\sum_{i \in \mathcal{M}} y_i K(x_0, x_i)\right) \end{aligned}$$
    - E.g. with RBF:
        - $$y_0 = \text{sign}\left(\sum_{i \in \mathcal{M}} y_i\, e^{-\frac{1}{b}\|x_0 - x_i\|^2}\right)$$
            - Notice $0 < K(x_0, x_i) \leq 1$, with bigger values when x_0 is closer to x_i. This is like a "soft voting" among the data picked by Perceptron.
- Kernel K-NN => like perceptron except sum over all data (not just misclassified)
- Kernel Regression => take a locally weighted average of all y_i for which x_i is close to x_0 (as decided by the kernel width)
- Gaussian processes
    - Let $f(x) \in R$ and $x \in R^d$. Define the kernel $K(x, x_0)$ between two points x and x_0.
    - Then f(x) is a Gaussian process and y(x) the noise-added process if:
        - $$y\,|f \sim N(f, \sigma^2 I), \quad f \sim N(0, K) \quad \Leftrightarrow \quad y \sim N(0, \sigma^2 I + K)$$
          where $y = (y_1, \ldots, y_n)^T$ and K is $n \times n$ with $K_{ij} = K(x_i, x_j)$.
    - Predictions: Given measured data $D_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, the distribution of y(x) can be calculated at any new x to make predictions
        - $$y(x)|\mathcal{D}_n \sim N(\mu(x), \Sigma(x))$$
        - Uses measure of the similarity between points (kernel function) to predict the value for an unseen point from training data. The prediction is not just an estimate for that point, but also has uncertainty information—is itself a Gaussian distribution

Lecture 11 - Maximum Margin Classifiers and the SVM

- Convex hull => defined by all possible weighted averages of points in set
- Margin => shortest distance between hyperplane H and any point in the convex hull of either set



(dotted) poor generalization, (solid) better

Maximum margin solution

    - With no distribution assumptions, we can argue that max-margin is best
- Support Vector Machines
    - For n linearly separable points $(x_1, y_1), \dots, (x_n, y_n)$ with $y_i \in \{\pm 1\}$, solve:
        - $$\begin{aligned} \min_{w,w_0} \quad & \tfrac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(x_i^T w + w_0) \geq 1 \quad \text{for } i = 1, \dots, n \end{aligned}$$
        - This is the primal optimization problem
    - Intuitively, solution finds shortest line connecting the convex hulls and places hyperplane orthogonal to line and exactly at the midpoint.
    - Minimize:
        - $$\Big\| \underbrace{\left(\textstyle\sum_{x_i \in S_1} \alpha_{1i} x_i\right)}_{\text{in conv. hull of } S_1} - \underbrace{\left(\textstyle\sum_{x_i \in S_0} \alpha_{0i} x_i\right)}_{\text{in conv. hull of } S_0} \Big\|_2$$
    - Define Lagrange multipliers $\alpha_i > 0$ for i = 1, . . . , n. The Lagrangian is
        - $$\begin{aligned} \mathcal{L} &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i (x_i^T w + w_0) - 1) \\ &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y_i (x_i^T w + w_0) + \sum_{i=1}^{n} \alpha_i \end{aligned}$$
        - We want to minimize L over w and w_0 and maximize over $(\alpha_1, \dots, \alpha_n)$
        - The Lagrandian enables constrained optimization (shortest vector, max margin)
    - Minimize over w and w_0:
        - $$\begin{aligned} \nabla_w \mathcal{L} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \quad &\Rightarrow \quad w = \sum_{i=1}^{n} \alpha_i y_i x_i \\ \frac{\partial \mathcal{L}}{\partial w_0} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \quad &\Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$
    - Dual problem (notice no w):
        - $$\begin{aligned} \max_{\alpha_1, \dots, \alpha_n} \quad & \mathcal{L} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T x_j) \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \qquad \alpha_i \geq 0 \quad \text{for } i = 1, \dots, n \end{aligned}$$
    - Solve for w:
        - $$\nabla_w \mathcal{L} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad \text{(just plug in the learned } \alpha_i\text{'s)}$$
        - Direction of w is u-v; $u \in H(S_1)$, $v \in H(S_0)$
- Soft-Margin SVM
    - Objective function:
        - $$\begin{aligned} \min_{w, w_0, \xi_1, \dots, \xi_n} \quad & \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(x_i^T w + w_0) \geq 1 - \xi_i \quad \text{for} \quad i = 1, \dots, n \\ & \xi_i \geq 0 \quad \text{for} \quad i = 1, \dots, n \end{aligned}$$
        - ξ_i => slack variables
    - Role of λ:
        - λ => 0 : happy to misclassify
        - λ => ∞, we recover the original SVM because we want $\xi_i = 0$
- Kernelizing SVM:

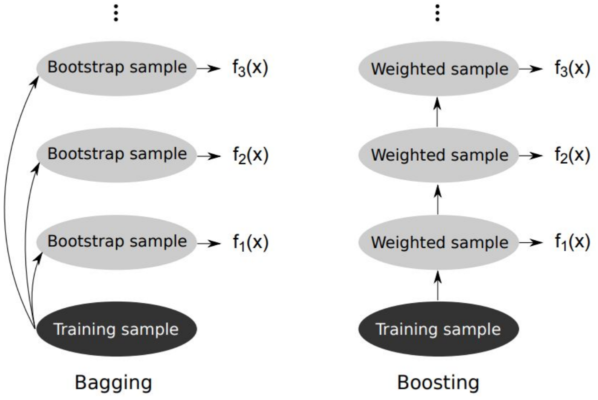
    - $y_i(\varphi(x_i)^T w + w_0) \geq 1 - \xi_i$
        - Perhaps maps x_i to higher dimension
- Basic SVM: linear classifier for linearly separable data
- Full fledged SVM:
    - Max margin - good generalization
    - Slack variables - overlapping classes, robust against outliers
    - Kernel - non-linear decision boundary in R^d

