General information

L0 norm- number of nonzero elements, L1 norm = sum of abs value of elements, L2 norm = sqrt (sum of (elements squared))
Variance = 1/2 (sum of (elements - mean)^2))

$ln(ab) = ln(a) + ln(b)$, (so $ln(\Pi\, x) = \sum ln(x)$), $ln(e^x) = x$

$p(A|B) = p(B|A)p(A)/p(B)$
$p(A,B) = p(A|B)p(B) = p(B|A)p(A)$

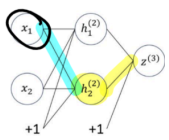
$$P(X) = \sum_{i=1}^{n} P(X, Y = y_i) = \sum_{i=1}^{n} P(X|Y = y_i)P(Y = y_i).$$

If we are trying to find the *x* that maximizes some expression, we can discard terms that don't depend on *x*

Neural networks

$w_{ji}^{(k)}$ is the weight from node *i* to node *j* in layer *k* ($k \geq 1$)
z is used for nodes pre-activation, while *h* is for nodes post-activation.
The forward pass is computing the value of the node
The backward pass is calculating the gradient to update the weight
When calculating backprop, only include weight you're differentiating with respect to

$$\frac{\partial L}{\partial w_{21}^{(1)}} = \frac{\partial L}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial h_2^{(2)}} \frac{\partial h_2^{(2)}}{\partial z_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial w_{21}^{(1)}}$$



Update step: new_weight = old_weight-eta*(result of backprop)
$ReLU(x) = max(0, x)$, $sigmoid(x) = 1/(1 + e^{-x})$
Output size of a convolutional layer is (input_size - filter_size + 2 * padding) / stride + 1 (size is magnitude of one dimension)
Number of channels in output = number of filters in previous layer
Use Filter F_1 on input X_1 and filter F_2 on input X_2, then add up their results
Number of trainable parameters in a fully connected layer is (# of input neurons) * (# of output neurons)
Number of trainable parameters in a convolutional layer is (filter_width * filter_height * # of input channels + 1) * (# of filters)

Clustering methods

Methods include k-means, spectral, and hierarchical
k-means clustering algorithm: Given dataset with *n* points x̄^(1) to x̄^(n):

1. Randomly choose *k* cluster means μ̄^(1) to μ̄^(n)
2. Iterate between two steps until convergence:
   a. Assign each data point x̄^(j) to cluster *i* with mean μ̄^(i) that is closest to x̄^(j)
   b. Recompute each cluster mean μ̄^(i) based on all data points assigned to cluster *i*

In spectral clustering, the number of 0 eigenvectors is the number of connected components in the graph.
Weight matrix W where W_{ij} is the similarity between points x̄^(i) and x̄^(j), and the similarity between any point and itself is 1
Degree matrix D where D_{ii} is the sum of all weights in row *i* of W
Laplacian matrix L = D - W. You find the eigenvalues/eigenvectors of this, put these eigenvectors as columns ordered from smallest to largest eigenvalue from left to right. Each row corresponds to a point.

Collaborative filtering

$$\min_{\hat{Y}} J(\hat{Y}) = \min_{\hat{Y}} \frac{1}{2} \sum_{(a,i)\in D} (Y_{ai} - \hat{Y}_{ai})^2 + \frac{\lambda}{2} \sum_{a,i} (\hat{Y}_{ai})^2$$

Y_{ai} : observed user-item rating that user *a* assigned item *i*
$\hat{Y}_{ai}$: filled approximation of the user-item rating
D : set of observed *(a, i)* ratings
λ : tunable hyperparameter

Minimize J(U,V) by:

1. Treat V as constant and minimize J(U,V) w.r.t U
2. Treat U as constant and minimize J(U, V) w.r.t V
3. Repeat until some stopping criteria

Maximum Likelihood Estimation

$$p_{\bar{\theta}}(X) = \prod_{i=1}^{n} p_{\bar{\theta}}(\bar{x}^{(i)})$$

"Likelihood of dataset X"

Process:

1. Compute log-likelihood of distribution
2. Optimize with respect to parameter
   a. Find derivative w.r.t parameter
   b. Set derivate equal to 0
   c. Solve for parameter

Estimation step:

$$p(j|i) = \frac{\gamma_j \mathcal{N}(\bar{x}^{(i)}|\bar{\mu}_j, \sigma_j^2)}{\sum_{t=1}^{k} \gamma_t \mathcal{N}(\bar{x}^{(i)}|\bar{\mu}_t, \sigma_t^2)}$$

Maximization step:

$$\hat{n}_j = \sum_{i=1}^{n} p(j|i)$$

$$\hat{\gamma}_j = \frac{\hat{n}_j}{n}$$

$$\hat{\mu}^{(j)} = \frac{1}{\hat{n}_j} \sum_{i=1}^{n} p(j|i)\bar{x}^{(i)}$$

$$\hat{\sigma}_j^2 = \frac{1}{d\hat{n}_j} \sum_{i=1}^{n} p(j|i)||\bar{x}^{(i)} - \hat{\mu}^{(j)}||^2$$

$$p(\bar{x}^{(i)}; \bar{\theta}) = \sum_{j=1}^{k} \gamma_j \mathcal{N}(\bar{x}^{(i)}; \bar{\mu}_j, \Sigma_j)$$

