Topics Since the Midterm

- K-NN regression and classification
- Kernel regression
- Perceptron
- Kernel trick, kernelized perceptron
- SVM
- K-means
- PCA
- GMM
- EM for GMM
- Bayes optimal classification
- Naive Bayes
- Bayes' Nets
- Deep learning

### Instance-based Learning

Online learning, margin-based approaches

### Unsupervised Learning

Structural models

### K-NN

Predict using any of K nearest neighbors

Possible distance functions:
- Euclidean
- Manhattan
- Other
Biais-variance tradeoff as function of $K$.

Note: KNN regression has discontinuities.

**Kernel regression - classification**

"soft" nearest neighbors

Given kernel function $k_x$,

$$
\hat{y}_q = \frac{\sum_{i} k_x(x_i, x_q) y_i}{\sum_{i} k_x(x_i, x_q)}
$$

Kernel bandwidth $d$: bias/variance

Gaussian kernel: $k_x(x_i, x_q) = \exp\left(-\frac{(x_i - x_q)^2}{2d^2}\right)$

Boxcar kernel, others
Perceptron

Goal: Learn linear decision boundary determined by \( w \)

Training:

\[ w(0) = 0 \]

At each \( t \):

\[ y_t^+ = \text{sign}(w(t) \cdot x_t) \]

If \( y_t^+ = y_t \)

Do nothing

Else

\[ w(t+1) = w(t) + y_t x_t \]

Perceptron loss function:

\[
L(x, w) = \begin{cases} 
0 & \text{if } y(x, w) \geq 0 \\
-y(w \cdot x) & \text{o.w.}
\end{cases}
\]

Distance from boundary
Minimize: \( \min_w \frac{1}{N} \sum_{i=1}^{N} L(w, x_i) = \frac{1}{N} \sum_{i=1}^{N} (-y_i (w^T x_i))^+ \)

\( \nabla_w L(x, w) = -y x \)

\( \nabla_w L(x, w) = 0 \)

Batch hinge min:

\( w^{t+1} = w^t - \eta \frac{1}{N} \sum_{i=1}^{N} \left[ \min \{ y_i (w^T x_i), 0 \} \right] y_i x_i \)

Perceptron is SGD for hinge loss
Kernel perceptron

Data not linearly separable: compute $\phi(x)$

$$\phi(x) = \begin{pmatrix} 1 \\ x \\ x^2 \\ \vdots \end{pmatrix}$$

$$w(x) = \sum_{i,j} y_j x_j \Rightarrow \text{sign}(w(x) \cdot x) = \text{sign}\left[ \sum_{i \in M(x)} y_j \phi(x_j \cdot x) \right]$$

Replace $x$ with $\phi(x)$

$$\text{sign}(w(x) \cdot \phi(x)) = \sum_{i \in M(x)} y_i \phi(x_i \cdot x) = \sum_{i \in M(x)} y_i \phi(x_i, x)$$

No computing $\phi$. This is the "kernel trick!"

Instead of keeping track of $w(x)$, just remember $M(x)$.

Many kernel functions...
SVM

Max - margin

\[ w \cdot x_i + w_0 = \pm 1 \]

with

SVM objective:

\[ \min ||w||_2^2 \text{ s.t.} \]

\[ y_i (w \cdot x_i + w_0) \geq 1 \quad \forall i \in 1, ..., N \]

Data not linearly separable:

\[ \text{violation} = \begin{cases} 0 & \text{if } |1 - y_i (w \cdot x_i + w_0)| < C \\ 1 - y_i (w \cdot x_i + w_0) & \text{otherwise} \end{cases} \]

Trade off margin violation vs. \( ||w||_2 \):

\[ \min ||w||_2^2 + C \sum_{i=1}^{N} \max(0, 1 - y_i (w \cdot x_i + w_0)) \text{ as objective when data not linearly separable.} \]

SVM minimizes regularized hinge loss (similar to perceptron)

Can do SGD, kernel trick; see slides for details.
**K-Means**

$$\text{minimize: } \sum_{i=1}^{k} \sum_{j=1}^{d} \| x_{ij} - c_{ij} \|^2$$

Choosing $k$, plot distortion lead to "kink".

**PCA**

Given data $X$, select a basis $(u_1, \ldots, u_k)$ $k < D$ that gives the "best" lower-dim projection of $X$.

Projection of $x_i$ is $z_i$.

$$z_i = (z_i[1], \ldots, z_i[k]), \text{ with } z_i[j] = x_i \cdot u_j$$

Assume mean-centered data are mean-centered.

Reconstruct $x_i$ from $z_i$ as

$$\hat{x}_i = \sum_{j=1}^{k} z_i[j] u_j$$

"Best" projection, the one that minimizes

$$\sum_{i=1}^{N} (x_i - \hat{x}_i)^2$$
Big result: The vectors $u_1, \ldots, u_K$ that give the best reconstruction are the $K$ eigenvectors of

$$
\Sigma = \frac{1}{N} \sum_{i=1}^{N} x_i x_i^T
$$

with the largest eigenvalues. The eigenvalues are proportional to the variance of the data, projected onto each $u_i$.

---

**GMM**

Generative process:

1. Sample a class $k_i \sim \text{cat}(\Pi)$
2. Sample data $x_i \sim \mathcal{N}(\mu_{k_i}, \Sigma_{k_i})$

Likelihood:

$$
P(x_i | \Theta) = \sum_{j=1}^{K} \Pi_j \ p(x_i | \mu_j, \Sigma_j)
$$

why GMM: lets you represent complex, multimodal distributions.

---

**EM**

Motivation: GMM has observed $x_i$, unobserved $z_i$, parameters $\Theta = \{ \Pi_j, \mu_j, \Sigma_j \}_{j=1}^{K}$

want to learn $\Theta$, take best guess at $z_i$. 
Intuition: Alternate by updating $O$ using a good guess at $z$, and updating our guess at $z$ using our current $O$.

Instead of making "hard" assignments $z_i$, make "soft" assignments $p(z_i)$.

See slides for more details.

Pathologies of EM: A mixture component can collapse onto a single paint:

![Diagram of cluster collapse]

No. Bayes

Features $X$, output class $Y$.

No. Bayes assumption: $f(x, y) = P(y) \prod_{i=1}^{D} P(x_i | y)$

Thus the distribution factorizes according to:

$$P(x_1, \ldots, x_D, y) = P(y) \prod_{d=1}^{D} P(x_d | y)$$

To predict $y$: $\hat{y} = \arg\max_y \prod_{d=1}^{D} P(x_d | y)$.
MLE for Naive Bayes: just counts

\[
P(X_j = x_j | Y = y) = \frac{\text{count}(X_j = x_j, Y = y)}{\text{count}(Y = y)}
\]

NB is unrealistic, but often performs well in practice.

Bayesian Networks

- Represent joint distribution as directed acyclic graph (DAG)
- Vertices: Random variables
- Edges: Conditional dependencies
- One CPT for each variable: \( P(\text{variable} | \text{parents}) \)

\[
P(A, B, C, D) = P(C | A, B) \frac{1}{1} P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \text{parents}(X_i))
\]
Assume each variable is binary.

# parameters needed to specify the joint: 10 (check yourself)

# parameters without knowing graph structure: $2^5 - 1 = 31$.

Local Marker Assumption:

\[ \mathbf{x} \text{ is independent of non-descendants given parents} \]

\[ \text{Headache} \perp \text{Flu, Allergy, Nose} \mid \text{Sinus} \]

Explaining away: Flu, allergy are marginally independent. However, given Sinus, they become dependent. (Think about why...