Lecture 12 - Decision Trees

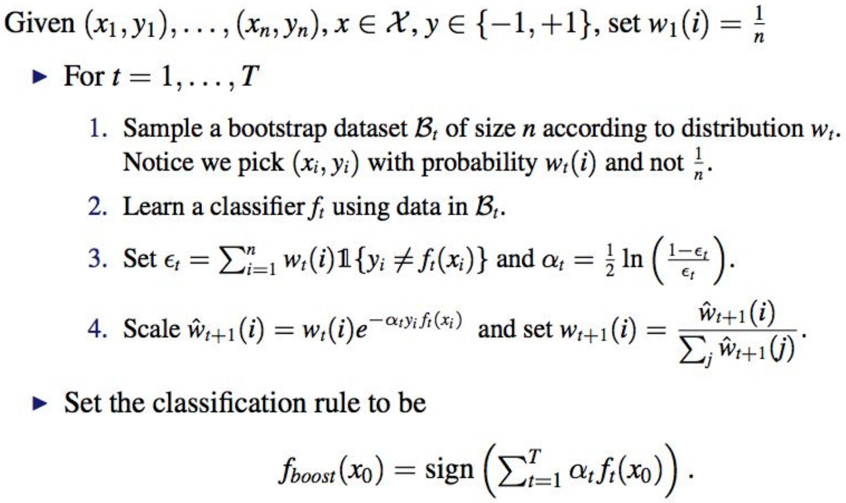
- Decision tree maps input $x \in R^d$ to output y using binary decision rules
    - Splitting rules look like $h(x) = 1\{x_j > t\}$
    - Each non-leaf node has a splitting rule and 2 children, leaf nodes give output value
- Learning Trees: split leaf that reduces uncertainty the most
    - Top-down greedy algorithm.
        - Start with a single leaf node containing all data, loop through the following steps:
            - Pick the leaf to split that reduces uncertainty the most.
            - Figure out the $\lesseqgtr$ decision rule on one of the dimensions.
            - Stopping rule is chosen.
        - Label/response of the leaf is majority-vote/average of data assigned to it
- Regression Trees: Partition the space so that data in a region have same prediction
- Classification Trees:
    - For all $x \in R_m$, let p_k be empirical fraction labeled k
    - Measures of quality of R_m include:
        - 1. Classification error: $1 - \max_k p_k$

          2. Gini index: $1 - \sum_k p_k^2$

          3. Entropy: $-\sum_k p_k \ln p_k$
        - These are all maximized when p_k is uniform on the K classes in R_m
        - These are minimized when $p_k = 1$ for some k (R_m only contains one class)
    - Training error goes to zero as size of tree increases
    - Testing error decreases but then increases because of overfit.
- Ensemble Methods
    - Bootstrap => technique for improving estimators (i.e. resampling)
        - For rule rule Sˆ of a statistic S, generate bootstrap samples $B_1, \ldots, B_B$
        - Evaluate the estimator Sˆ on each B_b
        - Estimate mean and variance of Sˆ
            - $$\mu_B = \frac{1}{B}\sum_{b=1}^{B}\hat{S}_b, \quad \sigma_B^2 = \frac{1}{B}\sum_{b=1}^{B}(\hat{S}_b - \mu_B)^2$$
    - Bagging (Bootstrap Aggregation) => train classification/regression model f_b on B_b
        - For a new point predict:
            - $$f_{\text{avg}}(x_0) = \frac{1}{B}\sum_{b=1}^{B} f_b(x_0)$$
        - High bias, low variance (however bagged trees are correlated)
    - Random Forests => only consider random subset of dimensions of $x \in R^d$ (reduce correlation)



Lecture 13 - Boosting

- Boosting
    - $f_{boost}(x_0)$ = sign $\Sigma^T_{t=1}$ $\alpha_t f_t(x_0)$
    - Combine many weak classifiers
    - Weight samples to improve f_t on every iteration
    - Boosting a bad classifier is often better than not boosting a good one
    - ξ_t is weighted error of f_t
- Adaboost Algorithm
    - 

Given $(x_1, y_1), \dots, (x_n, y_n), x \in \mathcal{X}, y \in \{-1, +1\}$, set $w_1(i) = \frac{1}{n}$

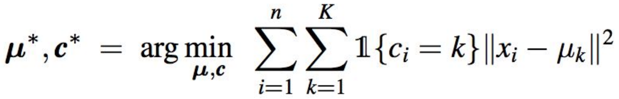
▸ For $t = 1, \dots, T$

1. Sample a bootstrap dataset $\mathcal{B}_t$ of size n according to distribution w_t. Notice we pick (x_i, y_i) with probability $w_t(i)$ and not $\frac{1}{n}$.
2. Learn a classifier f_t using data in $\mathcal{B}_t$.
3. Set $\epsilon_t = \sum_{i=1}^n w_t(i) \mathbb{1}\{y_i \neq f_t(x_i)\}$ and $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
4. Scale $\hat{w}_{t+1}(i) = w_t(i) e^{-\alpha_t y_i f_t(x_i)}$ and set $w_{t+1}(i) = \frac{\hat{w}_{t+1}(i)}{\sum_j \hat{w}_{t+1}(j)}$.

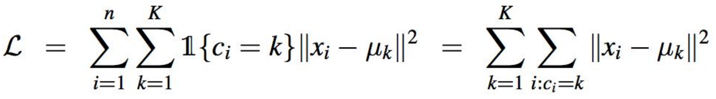
▸ Set the classification rule to be

$$f_{boost}(x_0) = \text{sign}\left(\sum_{t=1}^T \alpha_t f_t(x_0)\right).$$

Lecture 14 - Clustering and K-Means

- Supervised learning: p(x|y) p(y)
- Unsupervised learning: p(y|x) p(x) <= focus on p(x)
- Clustering: given $x_1,...,x_n$ group data into K clusters
- K-Means
    - outputs:
        - vector c of cluster assignemnts, $c_i \in \{1,...,K\}$
        - K mean vectors μ_k; $\mu = \{\mu_1,...,\mu_K\}$
    - The K-means objective function can be written as
        - 

$$\boldsymbol{\mu}^*, \boldsymbol{c}^* = \arg\min_{\boldsymbol{\mu}, \boldsymbol{c}} \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}\{c_i = k\} \|x_i - \mu_k\|^2$$

-
    -
        - Uses the squared Euclidean distance of x_i to the centroid μ_k (only penalizes the distance of x_i to assigned centroid c_i)
        - 

$$\mathcal{L} = \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}\{c_i = k\} \|x_i - \mu_k\|^2 = \sum_{k=1}^K \sum_{i:c_i=k} \|x_i - \mu_k\|^2$$

-
    -
        -
            - “Non-convex” => we can’t actually find the optimal μ* and c*
            - We can only derive an algorithm for finding a local optimum
    - Coordinate descent:
        - Fix μ, find optimal c
        - Fix c, find optimal μ
            - This sort of dependent optimization paradigm is common in unsupervised models
            - Different initializations can give different results
    - Algorithm:
        - Assignment Step:

      - $$c_i = \arg\min_k \quad \|x_i - \mu_k\|^2$$
      - c_i is centroid closest to x_i
    - Update Step:
      - $$\mu_k = \frac{1}{n_k}\sum_{i=1}^{n} x_i \mathbb{1}\{c_i = k\}$$
      - Mean of data asssigned to cluster k
  - Every update to c_i or μ_k decreases L compared to the previous value. Therefore, L is monotonically decreasing.
  - Selecting K => if K = n then L = 0
  - K-Medoids: e.g. D(x,μ) = $||x-\mu||_1$
    - More robust to outliers

