Announcements
Review: PCA

- Finds subspace that minimized residuals $= \text{maximizes variance.}$

- Compute data covariance $S = \frac{1}{N} \sum_i (x_i - \mu)(x - \mu)^T$

- Calculate $\phi_1, \ldots, \phi_d$ that are orthonormal eigenvectors of $S$ corresponding to the eigenvalues $\lambda_1 \geq \ldots \geq \lambda_d$.

- The PCA subspace is given by
  $$\Phi = [\phi_1, \ldots, \phi_k].$$

- Low-dim. representation: $z = \Phi^T (x - \mu)$

- Reconstruction: $\tilde{x} = \mu + \Phi z$
Plan for today

- Finish discussion of PCA and Probabilistic PCA
- Feature selection and ensemble methods
PCA and compression

- Suppose we have computed $k$-dimensional PCA representation.

- We need to transmit/store:
  - The $1 \times d$ mean vector;
  - The $k \times d$ projection matrix.

- For each new example, we only need to convey $z$ which is $1 \times k$.
  - If we transmit $N$ examples, we need $d + dk + Nk$ numbers instead of $Nd$.
  - Tradeoff between accuracy and compression.
PCA and classification

- A very common methodology: perform PCA on all data and learn a classifier in the low-dimensional space.

- Tempting: may turn computationally infeasible into practical.

- Careful! Direction of largest variance need not be the most discriminative direction.
PCA and Gaussians

- Suppose \( p(x) = \mathcal{N}(x; \mu, \Sigma) \).

- Recall:
  \[
  \Sigma = R \begin{bmatrix}
  \lambda_1 & \cdots \\
  & \ddots \\
  & & \lambda_d
\end{bmatrix} R^T.
  \]
  Rotation \( R \) determines the orientation of the ellipse; \( \text{diag}(\lambda_1, \ldots, \lambda_d) \) specifies the scaling along the principal directions.

- Suppose we take all \( d \) eigenvectors of \( \Sigma \).

- Columns of \( \Phi \) are \( d \) orthonormal eigenvectors \( \Rightarrow \) it’s a rotation matrix.
Probabilistic PCA

- Probabilistic PCA is a method of fitting a constrained Gaussian ("pancake"):

\[
\Sigma = \Phi \begin{bmatrix}
\lambda_1 & \cdots & 0 & \cdots & \cdots \\
\vdots & \ddots & 0 & \ddots & \\
0 & \cdots & \lambda_k & \cdots & \cdots \\
0 & \cdots & 0 & \sigma^2 & 0 \\
0 & \cdots & \cdots & 0 & \sigma^2
\end{bmatrix} \Phi^T
\]

- ML estimate for the noise variance \( \sigma^2 \):

\[
\sigma^2 = \frac{1}{d-k} \sum_{j=k+1}^{d} \lambda_j
\]
Linear subspaces vs. manifolds

- Linearity assumption constrains the type of subspaces we can find.

- A general formulation: a hidden manifold.

- One possible method: kernel PCA

- Very active area of research...
Summary: unsupervised learning

- Density estimation:
  - parametric closed-form (Gaussian, Bernoulli);
  - non-parametric (kernel-based);
  - semi-parametric (the EM algorithm for mixture models).

- Clustering: $k$-means/medoids, hierarchical, spectral, ...

- Unsupervised dimensionality reduction (PCA).

- Main points:
  - Need to define criterion carefully;
  - usually have to accept local optimum.
Feature selection

- Suppose we are considering a finite number of features (or basis functions).
  \[ \mathbf{x} = [x_1, \ldots, x_d]^T \]

- We are interested in selecting a *subset* of these features, \( x_{s_1}, \ldots, x_{s_k} \), that lead to the best classification or regression performance.

- We have already seen this:
Feature selection

• Suppose we are considering a finite number of features (or basis functions). 
  \( \mathbf{x} = [x_1, \ldots, x_d]^T \)

• We are interested in selecting a \textit{subset} of these features, \( x_{s_1}, \ldots, x_{s_k} \), that lead to the best classification or regression performance.

• We have already seen this: lasso regularization.

• PCA: more like “feature generation”
  
  \[ z_j = \phi_j^T \mathbf{x} \text{ is a linear combination of all } x_1, \ldots, x_d \]
Wrapper versus filter methods

- **Wrapper** methods: try to optimize the feature subset for a given supervised learning algorithm (e.g., for a given classifier).
  - Regularization
  - Greedy methods.

- **Filter** methods: evaluate features based on a criterion independent of a classification/regression method.
  - Information value: good feature contains large amount of information regarding the label.
Mutual information

- **Mutual Information** between the random variables $X$ and $Y$ is defined as the reduction in entropy (uncertainty) of $X$ given $Y$:

$$I(X; Y) \triangleq H(X) - H(X|Y)$$
Mutual information

- **Mutual Information** between the random variables $X$ and $Y$ is defined as the reduction in entropy (uncertainty) of $X$ given $Y$:

\[
I(X; Y) \triangleq H(X) - H(X|Y) = - \sum_x p(x) \log p(x) + \sum_x \sum_y p(x, y) \log p(x | y)
\]
**Mutual information**

- **Mutual Information** between the random variables $X$ and $Y$ is defined as the reduction in entropy (uncertainty) of $X$ given $Y$:

$$I(X; Y) \triangleq H(X) - H(X|Y)$$

$$= - \sum_x p(x) \log p(x) + \sum_y \sum_x p(x, y) \log p(x | y)$$

$$= - \sum_x \sum_y p(x, y) \log p(x) + \sum_x \sum_y p(x, y) \log p(x | y)$$
Mutual information