If it is known what cluster each data point belongs to, simplifies to

$$\delta(j|i) = \begin{cases} 1, & \text{if } z^{(i)} = j \\ 0, & \text{otherwise} \end{cases}$$

$$p(\bar{x}^{(i)}|z^{(i)}; \bar{\theta}) = \sum_{j=1}^{k} \delta(j|i)\mathcal{N}(\bar{x}^{(i)}; \bar{\mu}_j, \Sigma_j)$$

MLE for a spherical Gaussian:

$$\bar{\mu}_{MLE} = \frac{\sum_{i=1}^{n} \bar{x}^{(i)}}{n}$$

$$\sigma_{MLE}^2 = \frac{\sum_{i=1}^{n} \|\bar{x}^{(i)} - \bar{\mu}_{MLE}\|^2}{nd}$$

Gaussian Mixture Models

Notation:

$\gamma_1, \ldots, \gamma_k$	mixing weights
$\bar{\mu}_1, \ldots, \bar{\mu}_k$	cluster means
$\Sigma_1, \ldots, \Sigma_k$	cluster covariance matrices
$\bar{\theta} = [\gamma_1, \ldots, \gamma_k, \bar{\mu}_1, \ldots, \bar{\mu}_k, \Sigma_1, \ldots, \Sigma_k]$	model parameters

Normal distribution:

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

The probability density function of a spherical Gaussian is:

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

where x̄ and μ̄ are d × 1 vectors corresponding to a d-dimensional feature space (for R2, d = 2). Σ is a d × d covariance matrix.
$\delta(j|i)$ is 1 if point $x^{(i)}$ belongs to cluster *j* and 0 otherwise, $p(j|i)$ is the probability that point $x^{(i)}$ belongs to cluster *j*

Bayesian Networks

Helps write pdf: $p(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | x_{pa_i})$ where x_{pa_i} is the set of parents of node x_i

1. Draw graph containing only variables of interest and all their ancestors
2. Connect any nodes with a common child and make graph undirected

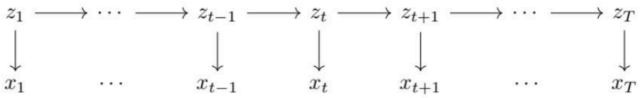
If there is no path between two nodes, they are marginally independent
If all paths between two nodes pass through node X, the two nodes are conditionally independent given node X

Hidden Markov Models

Definition: type of graphical model where we assume data comes from a sequence of hidden states, where:

- Each observation is associated with a state
- Each state is independent of all other previous states and observations given the parent state
- Each observation is independent of all other observations and states given the associated state

Diagram:



Where *x*'s are known observations and *z*'s are unknown hidden states we want to estimate.

- H is the set of possible hidden states, $|H| = M$
- O is the set of possible observations, $|O| = N$
- $\theta = [A, B, \pi]$
  - A is an $M \times M$ matrix where $A(h_i, h_j) = P(z_{t+1} = h_j \mid z_t = h_i)$ is the probability of transitioning from state h_i to h_j
  - B is an $M \times N$ matrix where $B(h_i, o_\ell) = P(x_t = o_\ell \mid z_t = h_i)$ is the probability of emitting symbol o_ℓ from state h_i.
  - π is an $M \times 1$ vector where $\pi(h_i) = P(z_1 = h_i)$ is the probability of starting in state h_i.

Likelihood of a sequence of observations and states:

$$P(x_1, \ldots, x_T, z_1, \ldots, z_T; \theta) = \pi(z_1) \prod_{t=1}^{T-1} A(z_t, z_{t+1}) \prod_{t=1}^{T} B(z_t, x_t)$$

Other stuff

$$t^{(i)} \sim \text{Bernoulli}(t; \sigma(\theta x^{(i)} + b))$$

$$P(y_n | x_n, \theta) = \prod_{i=1} P(y^{(i)} | x^{(i)}, \theta)$$

$$= \prod_{i:y^{(i)}=+1} \sigma(\theta x^{(i)} + b) \prod_{i:y^{(i)}=-1} (1 - \sigma(\theta x^{(i)} + b))$$

(a) Show that

$$\arg\max_h p(h|i) = \arg\min_h \frac{1}{\sigma_h^2} \|\bar{x}^{(i)} - \bar{\mu}^{(h)}\|^2 + d \ln \sigma_h^2 - 2 \ln \gamma_h$$