Lecture 15 - The EM Algorithm for Maximum Likelihood, Missing Data

- Probabilistic models:
  - Bayes classifier, logistic regression, least squares and ridge regression (ML and MAP interpretations)
- Non-probabilistic models:
  - Perceptron, SVM, Decisions Trees, K-Means
- Maximum Likelihood
  - Maximum likelihood seeks to find the θ that maximizes the likelihood
    - $$\theta_{\text{ML}} = \arg\max_{\theta}\ p(x_1,\ldots,x_n|\theta) \overset{(a)}{=} \arg\max_{\theta} \prod_{i=1}^{n} p(x_i|\theta) \overset{(b)}{=} \arg\max_{\theta} \sum_{i=1}^{n} \ln p(x_i|\theta)$$
    - Can't always solve analytically
- Coordinate Ascent => split parameters into groups, solve for one given the other
  - $$\theta_{1,\text{ML}}, \theta_{2,\text{ML}} \;=\; \arg\max_{\theta_1,\theta_2} \;\sum_{i=1}^{n} \ln p(x_i|\theta_1,\theta_2)$$
- Expectation Maximization:
  - Motivating example: given x_i with missing data, assign expectation of missing data and then maximize over parameters θ
  - Objective function:
    - $$\ln p(x|\theta_1) = \underbrace{\int q(\theta_2)\ln\frac{p(x,\theta_2|\theta_1)}{q(\theta_2)}\,d\theta_2}_{\text{A function only of } \theta_1\text{, we'll call it } \mathcal{L}} + \underbrace{\int q(\theta_2)\ln\frac{q(\theta_2)}{p(\theta_2|x,\theta_1)}\,d\theta_2}_{\text{Kullback-Leibler divergence}}$$
    - The KL divergence is always ≥ 0 and only = 0 when q = p.
    - We are assuming that the integral in L can be calculated, leaving a function only of θ_1 (for a particular setting of the distribution q)
  - The EM Algorithm
    - **Given** the value $\theta_1^{(t)}$, **find** the value $\theta_1^{(t+1)}$ as follows:

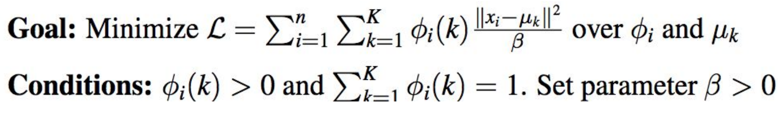
      E-step: Set $q_t(\theta_2) = p(\theta_2|x,\theta_1^{(t)})$ and calculate

      $$\mathcal{L}_t(x,\theta_1) = \int q_t(\theta_2)\ln p(x,\theta_2|\theta_1)\,d\theta_2 \;-\; \underbrace{\int q_t(\theta_2)\ln q_t(\theta_2)\,d\theta_2}_{\text{can ignore this term}}$$

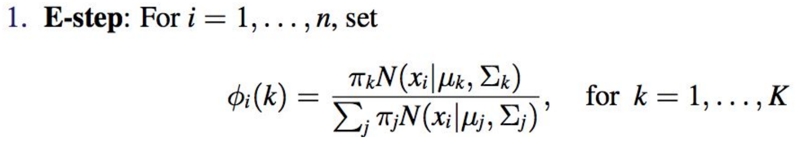
      M-step: Set $\theta_1^{(t+1)} = \arg\max_{\theta_1}\ \mathcal{L}_t(x,\theta_1)$.

        - E-Step assumes log likelihood and conditional posterior distribution of auxiliary variable are in closed form
        - Sequence $\theta_1^{(t)}$ monotonically improves
    - Generally => coordinate ascent instance
        - E-Step: Set expected value of θ_2 given θ_1
        - M-Step: Set θ_1 that maximizes likelihood of data given θ_2
    - For EM with missing data, the output is μ_{ML}, Σ_{ML} and $q(x_i^m)$ for all missing entries

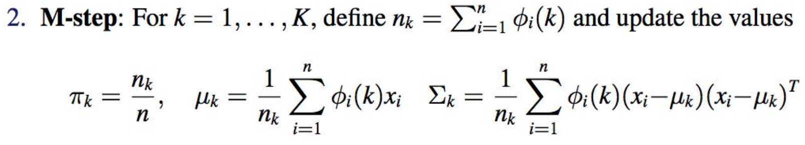
Lecture 16 - Mixture Models, Gaussian Mixture Models

- Hard clustering => assign observation to one group
    - e.g. K-means
- Soft clustering => break data across clusters
    - e.g. weighted K-means
- Weighted K-Means
    - 

      Goal: Minimize $\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} \phi_i(k) \frac{\|x_i - \mu_k\|^2}{\beta}$ over ϕ_i and μ_k

      Conditions: $\phi_i(k) > 0$ and $\sum_{k=1}^{K} \phi_i(k) = 1$. Set parameter $\beta > 0$
    - Update φ: For each i, update the word allocation weights
    - Update μ: For each k, update μ_k with the weighted average
    - Weight vector φ_i is like a probability of x_i being each cluster
- Mixture Model: φ_i actually is probability distribution from model
    - Works by defining:
        - Prior on the cluster assignment c_i
        - Likelihood on x_i given c_i
    - Generative model; weighted combination of simpler distributions in same family
    - Process: For observation number i = 1, . . . , n
        - Generate cluster assignment: c_i iid ~ Discrete(π) ⇒ Prob(c_i = k|π) = π_k
        - Generate observation: $x_i \sim p(x|\theta_{ci})$
        - c_i picks out the index of θ most probably used to generate x_i
- Gaussian Mixture Models
    - Params: Let π be a K-dimensional probability distribution and (μ_k, Σ_k) be the mean and covariance of the kth Gaussian in R^d
    - Generate data:
        - 1. Assign the ith observation to a cluster, c_i ~ Discrete(π)
        - 2. Generate the value of the observation, $x_i \sim N(\mu_{ci}, \Sigma_{ci})$
    - Objective: Maximize the likelihood over model parameters π, μ and Σ
        - Treat c_i as auxillary data and use Expectation Maximization
            - EM is helpful because it uses sum of logs instead of log of sum
- Maximum Likelihood for Gaussian Mixture Model with EM
    - 

      1. **E-step**: For $i = 1, \dots, n$, set

      $$\phi_i(k) = \frac{\pi_k N(x_i|\mu_k, \Sigma_k)}{\sum_j \pi_j N(x_i|\mu_j, \Sigma_j)}, \quad \text{for } k = 1, \dots, K$$
    - 