- Mutual Information between the random variables $X$ and $Y$ is defined as the reduction in entropy (uncertainty) of $X$ given $Y$:

\[
I(X; Y) \triangleq H(X) - H(X|Y)
\]

\[
= - \sum_x p(x) \log p(x) + \sum_x \sum_y p(x, y) \log p(x | y)
\]

\[
= - \sum_x \sum_y p(x, y) \log p(x) + \sum_x \sum_y p(x, y) \log p(x | y)
\]

\[
= \sum_{x, y} p(x, y) \log \frac{p(x | y)}{p(x)}
\]
Mutual Information

- **Mutual Information** between the random variables $X$ and $Y$ is defined as the reduction in entropy (uncertainty) of $X$ given $Y$:

$$I(X;Y) \triangleq H(X) - H(X|Y)$$

$$= - \sum_{x} p(x) \log p(x) + \sum_{x} \sum_{y} p(x,y) \log p(x|y)$$

$$= - \sum_{x} \sum_{y} p(x,y) \log p(x) + \sum_{x} \sum_{y} p(x,y) \log p(x|y)$$

$$= \sum_{x,y} p(x,y) \log \frac{p(x|y)}{p(x)} = \sum_{x,y} p(x,y) \log \frac{p(x|y)p(y)}{p(x)p(y)}$$
Mutual information

- **Mutual Information** between the random variables $X$ and $Y$ is defined as the reduction in entropy (uncertainty) of $X$ given $Y$:

$$I(X; Y) \triangleq H(X) - H(X|Y)$$

$$= -\sum_x p(x) \log p(x) + \sum_x \sum_y p(x, y) \log p(x | y)$$

$$= -\sum_x \sum_y p(x, y) \log p(x) + \sum_x \sum_y p(x, y) \log p(x | y)$$

$$= \sum_{x,y} p(x, y) \log \frac{p(x | y)}{p(x)} = \sum_{x,y} p(x, y) \log \frac{p(x | y) p(y)}{p(x)p(y)}$$

$$= D_{KL} (p(x, y) \parallel p(x)p(y)).$$
\[ I(X; Y) = H(X) - H(X|Y) = D_{KL}(p(X, Y) \parallel p(X)p(Y)) \]

- Continuous version:
  \[ I(X; Y) = \int_y \int_x p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \, dx \, dy. \]

- MI is always non-negative (since KL-divergence is)

- Since \( p(x, y) = p(y, x) \), and \( p(x)p(y) = p(y)p(x) \), MI is symmetric.

- The data processing inequality: for any function \( f \),
  \[ I(X; Y) \geq I(X; f(Y)). \]
Max-MI feature selection: classification

- We can evaluate MI between class label $y$ and a feature $x_j$.

$$I(x_j; y) = \sum_{y \in Y} \int p(x, y) \log \frac{p(x \mid y) p(y)}{p(x)p(y)}$$

- This requires estimating $p(y)$ (easy), $p(x_j)$ and $p(x_j \mid y)$ (may be hard).

- Sanity check: for binary classification problem, $I(x_j; y) \leq 1$ for any feature $x_j$. 
Filter methods: shortcomings

- How many features to include? Where to place the threshold?
Filter methods: shortcomings

- How many features to include? Where to place the threshold?

- Ignores redundancy between features
  - If the same (informative) feature is repeated 100 times, it will get selected 100 times.

- Ignores dependency between features. I.e., $x_1$ and $x_2$ may each be uninformative, but together provide perfect prediction.

- The classifier at hand may take advantage of information in some features but not others.
Wrapper methods

- Wrapper methods are defined for a particular regressor/classifier.

- In general, selecting optimal subset of features is NP-hard
  - Combinatorics: need to consider all $\binom{d}{k}$ subsets.

- A (heuristic) solution: *greedy feature selection.*
Combination of regressors

- Consider linear regression model

\[ y = f(x; w) \underbrace{w_0 \phi_0(x) + w_1 \phi_1(x) + \ldots + w_d \phi_d(x)}_{\equiv 1} \]

- We can see this as a combination of \(d+1\) simple regressors:

\[ y = \sum_{j=0}^{d} f_j(x; w), \quad f_j(x; w) \triangleq w_j \phi_j(x) \]
Forward stepwise regression

\[ y = \sum_{j=0}^{d} f_j(x; \mathbf{w}), \quad f_j(x; \mathbf{w}) = w_j \phi_j(x) \]

- We can build this combination greedily, one function at a time.
- Parametrize the set of functions: \( f(x; \theta), \theta = [w, j] \)
- Step 1: fit the first simple model

\[ \theta_1 = \arg\min_{\theta} \sum_{i=1}^{N} (y_i - f(x_i; \theta))^2 \]
Forward stepwise regression

- **Step 1:** fit the first simple model

\[ \theta_1 = \arg\min_\theta \sum_{i=1}^{N} (y_i - f(x_i; \theta))^2 \]

- **Step 2:** fit second simple model to the residuals of the first:

\[ \theta_2 = \arg\min_\theta \sum_{i=1}^{N} (y_i - f(x_i; \theta_1) - f(x_i; \theta))^2 \]

- . . . Step \( n \): fit a simple model to the residuals of the previous step.

- **Stop** when no significant improvement in training error.

- **Final estimate** after \( M \) steps:

\[ \hat{y}(x) = f(x; \theta_1) + \ldots + f(x; \theta_M) \]
Next time

Ensemble classifiers;
Boosting.