Outline

• Multivariate Linear Regression, Demo
• Cross-Validation, Review
• $k$-NN Classification, Demo
Multivariate Linear Regression

• We want to predict output, such as the median house price, from multi-dimensional observations.

• Each house is a data point $n$, with observations indexed by $j$:

$$
\mathbf{x}^{(n)} = (x_1^{(n)}, \ldots, x_d^{(n)})
$$

• Simple predictor is analogue of linear classifier, producing real-valued $y$ for input $\mathbf{x}$ with parameters $\mathbf{w}$ (assuming $x_0 = 1$):

$$
y = w_0 + \sum_{j=1}^{d} w_j x_j = \mathbf{w}^T \mathbf{x}
$$
Multivariate Data

- Multiple measurements (sensors)
- d inputs/features/attributes
- N instances/observations/examples

\[
X = \begin{bmatrix}
X_1^1 & X_2^1 & \cdots & X_d^1 \\
X_1^2 & X_2^2 & \cdots & X_d^2 \\
\vdots & \vdots & \ddots & \vdots \\
X_1^N & X_2^N & \cdots & X_d^N 
\end{bmatrix}
\]
Multivariate Parameters

Mean: $E[x] = [\mu_1, ..., \mu_d]^T$

Covariance: $\sigma_{ij} \equiv \text{Cov}(X_i, X_j)$

Correlation: $\text{Corr}(X_i, X_j) \equiv \rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$

$\Sigma \equiv \text{Cov}(X) = E[(X - \mu)(X - \mu)^T] = \begin{bmatrix}
\sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\
\sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2
\end{bmatrix}$
Multivariate Normal Distribution

\[ x \sim \mathcal{N}_d (\mu, \Sigma) \]

\[ p(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right] \]

- Mahalanobis distance: \((x - \mu)^T \Sigma^{-1} (x - \mu)\)
  measures the distance from \(x\) to \(\mu\) in terms of \(\Sigma\) (normalizes for difference in variances and correlations)
Bivariate Normal

\[ \text{Cov}(x_1, x_2)=0, \ Var(x_1)=\text{Var}(x_2) \]

\[ \text{Cov}(x_1, x_2)=0, \ Var(x_1)>\text{Var}(x_2) \]

\[ \text{Cov}(x_1, x_2)>0 \]

\[ \text{Cov}(x_1, x_2)<0 \]
\[ \text{Cov}(x_1, x_2) = 0, \ Var(x_1) = \text{Var}(x_2) \]

\[ \text{Cov}(x_1, x_2) > 0 \]

\[ \text{Cov}(x_1, x_2) < 0 \]
Independent Inputs: Naive Bayes

• If $x_i$ are independent, offdiagonals of $\Sigma$ are 0, Mahalanobis distance reduces to weighted (by $1/\sigma_i$) Euclidean distance:

$$p(x) = \prod_{i=1}^{d} p_i(x_i) = \frac{1}{(2\pi)^{d/2} \prod_{i=1}^{d} \sigma_i} \exp \left[ -\frac{1}{2} \sum_{i=1}^{d} \left( \frac{x_i - \mu_i}{\sigma_i} \right)^2 \right]$$

• If variances are also equal, reduces to Euclidean distance
Parametric Classification

- If $p(x | C_i) \sim N(\mu_i, \Sigma_i)$

\[
p(x | C_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \right]
\]

- Discriminant functions

\[
g_i(x) = \log p(x | C_i) + \log P(C_i)
= -\frac{d}{2} \log 2\pi - \frac{1}{2} \log |\Sigma_i| - \frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) + \log P(C_i)
\]
MATLAB Demo

- Multivariate Linear Regression
Cross-Validation

• why validation?
  • performance estimation
  • model selection (e.g. hyper parameters)
• hold-out validation
  • split dataset into training set and test set
  • drawbacks: waste of dataset, estimation of error rate maybe misleading
• cross-validation
Cross-Validation

- random subsampling
- k-fold cross-validation
- leave-1-out cross-validation (k=N)
Cross-Validation

- random subsampling
- **k-fold cross-validation**
- leave-1-out cross-validation ( k=N )
Cross-Validation

- random subsampling
- k-fold cross-validation
- leave-1-out cross-validation (k=N)
Cross-Validation

• how many folds do we need?
• with larger k
  • error estimation tends to be more accurate
  • but computational time will be larger
• in practice, larger dataset, smaller k
• a common choice for k-fold cross-validation is k = 10
Some Issues with Cross-Validation

• intensive use of cross-validation can overfit if you explore too many models, by tuning hyper parameters to predict the whole training set well
  • hold out an additional test set before doing any model selection. Check the best model performs well even on the additional test set
• time consuming (always if done naively)
  • there are efficient tricks that can save work over brute force
$k$-Nearest Neighbors

- $k$-NN is a simple algorithm which stores all available training examples and predict value/class of an unseen instance based on a similarity measure
  - $k = 1$
    - predict the same value/class as the nearest instance in the training set
  - $k > 1$
    - find the $k$ closet training examples
    - predict class: majority vote
    - predict value: average weighted by inverse distance
- memory based, no explicit training or model
**k-NN Classification**

- Similarity measure: Euclidean distance, etc.
  - Assumption behind Euclidean distance: uncorrelated inputs with equal variances
- Predict class: majority vote
- $k$ preferably odd to avoid ties for binary classification
- Choice of $k$
  - Smaller $k$: higher variance (less stable)
  - Larger $k$: higher bias (less precise)
- Cross-validation can help
- MATLAB demo