      2. **M-step**: For $k = 1, \dots, K$, define $n_k = \sum_{i=1}^{n} \phi_i(k)$ and update the values

      $$\pi_k = \frac{n_k}{n}, \quad \mu_k = \frac{1}{n_k} \sum_{i=1}^{n} \phi_i(k) x_i \quad \Sigma_k = \frac{1}{n_k} \sum_{i=1}^{n} \phi_i(k)(x_i - \mu_k)(x_i - \mu_k)^T$$
    - E-Step: Find expected weight of each observations towards each cluster
    - M-Step: Set μ, Σ, π to maximize likelihood of data

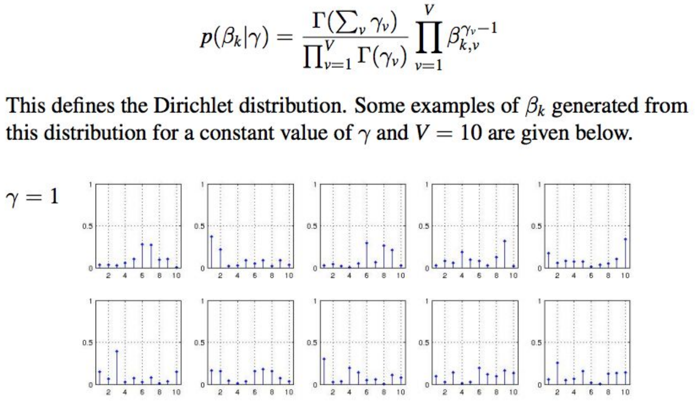
- GMM vs. Bayes Classifier
    - The GMM feels a lot like a K-class Bayes classifier, where the label of x_i is:
        - label(x_i) = $\text{argmax}_k\ \pi_k\ N(x_i \mid \mu_k, \Sigma_k)$
    - GMM doesn't know label, can't optimize π, μ and Σ directly
    - With Bayes, φi encodes a label and is known

Lecture 17 - Matrix Factorization, Collaborative Filtering for Recommendation

- Content filtering => use known info about products and users
- Collaborative filtering => use previous users' input/behavior
- Matrix factorization
    - M will have many missing values, use matrix factorization to predict missing M_{ij}
    - Model for low-rank factorization of M:
        - Account for the fact most values are missing
        - Be low-rank, where d << min{N_1, N_2}
        - Learn $u_i \in R^d$ for user i and $v_j \in R^d$ for object j
    - Models correlations
- Probabilistic Matrix Factorization (PMF)
    - For N_1 users and N_2 objects, generate
        - User locations: $u_i \sim N(0, \lambda^{-1}I)$, i = 1, . . . , N_1
        - Object locations: $v_j \sim N(0, \lambda^{-1}I)$, j = 1, . . . , N_2
    - Given these locations the distribution on the data is
        - $M_{ij} \sim N(u^T_i v_j, \sigma^2)$, for each (i, j) ∈ Ω
    - The MAP solution for U and V is the maximum of the log joint likelihood
        - $$U_{\text{MAP}}, V_{\text{MAP}} = \arg\max_{U,V} \sum_{(i,j)\in\Omega} \ln p(M_{ij}|u_i, v_j) + \sum_{i=1}^{N_1} \ln p(u_i) + \sum_{j=1}^{N_2} \ln p(v_j)$$
    - Calling the MAP objective function L, we want to maximize
        - $$\mathcal{L} = -\sum_{(i,j)\in\Omega} \frac{1}{2\sigma^2}\|M_{ij} - u_i^T v_j\|^2 - \sum_{i=1}^{N_1} \frac{\lambda}{2}\|u_i\|^2 - \sum_{j=1}^{N_2} \frac{\lambda}{2}\|v_j\|^2 + \text{constant}$$
    - MAP Inference Coordinate Ascent Algorithm:
        - For i = 1, . . . , N_1 => update user location
        - For j = 1, . . . , N_2 => update object location
- Matrix factorization and Ridge regression
    - Matrix factorization looks like RR from the perspective of v_j, minimize squared error with penalty $\lambda||v_j||^2$
    - So this model is a set of N1 + N2 coupled ridge regression problems
    - Least squares solution requires every user and object has at least d ratings

Lecture 18 - Topic Modeling, Non-negative Matrix Factorization

- Topic modeling => Learn distribution on topics shared by documents
    - Learn distribution on topics for each document
    - Assign every word in document to topic
- Latent Dirichlet Allocation
    - β : A collection of distributions on words (topics).
    - θ : A distribution on topics for each document
    - The generative process for LDA is:

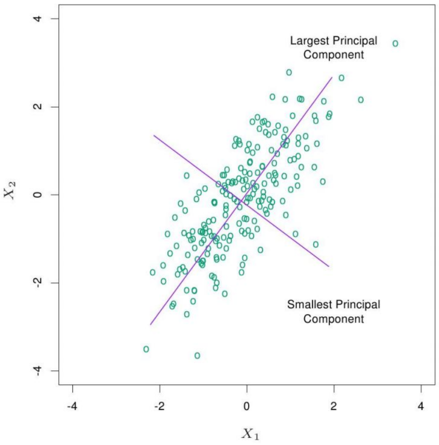


$$p(\beta_k|\gamma) = \frac{\Gamma(\sum_v \gamma_v)}{\prod_{v=1}^V \Gamma(\gamma_v)} \prod_{v=1}^V \beta_{k,v}^{\gamma_v - 1}$$

This defines the Dirichlet distribution. Some examples of β_k generated from this distribution for a constant value of γ and $V = 10$ are given below.

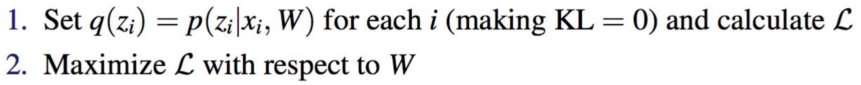
        - Generate each topic, which is a distribution on words
            - β_k ~ Dirichlet(γ), k = 1, . . . , K
        - For each document, generate a distribution on topics
            - θ_d ~ Dirichlet(α), d = 1, . . . , D
    - Bag of words model
    - As γ gets smaller, puts p on less topics
    - For a particular document, what is $P(x_{dn} = i|\beta, \theta_d)$?
        - LDA can be thought of as an instance of nonnegative matrix factorization: we can read the probabilities from a matrix formed by taking the product of two matrices that have nonnegative entries (B and Θ)
- Nonnegative Matrix Factorization => parts-based model
    - X has nonnegative entries, none missing but likely many zero's
    - Learned factorization is also nonnegative
    - $X_{ij} \approx \Sigma_k W_{ik}H_{kj}$
    - NMF minimizes one of the following two objective functions over W and H
    - Squared error objective
        - $||X - WH||^2 = \Sigma_i\Sigma_j (X_{ij} - (WH)_{ij})^2$
    - Divergence objective
        - $D(X || WH) = - \Sigma_i\Sigma_j [X_{ij} \ln(WH)_{ij} - (WH)_{ij}]$ <= Poisson Model
- NMF + Topic Modeling => divergence penalty related to LDA
    - LDA is Bayesian (MAP) => p(w|x)
    - NMF is Maximum Likelihood => p(x|w)
- NMF + Face Modeling => NMF learns parts-based representation

