Linear Models for Classification

Greg Mori - CMPT 419/726

Bishop PRML Ch. 4

Classification: Hand-written Digit Recognition

\[ x_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad t_0 = (0, 0, 0, 0, 0) \]

- Each input vector classified into one of \( K \) discrete classes
- Denote classes by \( C_k \)
- Represent input image as a vector \( x_i \in \mathbb{R}^{784} \).
- We have target vector \( t_i \in \{0, 1\}^{10} \)
- Given a training set \( \{(x_1, t_1), \ldots, (x_N, t_N)\} \), learning problem is to construct a “good” function \( y(x) \) from these.

\[ y : \mathbb{R}^{784} \rightarrow \mathbb{R}^{10} \]

Generalized Linear Models

- Similar to previous chapter on linear models for regression, we will use a “linear” model for classification:

\[ y(x) = f(w^T x + w_0) \]

- This is called a generalized linear model
- \( f(\cdot) \) is a fixed non-linear function
  - e.g.

\[ f(u) = \begin{cases} 
1 \text{ if } u \geq 0 \\
0 \text{ otherwise}
\end{cases} \]

- Decision boundary between classes will be linear function of \( x \)
- Can also apply non-linearity to \( x \), as in \( \phi(x) \) for regression
### Discriminant Functions with Two Classes

- Start with 2 class problem, \( t_i \in \{0, 1\} \)
- Simple linear discriminant
  \[ y(x) = w^T x + w_0 \]
- Apply threshold function to get classification
- Projection of \( x \) in \( w \) dir. is \( \frac{w^T x}{\|w\|} \)

### Multiple Classes

- A linear discriminant between two classes separates with a hyperplane
- How to use this for multiple classes?
  - One-versus-the-rest method: build \( K - 1 \) classifiers, between \( C_k \) and all others
  - One-versus-one method: build \( \frac{K(K-1)}{2} \) classifiers, between all pairs

### Least Squares for Classification

- How do we learn the decision boundaries \((w_k, w_{0k})\)?
- One approach is to use least squares, similar to regression
- Find \( W \) to minimize squared error over all examples and all components of the label vector:
  \[ E(W) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k(x_n) - t_{nk})^2 \]
- Some algebra, we get a solution using the pseudo-inverse as in regression
• **Discriminant Functions**

• **Generative Models**

• **Discriminative Models**

---

### Problems with Least Squares

- Looks okay... least squares decision boundary
- Similar to logistic regression decision boundary (more later)
- Gets worse by adding easy points?!
- Why?
  - If target value is 1, points far from boundary will have high value, say 10; this is a large error so the boundary is moved

### More Least Squares Problems

- Easily separated by hyperplanes, but not found using least squares!
- We’ll address these problems later with better models

---

### Perceptrons

- **Perceptrons** is used to refer to many neural network structures (more next week)
- The classic type is a fixed non-linear transformation of input, one layer of adaptive weights, and a threshold:
  \[ y(x) = f(w^T \phi(x)) \]
  - Developed by Rosenblatt in the 50s
  - The main difference compared to the methods we’ve seen so far is the learning algorithm

### Perceptron Learning

- Two class problem
- For ease of notation, we will use \( t = 1 \) for class \( C_1 \) and \( t = -1 \) for class \( C_2 \)
- We saw that squared error was problematic
- Instead, we’d like to minimize the number of misclassified examples
  - An example is mis-classified if \( w^T \phi(x_i) t_i < 0 \)
  - Perceptron criterion:
    \[ E_P(w) = - \sum_{x_i,t_i \in M} w^T \phi(x_i) t_i \]
    sum over mis-classified examples only
**Perceptron Learning Algorithm**

- Minimize the error function using stochastic gradient descent (gradient descent per example):
  \[ w(t+1) = w(t) - \eta \nabla E_p(w) = w(t) + \eta \phi(x_n) t_n \text{ if incorrect} \]

- Iterate over all training examples, only change \( w \) if the example is mis-classified
- Guaranteed to converge if data are linearly separable
- Will not converge if not
- May take many iterations
- Sensitive to initialization

**Limitations of Perceptrons**

- Perceptrons can only solve linearly separable problems in feature space
  - Same as the other models in this chapter
- Canonical example of non-separable problem is X-OR
  - Real datasets can look like this too

- Note there are many hyperplanes with 0 error
  - Support vector machines have a nice way of choosing one