E step: for all $i = 1, \ldots, n$

Calculate $p(j|i) = \frac{\gamma_j \mathcal{N}(\bar{x}^{(i)} | \bar{\mu}^{(j)}, \sigma_j^2)}{\sum_{h=1}^{k} \gamma_h \mathcal{N}(\bar{x}^{(i)} | \bar{\mu}^{(h)}, \sigma_h^2)}$ for all $j = 1, \ldots, k$

$$\mathcal{N}(\bar{x} | \bar{\mu}, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(\frac{-\|\bar{x} - \bar{\mu}\|^2}{2\sigma^2}\right)$$

$$\arg\max_h p(h|i) = \arg\max_h \frac{\gamma_h \mathcal{N}(\bar{x}^{(i)} | \bar{\mu}^{(h)}, \sigma_h^2)}{\sum_a \gamma_a \mathcal{N}(\bar{x}^{(i)} | \bar{\mu}^{(a)}, \sigma_a^2)}$$

$$= \arg\max_h \gamma_h \mathcal{N}(\bar{x}^{(i)} | \bar{\mu}^{(h)}, \sigma_h^2)$$

$$= \arg\max_h \frac{\gamma_h}{(2\pi\sigma_h^2)^{d/2}} \exp\left(\frac{-\|\bar{x}^{(i)} - \bar{\mu}^{(h)}\|^2}{2\sigma_h^2}\right)$$

$$= \arg\max_h \ln\left(\frac{\gamma_h}{(2\pi\sigma_h^2)^{d/2}} \exp\left(\frac{-\|\bar{x}^{(i)} - \bar{\mu}^{(h)}\|^2}{2\sigma_h^2}\right)\right)$$

$$= \arg\max_h \left(\ln(\gamma_h) - \frac{d}{2} \ln(2\pi\sigma_h^2) - \frac{\|\bar{x}^{(i)} - \bar{\mu}^{(h)}\|^2}{2\sigma_h^2}\right)(-2)$$

$$= \arg\min_h -2\ln(\gamma_h) + d\sigma_h^2 + \frac{\|\bar{x}^{(i)} - \bar{\mu}^{(h)}\|^2}{\sigma_h^2}$$

$\arg\max f(x) = \arg\max \ln(f(x))$

iii. We are interested in doing regression, in which the input to our model will be strings of arbitrary length, and the output will be a real number. We plan to extend ordinary least-squares regression to use a kernel function. Emre suggests using the following kernel:

$$K(\bar{x}, \bar{x}') = \sum_{w \in \text{alphabet}} (\#\text{ of occurrences of } w \text{ in } \bar{x}) \cdot (\#\text{ of occurrences of } w \text{ in } \bar{x}'),$$

where alphabet is the English alphabet from "a" to "z". Emre performs a kernelized regression, finding parameters α_i, so that the predictions are of the form:

$$f(\bar{x}) = \sum_{i=1}^{n} \alpha_i K(\bar{x}^{(i)}, \bar{x})$$

i	$\bar{x}^{(i)}$	$y^{(i)}$
1	upelkuchen	10
2	madhatter	1
3	witzgindy	3

a) (2 pts) What is the feature mapping associated with Emre's kernel? You may use a mathematical representation of the mapping function $\phi(\bar{x})$ or describe it in English. Your answer must include the dimension d of the mapped feature vectors.

Solution: Each feature is the number of occurrences for a specific letter in the English alphabet. The dimension $d = 26$ is the number of letters in the English alphabet

b) (2 pts) Give the predicted label associated with the test input $\bar{x} =$ ziggy. State your answer in terms of the α_i parameters.

Solution:

$$\begin{aligned} f(\text{ziggy}) &= \alpha_1 K(\bar{x}^{(1)}, \bar{x}) + \alpha_2 K(\bar{x}^{(2)}, \bar{x}) + \alpha_3 K(\bar{x}^{(3)}, \bar{x}) \\ &= \alpha_1 K(\text{upelkuchen}, \text{ziggy}) + \alpha_2 K(\text{madhatter}, \text{ziggy}) + \alpha_3 K(\text{witzgindy}, \text{ziggy}) \\ &= 0\alpha_1 + 0\alpha_2 + 6\alpha_3 \\ &= 6\alpha_3 \end{aligned}$$

Algorithm 1 Perceptron Algorithm

$k \leftarrow 0$, $\bar{\theta}^{(k)} \leftarrow \bar{0}$
while at least one point is misclassified **do**
  for $i = 1, \ldots, n$ **do**
    if $y^{(i)}(\bar{\theta}^{(k)} \cdot \bar{x}^{(i)}) \leq 0$ **then**
      $\bar{\theta}^{(k+1)} \leftarrow \bar{\theta}^{(k)} + y^{(i)}\bar{x}^{(i)}$
      $k{+}{+}$
    end if
  end for
end while
return $\bar{\theta}^{(k)}$

original problem: $\min_{\bar{w}} f(\bar{w})$ s.t. $h_i(\bar{w}) \leq 0$ for $i = 1, \ldots, n$

Lagrangian: $L(\bar{w}, \bar{\alpha}) = f(\bar{w}) + \sum_{i=1}^{n} \alpha_i h_i(\bar{w})$ $\alpha_i \geq 0$

Soft margin SVM:

$$\min_{\bar{\theta}, b, \bar{\xi}} \frac{\|\bar{\theta}\|^2}{2} + C \sum_{i=1}^{n} \xi_i$$

subject to $y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) \geq 1 - \xi_i$
for $i \in \{1, \ldots, n\}$ and $\xi_i \geq 0$