Lecture 19 - Principal Component Analysis



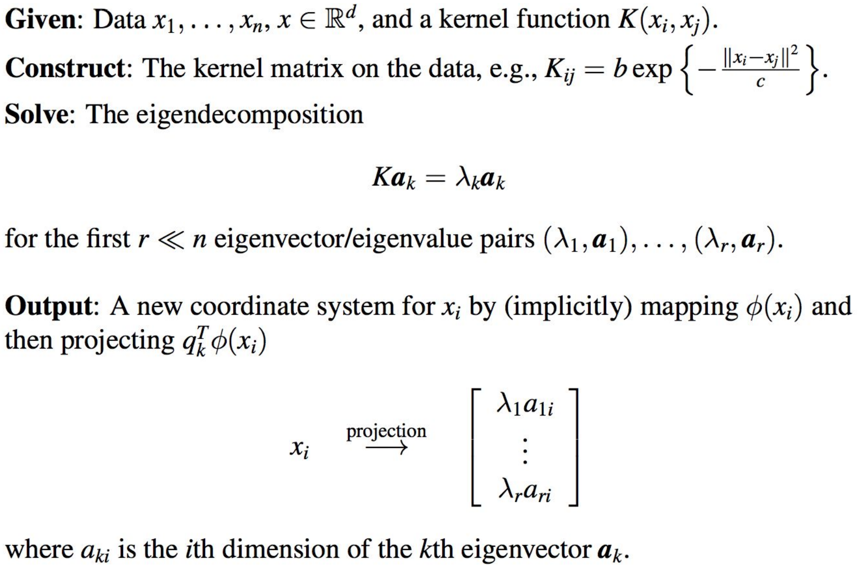
- Principal Component Analysis => maps x_i to low-d space defined by "principal components" while conserving important data
    - For example: define $X = [x_1, . . . , x_n]$
        - $$q = \arg\max_q \; q^T(XX^T)q \qquad \text{subject to} \quad q^Tq = 1$$
        - q is the first eignvector of XX^T
        - $\lambda = q^T(XX^T)q$ is first eigenvalue
    - Intuitively, first principal component captures as much of the variance as possible, second captures second most, etc. Can often explain most variance with a few dimensions.
- General PCA
    - Consider K eigenvectors
    - $$q \quad = \quad \arg\min_q \sum_{i=1}^{n} \|x_i - \underbrace{\sum_{k=1}^{K}(x_i^T q_k)q_k}_{\text{approximates } x} \|^2 \qquad \text{s.t. } q_k^T q_{k'} = \begin{cases} 1, & k = k' \\ 0, & k \neq k' \end{cases}$$
    - $Q = [q_1,...,q_K]$ give us K dimensionl subspace to represent data
        - $$x_{\text{proj}} = \begin{bmatrix} q_1^T x \\ \vdots \\ q_K^T x \end{bmatrix}, \qquad x \;\approx\; \sum_{k=1}^{K}(q_k^T x)q_k = Qx_{\text{proj}}$$
    - An equivalent formulation of the problem is to find (λ, q) such that:
        - $(XX^T)q = \lambda q$
- Probababilistic PCA

- With SVD: $X = USV^T$, $U^TU = I$, $V^TV = I$
- Approximate $X \approx WZ$, where
  - W is a d × K matrix. In different settings this is called a “factor loadings” matrix, or a “dictionary.” It’s like the eigenvectors, but no orthonormality
  - The ith column of Z is called $z^i \in R^K$. Think of it as a low-dimensional representation of x_i
- Learn with EM algo, goal is to find w_{ML}
  - $x_i \sim N(W_{zi}, \sigma^2)$, $z_i \sim N(0, I)$



1. Set $q(z_i) = p(z_i|x_i, W)$ for each i (making $\text{KL} = 0$) and calculate $\mathcal{L}$
2. Maximize $\mathcal{L}$ with respect to W

- Kernel PCA
  - Let φ(x) be a feature mapping from R^d to R^D, where D >> d



Given: Data $x_1, \dots, x_n$, $x \in \mathbb{R}^d$, and a kernel function $K(x_i, x_j)$.

Construct: The kernel matrix on the data, e.g., $K_{ij} = b \exp\left\{-\frac{\|x_i - x_j\|^2}{c}\right\}$.

Solve: The eigendecomposition

$$K\boldsymbol{a}_k = \lambda_k \boldsymbol{a}_k$$

for the first $r \ll n$ eigenvector/eigenvalue pairs $(\lambda_1, \boldsymbol{a}_1), \dots, (\lambda_r, \boldsymbol{a}_r)$.

Output: A new coordinate system for x_i by (implicitly) mapping $\phi(x_i)$ and then projecting $q_k^T \phi(x_i)$

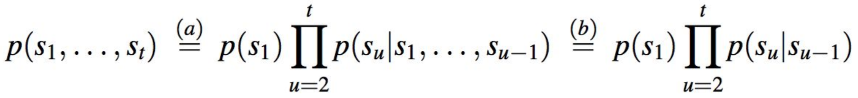
$$x_i \xrightarrow{\text{projection}} \begin{bmatrix} \lambda_1 a_{1i} \\ \vdots \\ \lambda_r a_{ri} \end{bmatrix}$$

where a_{ki} is the ith dimension of the kth eigenvector $\boldsymbol{a}_k$.