**Probabilistic Generative Models**

- Up to now we’ve looked at learning classification by choosing parameters to minimize an error function
- We’ll now develop a probabilistic approach
- With 2 classes, \( C_1 \) and \( C_2 \):
  \[ p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x)} \text{ Bayes’ Rule} \]
  \[ p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1) + p(x|C_2)} \text{ Sum rule} \]
  \[ p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)} \text{ Product rule} \]

- In generative models we specify the distribution \( p(x|C_k) \) which generates the data for each class
Let’s say we observe $x$ which is the current temperature

- **Determine if we are in Vancouver ($C_1$) or Honolulu ($C_2$)**

  **Generative model:**
  
  $p(C_1 | x) = \frac{p(x | C_1)p(C_1)}{p(x | C_1)p(C_1) + p(x | C_2)p(C_2)}$

  - $p(x | C_1)$ is distribution over typical temperatures in Vancouver
    - e.g. $p(x | C_1) = \mathcal{N}(x; 10, 5)$
  - $p(x | C_2)$ is distribution over typical temperatures in Honolulu
    - e.g. $p(x | C_2) = \mathcal{N}(x; 25, 5)$
  - Class priors $p(C_1) = 0.1, p(C_2) = 0.9$
  - $p(C_1 | x = 15) = \frac{0.0084 \cdot 0.1}{0.0084 \cdot 0.1 + 0.0105 \cdot 0.9} \approx 0.33$

  - **Logistic Sigmoid**
    - The function $\sigma(a) = \frac{1}{1 + \exp(-a)}$ is known as the logistic sigmoid
    - It squashes the real axis down to $[0, 1]$
    - It is continuous and differentiable
    - It avoids the problems encountered with the too correct least-squares error fitting (later)

  - **Multi-class Extension**
    - There is a generalization of the logistic sigmoid to $K > 2$ classes:
      
      $p(C_k | x) = \frac{p(x | C_k)p(C_k)}{\sum_j p(x | C_j)p(C_j)}$

      where $a_k = \ln p(x | C_k)p(C_k)$

      - a. k. a. **softmax function**
        - If some $a_k \gg a_j$, $p(C_k | x)$ goes to 1
Discriminant Functions
Generative Models
Discriminative Models

Gaussian Class-Conditional Densities

- Back to that $a$ in the logistic sigmoid for 2 classes
- Let’s assume the class-conditional densities $p(x|C_k)$ are Gaussians, and have the same covariance matrix $\Sigma$:
  \[
p(x|C_k) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2} (x - \mu_k)^T \Sigma^{-1} (x - \mu_k) \right\}
\]
- $a$ takes a simple form:
  \[
a = \ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)} = w^T x + w_0
\]
- Note that quadratic terms $x^T \Sigma^{-1} x$ cancel

Maximum Likelihood Learning

- We can fit the parameters to this model using maximum likelihood
  - Parameters are $\mu_1, \mu_2, \Sigma^{-1}, p(C_1) \equiv \pi, p(C_2) \equiv 1 - \pi$
  - Refer to as $\theta$
- For a datapoint $x_n$ from class $C_1$ ($t_n = 1$):
  \[
p(x_n, C_1) = p(C_1)p(x_n|C_1) = \pi N(x_n|\mu_1, \Sigma)
\]
- For a datapoint $x_n$ from class $C_2$ ($t_n = 0$):
  \[
p(x_n, C_2) = p(C_2)p(x_n|C_2) = (1 - \pi) N(x_n|\mu_2, \Sigma)
\]

Maximum Likelihood Learning - Class Priors

- The likelihood of the training data is:
  \[
p(t|\pi, \mu_1, \mu_2, \Sigma) = \prod_{n=1}^{N} \left[ \pi N(x_n|\mu_1, \Sigma)^{t_n} \left[ (1 - \pi) N(x_n|\mu_2, \Sigma) \right]^{1-t_n} \right]
\]
- As usual, $\ln$ is our friend:
  \[
\ell(t; \theta) = \sum_{n=1}^{N} t_n \ln \pi + (1 - t_n) \ln (1 - \pi) + t_n \ln N_1 + (1 - t_n) \ln N_2
\]
- Maximize for each separately