- Related to spectral clustering, manifold learning

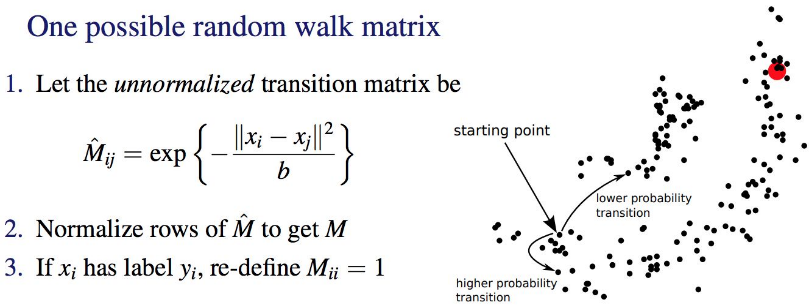
Lecture 20 - Markov Models, Ranking and Semi-supervised Classification Examples

- Sometimes i.i.d. Assumption is clearly wrong, e.g. with sequential data
- Markov Chain
  - First order: next location depends only on previous 1 (i.e. current)
  - Second order: next location depends only on previous 2
  - Let s ∈ {1, . . . , S}. A sequence (s1, . . . ,st) is a first-order Markov chain if



$$p(s_1, \dots, s_t) \overset{(a)}{=} p(s_1) \prod_{u=2}^{t} p(s_u|s_1, \dots, s_{u-1}) \overset{(b)}{=} p(s_1) \prod_{u=2}^{t} p(s_u|s_{u-1})$$

  - Transition Matrix - M_{ij} is the probability that next position is j given current position i
    - $M_{ij} = p(s_t = j \mid s_{t-1} = i)$, M is SxS and rows sum to 1
  - Maximum Likelihood: $M_{ML} = \text{argmax}_M\ p(s_1, \dots, s_t \mid M)$
    - Form by counting, empirically
  - State Distribution
    - $w_{t+1} = w_tM$
  - Stationary Distribution
    - w_∞ is the same vector for all w_0 if:
      - We can reach any state from any starting state
      - Sequence doesn’t loop in predefined pattern
    - $w_\infty = w_\infty M$ since w_t is converging and $w_{t+1} = w_tM$
  - Markov Chains and Ranking

        - Data is pairwise comparisons between objects
- Semi-supervised learning
    - Data with few labels given, most y_i missing
    - Want to capture structure in X
- Random Walk Classifier => starting from x_i:
    - Move point to point, transition to nearby point more likely
    - Transition to labeled point => terminate walk
    - x_i is given label of terminal point
    - 

      One possible random walk matrix

      1. Let the *unnormalized* transition matrix be

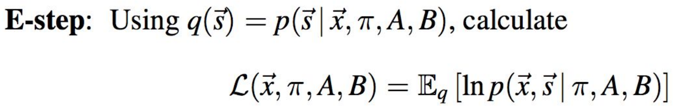
      $$\hat{M}_{ij} = \exp\left\{-\frac{\|x_i - x_j\|^2}{b}\right\}$$

      2. Normalize rows of $\hat{M}$ to get M
      3. If x_i has label y_i, re-define $M_{ii} = 1$

    - Absorbing States: $p(s_t = i \mid s_{t-1} = i) = 1$
        - Random walk will terminate in absorbing state if one exists

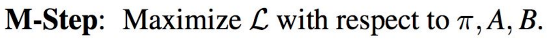
Lecture 21 - Hidden Markov Models

- Hidden Markov Model => assume hidden sequence of discrete states
    - Draw observation from distribution associated with states
        - $x_i = \mu_1 + \xi_i$ if $s_i = 1$
    - Problem definition:
        - SxS Markov Transition Matrix A
        - Initial state distribution π for selecting the first state
        - A state-dependent emission distribution, $\text{Prob}(x_i \mid s_i = k) = p(x_i \mid \theta_{si})$
    - Continuous HMM: $p(x|\theta_s)$ is a continuous distribution, often Gaussian
    - Discrete HMM: $p(x|\theta_s)$ is a discrete distribution, θ_s a vector of probabilities
        - Discrete valued observations
        - Case: Let B be a matrix, where $B_{s,:} = \theta_s$ (Emissions matrix)
    - State Estimation:
        - Estimate: State probability for x_i using "forward-backward algorithm,"
        - $p(s_i = k \mid x_1, \ldots, x_T, \pi, A, B)$
    - State Sequence:
        - Estimate: Most probable state sequence using the "Viterbi algorithm,"
        - $s_1, \ldots, s_T = \text{argmax}_s\, p(s_1, \ldots, s_T \mid x_1, \ldots, x_T, \pi, A, B)$
    - Maximum Likelihood
        - Estimate: HMM parameters π, A, B using maximum likelihood
        - $\pi_{ML}, A_{ML}, B_{ML} = \text{argmax}_{\pi,A,B}\, p(x_1, \ldots, x_T \mid \pi, A, B)$
- Learning the HMM
    - Assuming discrete model, use EM

      E-step: Using $q(\vec{s}) = p(\vec{s} \mid \vec{x}, \pi, A, B)$, calculate

      $$\mathcal{L}(\vec{x}, \pi, A, B) = \mathbb{E}_q\left[\ln p(\vec{x}, \vec{s} \mid \pi, A, B)\right]$$

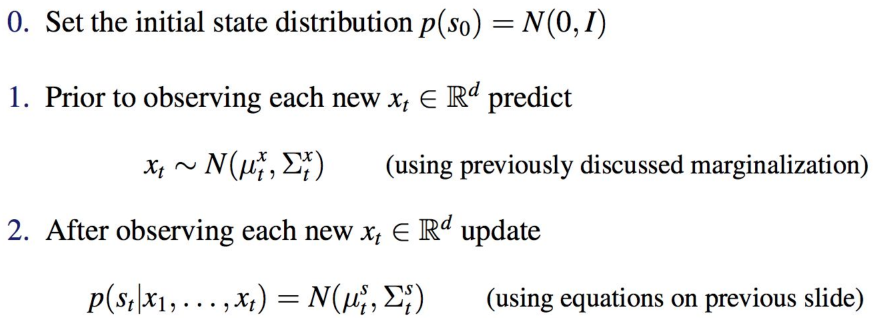
        - 

          M-Step: Maximize $\mathcal{L}$ with respect to π, A, B.
            - $A_{j,k}$ is the expected fraction of transitions j → k when we start at j
            - $B_{k,v}$ is the expected fraction of data coming from state k and equal to v

        - Set expectation of state sequence given A, B, π
        - E-Step: Find A, B, π that maximize likelihood of observed data
        - M-Step: Integrate out state transition sequence (because it's hidden!)
- HMM + Speech Recognition
    - Model with states representing phonemes
    - w_{new} = argmax$_w$ p(x_{new} | π_w, A_w, B_w) ← requires forward-backward
        - This is a Bayes classifier with a uniform prior on the word

Lecture 22 - Continuous State-Space Models

- Continuous State Markov Models => s ∈ R^d (instead of s ∈ {1,...,S})
    - Only learns hidden state sequence
- Linear Gaussian Markov Model
    - The most basic continuous-state version of the Hidden Markov Model is called a Linear Gaussian Markov Model (also called the Kalman Filter)
        - $$\underbrace{s_t = Cs_{t-1} + \epsilon_{t-1},}_{\text{latent process}} \qquad \underbrace{x_t = Ds_t + \varepsilon_t}_{\text{observed process}}$$
        - s_t ∈ R^p is a continuous-state latent (unobserved) Markov process
        - x_t ∈ R^d is a continuous-valued observation
    - Learning:
        - Given the sequence (x_1,x_2,x_3,...), where each x ∈ R^d, the goal is to learn state sequence (s_1,s_2,s_3,...)
        - All distributions are Gaussian: p(s_{t+1} = s|s_t) = N(C_{st}, Q), p(x_t = x|s_t) = N(D_{st}, V)
        - No A, B to learn, only posterior distributions p(s_t | x_1,...,x_T)
- The Kalman Filter (setup as above)



0. Set the initial state distribution $p(s_0) = N(0, I)$

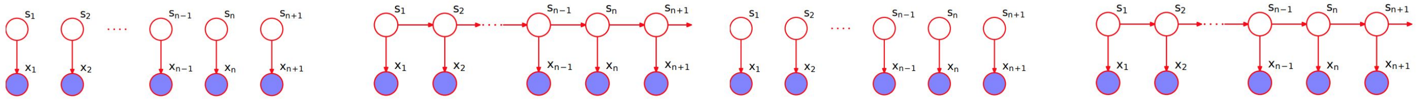
1. Prior to observing each new $x_t \in \mathbb{R}^d$ predict

$$x_t \sim N(\mu_t^x, \Sigma_t^x) \qquad \text{(using previously discussed marginalization)}$$

2. After observing each new $x_t \in \mathbb{R}^d$ update

$$p(s_t|x_1, \ldots, x_t) = N(\mu_t^s, \Sigma_t^s) \qquad \text{(using equations on previous slide)}$$

- GMM + Continuous HMM => discrete state space, distribution on observations
- Probabilistic PCA + Linear Gaussian Markov Model => continuous state space, distribution on observations



Gaussian mixture model
- $s_t \sim \text{Discrete}(\pi)$
- $x_t|s_t \sim N(\mu_{s_t}, \Sigma_{s_t})$

Continuous hidden Markov model
- $s_t|s_{t-1} \sim \text{Discrete}(A_{s_{t-1}})$
- $x_t|s_t \sim N(\mu_{s_t}, \Sigma_{s_t})$

||

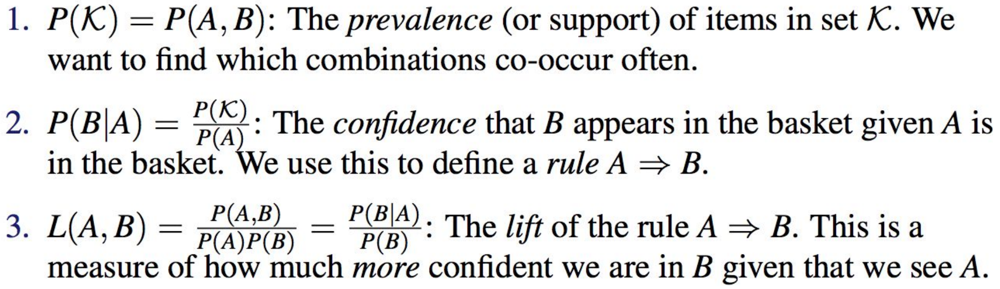
Probabilistic PCA
- $s_t \sim N(0, Q)$
- $x_t|s_t \sim N(Ds_t, V)$

Linear Gaussian Markov model
- $s_t|s_{t-1} \sim N(Cs_{t-1}, Q)$
- $x_t|s_t \sim N(Ds_t, V)$

- However for GMM and PCA, s_t not dependent on s_{t-1} (don't have Markov Property)

Lecture 23 - Association Analysis

- Association Analysis => find highly probably subsets of data
- Market Basket Analysis => analyze patterns to define strong association rules
- Association Analysis Setup:

    - p different objects indexed by {1, . . . , p}
    - A collection of subsets of these objects $X_n \subset \{1, . . . , p\}$. Think of X_n as the index of things purchased by customer n = 1, . . . , N
    - Gives 2^p possible vectors
- Quantities of interest:
    - 

      1. $P(\mathcal{K}) = P(A, B)$: The *prevalence* (or support) of items in set $\mathcal{K}$. We want to find which combinations co-occur often.
      2. $P(B|A) = \frac{P(\mathcal{K})}{P(A)}$: The *confidence* that B appears in the basket given A is in the basket. We use this to define a *rule* $A \Rightarrow B$.
      3. $L(A, B) = \frac{P(A,B)}{P(A)P(B)} = \frac{P(B|A)}{P(B)}$: The *lift* of the rule $A \Rightarrow B$. This is a measure of how much *more* confident we are in B given that we see A.
- Apriori Algorithm
    - Quickly find all subsets of K that have P(K) > threshold
        - Such a K will contain items that appear in at least (N x threshold) baskets
    - Use properties about P(K) to reduce number of sets checked
        - Start with K of size 1 item, 2 items, 3 items, …
        - Sets of size K-1 that survive determine sets of size K to check
        - Finds every set K such that P(K) > threshold
    - Frequency Dependence
        - If the set K is not big enough, then $K' = K \cup A$ with $A \subset \{1, . . . , p\}$ is not big enough. In other words: $P(K) < t$ implies $P(K') < t$
            - e.g., Let K = {a, b}. If these items appear together in x baskets, then the set of items K' = {a, b, c} appears in ≤ x baskets since $K \subset K'$
            - Mathematically: $P(K') = P(K, A) = P(A|K)P(K) \leq P(K) < t$
        - By the converse, if $P(K) > t$ and $A \subset K$, then $P(A) > P(K) > t$
        - Intuitively, if a set does not meet threshold, adding items cannot help
    - Finding Rules
        - $P(A|B) = P(K) / P(B)$
        - If $P(K) > t$ and A and B partition K, then $P(A) > t$ and $P(B) > t$
        - Since Apriori found all K such that $P(K) > t$, it found P(A) and P(B), so we can calculate P(A|B) without counting again

Lecture 24 - Model Selection

- Once we've selected model type, what order?
    - Gaussian mixture model: How many Gaussians?
    - Matrix factorization: What rank?
    - Hidden Markov models: How many states?
- Problem: more degrees of freedom brings more overfitting
- Solutions:
    - Stability
    - Bayesian non-parametric methods
    - Penalization
- Penalizing model complexity
    - Generally: define penalty function on # of model parameters
    - Instead of maximizing L, minimize −L and add the defined penalty
    - Two popular penalties are:
        - Akaike information criterion (AIC): $-L + K$

        - Bayesian information criterion (BIC): $-L + \frac{1}{2} K \ln N$
    - E.g. NMF with rank R, dimension $M_1 \times M_2$
        - AIC → (M1 + M2) R
        - BIC → $\frac{1}{2} (M_1 + M_2) R \ln(M_1 M_2)$
    - BIC => choose model with largest marginal likelihood of the data by integrating out all parameters
        - Approximate this integral using a second-order Taylor expansion