- Maximization with respect to the class priors parameter $\pi$ is straightforward:
  \[
\frac{\partial}{\partial \pi} \ell(t; \theta) = \sum_{n=1}^{N} \frac{t_n}{\pi} - \frac{1 - t_n}{1 - \pi}
\]
  \[
\Rightarrow \pi = \frac{N_1}{N_1 + N_2}
\]
- $N_1$ and $N_2$ are the number of training points in each class
- Prior is simply the fraction of points in each class
Maximum Likelihood Learning - Gaussian Parameters

- The other parameters can also be found in the same fashion
- Class means:
  \[
  \mu_1 = \frac{1}{N_1} \sum_{n=1}^{N_1} t_n x_n
  \]
  \[
  \mu_2 = \frac{1}{N_2} \sum_{n=1}^{N_2} (1 - t_n) x_n
  \]
- Means of training examples from each class
- Shared covariance matrix:
  \[
  \Sigma = \frac{N_1}{N} \frac{1}{N_1} \sum_{n \in C_1} (x_n - \mu_1)(x_n - \mu_1)^T + \frac{N_2}{N} \frac{1}{N_2} \sum_{n \in C_2} (x_n - \mu_2)(x_n - \mu_2)^T
  \]
- Weighted average of class covariances

Probabilistic Discriminative Models

- Generative model made assumptions about form of class-conditional distributions (e.g. Gaussian)
  - Resulted in logistic sigmoid of linear function of \( x \)
- Discriminative model - explicitly use functional form
  \[
  p(C_1|x) = \frac{1}{1 + \exp(-w^T x + w_0)}
  \]
  and find \( w \) directly
- For the generative model we had \( 2M + M(M + 1)/2 + 1 \) parameters
  - \( M \) is dimensionality of \( x \)
- Discriminative model will have \( M + 1 \) parameters

Generative vs. Discriminative

- Generative models
  - Can generate synthetic example data
  - Perhaps accurate classification is equivalent to accurate synthesis
    - e.g. vision and graphics
  - Tend to have more parameters
  - Require good model of class distributions
- Discriminative models
  - Only usable for classification
  - Don’t solve a harder problem than you need to
  - Tend to have fewer parameters
  - Require good model of decision boundary
### Maximum Likelihood Learning - Discriminative Model

- As usual we can use the maximum likelihood criterion for learning
  \[ p(f|w) = \prod_{n=1}^{N} \frac{1}{y_n} \prod_{i=1}^{y_n} \frac{1}{1-y_n} \quad \text{where} \ y_n = p(C_1 | x_n) \]

- Taking In and derivative gives:
  \[ \nabla \ell(w) = \sum_{n=1}^{N} (t_n - y_n) x_n \]

- This time no closed-form solution since \( y_n = \sigma(w^T x) \)
- Could use (stochastic) gradient descent
  - But there’s a better iterative technique

### Iterative Reweighted Least Squares

- **Iterative reweighted least squares** (IRLS) is a descent method
  - As in gradient descent, start with an initial guess, improve it
  - Gradient descent - take a step (how large?) in the gradient direction

- IRLS is a special case of a **Newton-Raphson** method
  - Approximate function using second-order Taylor expansion:
    \[ \hat{f}(w + v) = f(w) + \nabla f(w)^T (v - w) + \frac{1}{2} (v - w)^T \nabla^2 f(w) (v - w) \]

  - Closed-form solution to minimize this is straight-forward: quadratic, derivatives linear

  - In IRLS this second-order Taylor expansion ends up being a weighted least-squares problem, as in the regression case from last week
  - Hence the name IRLS

### Conclusion

- **Readings:** Ch. 4.1.1-4.1.4, 4.1.7, 4.2.1-4.2.2, 4.3.1-4.3.3
- **Generalized linear models** \( y(x) = f(w^T x + w_0) \)
- **Threshold/max function for** \( f(\cdot) \)
  - Minimize with least squares
  - Fisher criterion - class separation
  - Perceptron criterion - mis-classified examples

- **Probabilistic models:** logistic sigmoid / softmax for \( f(\cdot) \)
  - Generative model - assume class conditional densities in exponential family; obtain sigmoid
  - Discriminative model - directly model posterior using sigmoid (a. k. a. **logistic regression**, though classification)
  - Can learn either using maximum likelihood

- **All of these models are limited to linear decision boundaries in feature space**

---

### Newton-Raphson

- Figure from Boyd and Vandenberghe, *Convex Optimization*
- Excellent reference, free for download online