Course Concept Summary

Model Types =>

- Predictions:
    - Generative: generates prediction by modeling x, joint likelihood P(x,y)
        - Bayes classifier, Gaussian mixture models, Linear Discriminant Analysis
    - Discriminative: generates prediction without modeling x
        - Logistic regression, Perceptron, support vector machine.
- Assumptions:
    - Probabilistic:
        - Bayes classifier, logistic regression, ML and MAP
    - Non-probabilistic:
        - Perceptron, SVM, Decisions Trees, K-Means

Bayesian Inference =>

- P(A|B) = P(B|A)P(A)/P(B)
- P(A|B) => posterior, P(B|A) => likelihood
- P(A) => prior, P(B) => marginal

Regression =>

- Regression => maps input to a continuous output space (vs. classification's discrete output space)
- Least Squares/Max Likelihood => Maximize likelihood of labels given data and params w
    - $(X^TX)^{-1}X^Ty$
    - p(y|x,w)
- Ridge Regression/MAP => Maximize posterior of parameters and labels given data
    - $(\lambda I + X^TX)^{-1}X^Ty$
    - p(w,y|x)
- MAP solution has lesser norm than that of ML because λ (penalty term) pulls solution towards origin
- ML is unbiased estimator of w
- Bayesian Linear Regression
    - MAP solution is $(\lambda\sigma^2 I + X^TX)^{-1}X^Ty$, predict new data with likelihood on y_0 and posterior on w
- Lasso => L1 Penalty increases sparsity, drives most values to zero

Classification =>

- Classification => maps input to a discrete output space (vs. regression's continuous output space)
- Nearest neighbors => return label y_i of point x_i closest to x_{new}
- K-Nearest neighbors => return K-closest points, label determined by majority vote
- Optimal classifier => returns correct label with highest probability
- Bayes classifier => predicts most likely label $\text{argmax}_{y \in Y}$ p(X=x|Y=y) p(y), thus optimal
- Plug-in classifier => plug observation into class conditional distributions on y, find most likely label
- Naive Bayes => models x as independent given y, hence Naive
- Perceptron => find vector w that will have positive product with all examples from +1 class (and negative with all examples from -1 class)
- Logistic Regression => maximizes likelihood of labels over w, can't solve analytically (linear boundary in R^d)
- Bayesian Logistic Regression => adds penalty term, MAP solution using LaPlace approximation
- Support Vector Machines => find shortest vector that defines hyperplane that perfectly separates data (hyperplane placed at midpoint of and orthogonal to line connecting convex hulls)
- Decision Trees => generally split to minimize entropy
    - Entropy: MAX when $p_1=p_2=...=p_K$ (chaos) => MIN when $p_k = 1$ for some k (order)

  - Fun fact: the only difference between the future and the past is that entropy was lower in the past

Feature Expansion

- Kernels => useful when we only have to work with dot products, perhaps map features to a higher dimension
  - E.g. if we are going to square each dimension of two vectors and then take a dot product between them, we can instead take a dot product first and then square (loosely)
- Gaussian Processes => uses measure of the similarity between points (kernel function) to predict the value for an unseen point from training data. The prediction is not just an estimate for that point, but also has uncertainty information—is itself a Gaussian distribution

Ensemble Methods =>

- Bootstrap => estimate statistics by resampling and averaging
- Bagging => train models by resampling, predict new data by averaging bagged predictors
- Random Forests => Reduce correlation by using random subset of dimensions (bagged trees are correlated)
- Boosting => combine many weak classifiers, weight samples to improve every iteration (use "hard" examples to improve)

Unsupervised Learning =>

- Focus on p(x), finding model that explains data well
- K-Means clustering => Uses squared Euclidean distance, non-convex so must iterate:
  - Fix μ, find c
  - Fix c, find μ
- Max likelihood => maximizes likelihood of data givens parameters
- Coordinate ascent => solve for one group of parameters given another
- Expectation Maximization => coordinate ascent instance
  - E-Step: Set expected value of θ_2 given θ_1
  - M-Step: Set θ_1 that maximizes likelihood of data given θ_2
- Mixture model => defines prior on cluster assignment c_i and likelihood on x_i given c_i
- Gaussian mixture model => with EM:
  - Find expected weight of each observations towards each cluster
  - Set μ, Σ, π to maximize likelihood of data
- Matrix Factorization => predict missing values by projecting M into low-d space where correlated users and objects are “close”
- Probabilistic Matrix Factorization => find matrices projecting users, objects into low-d space that maximize probability of data using coordinate ascent over U and V
- Topic Modeling and Latent Dirichlet Allocation => finds topics and assign documents
- Non-negative Matrix Factorization => learns non-negative factorization that minimizes particular error, maximizes likelihood of data (parts based)
- Principal Component Analysis => Principal components define low-d space where original observations are projected, explain the most variance in data (first eigenvector is first P.C., second is second, etc.)
- Markov Property => sequential learning (not i.i.d.)
- Random Walk Classifier => semi-supervised learning
  - Move, transition to nearby point with higher probability and terminate when labeled point is reached, returning label of terminal points
- Hidden Markov Models => assume hidden sequence of discrete states
  - Continuous HMM => continuous observations
  - Discrete HMM => discrete observations
  - Forward/Backward algo: Estimate state probability
  - Viterbi algo: Estimate state sequence

- Maximum Likelihood and HMM with Expectation Maximization
    - Set expectation of state sequence given A, B, π
    - Find A, B, π that maximize likelihood of observed data
- Continuous State Markov Models => only learns hidden state sequence
- Linear Gaussian Markov Model (aka Kalman Filter) =>
    - predict latent continuous state variable given observed process
    - only learn $p(s_t|x_1,...,x_T)$, not A, B
-

State	Markov Property	No Markov Property
Discrete	Continuous HMM	Gaussian Mixture Models
Continuous	Linear Gaussian Markov Models	PCA

-

Association Analysis =>

- Find highly probably subsets of data
- P(A,B): Support/Prevalence => how often does K occur
- P(B|A): Confidence => confidence that B appears given a does
- P(B|A)/P(B): Lift => how much more confident that B appears given a does
- Apriori Algorithm => Start with baskets of 1 item, increase in size and use frequency dependence to trim lattice until no surviving baskets
- Association Rules => don't need to count again because apriori algorithm already did so

Model Selection =>

- Given model type, still need to determine order
- Use penalties on degrees of freedom to find model order K
    - Bayesian Information Criterion: $-L + \frac{1}{2} K \ln N$
    - E.g. NMF with rank R, dimension $M_1 \times M_2$
        - BIC $\rightarrow \frac{1}{2} (M_1 + M_2) R \ln(M_1 M_2)$

∈ μ Σ φ π λ ξ σ