Local Models

Steven J Zeil

Old Dominion Univ.

Fall 2010
Local Models

1. Localizing
   - Competitive Learning
     - Online k-Means
     - Adaptive Resonance Theory
     - Self-Organizing Maps
   - Learning Vector Quantization
   - Radial-Basis Functions
     - Falling Between the Cracks
     - Rule-Based Knowledge

2. Learning after Localizing
   - Hybrid Learning
   - Competitive Basis Functions
   - Mixture of Experts (MoE)
Local Models

Piecewise approaches to regression.

- Divide input space into local regions and learn simple models on each region.
- Localization can be supervised or unsupervised.
- Learning is then supervised.
- Or can do both at once.
Localizing

1. Localizing
   - Competitive Learning
     - Online k-Means
     - Adaptive Resonance Theory
     - Self-Organizing Maps
   - Learning Vector Quantization
   - Radial-Basis Functions
     - Falling Between the Cracks
     - Rule-Based Knowledge

2. Learning after Localizing
   - Hybrid Learning
   - Competitive Basis Functions
   - Mixture of Experts (MoE)
Competitive Learning

- **Competitive** methods will assign $\vec{x}$ to one region and apply a function associated with that single region.
- **Cooperative** methods will apply a mixture of functions weighted according to which region $\vec{x}$ is most likely to belong.
Competitive Learning Techniques

1. Localizing
   - Competitive Learning
     - Online k-Means
     - Adaptive Resonance Theory
     - Self-Organizing Maps
   - Learning Vector Quantization
   - Radial-Basis Functions
     - Falling Between the Cracks
     - Rule-Based Knowledge

2. Learning after Localizing
   - Hybrid Learning
   - Competitive Basis Functions
   - Mixture of Experts (MoE)
Online k-Means

\[
E \left( \{ \vec{m}_i \}_{i=1}^k | \mathcal{X} \right) = \sum_t \sum_i b_t^i \| \vec{x}_t - \vec{m}_i \|
\]

\[
b_t^i = \begin{cases} 
1 & \text{if } \| \vec{x}_t - \vec{m}_i \| = \min_j \| \vec{x}_t - \vec{m}_j \| \\
0 & \text{otherwise}
\end{cases}
\]

- batch k-means: \( \vec{m}_i = \frac{\sum_t b_t^i \vec{x}_t}{\sum_t b_t^i} \)
- online k-means:

\[
\Delta m_{ij} = -\eta \frac{\partial E_t}{\partial m_{ij}} = \eta b_t^i (x_t^i - m_{ij})
\]
Winner-take-all Network

- Online k-means can be implemented via a variant of perceptrons
- Blue lines are *inhibitory* connections - seek to suppress other values
- Red are *excitatory* - attempt to reinforce own output
- With appropriate weights, these suppress all but the maximum
Adaptive Resonance Theory (ART)

- Incrementally adds new cluster means
- $\rho$ denotes *vigilance*
- If a new $\vec{x}$ lies outside the vigilance of all cluster centers, use that $\vec{x}$ as the center of a new cluster
Self-Organizing Maps (SOM)

- Units (cluster means) have a *neighborhood*
  - More often, 2D
- Update both the closest mean \( \vec{m}_i \) to \( \vec{x} \) but also the ones in \( \vec{m}_i \)'s neighborhood
  - Strength of the update falls off with steps through the neighborhood

\[
\Delta \vec{m}_j = \eta e(j, i)(\vec{x}^t - \vec{m}_i)
\]

\[
e(j, i) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left[ -\frac{(j - i)^2}{2\sigma^2} \right]
\]
Learning Vector Quantization (LVQ)

- Supervised technique
- Assume that existing cluster means are labeled with classes
- If $\vec{x}$ is closest to $\vec{m}_i$,

$$
\begin{align*}
\Delta \vec{m}_i &= \eta(\vec{x}^t - \vec{m}_i) \quad \text{if label}(\vec{x}^t) = \text{label}(\vec{m}_i) \\
\Delta \vec{m}_i &= -\eta(\vec{x}^t - \vec{m}_i) \quad \text{otherwise}
\end{align*}
$$
Radial-Basis Functions

- A weighted distance from a cluster mean

\[ p^t_h = \exp \left( -\frac{||\vec{x}^t - \vec{m}_h||^2}{2s^2_h} \right) \]

- \( s_h \) is the “spread” around \( \vec{m}_h \)
Radial Functions and Perceptrons

\[ p_h^t = \exp \left[ -\frac{||\vec{x}^t - \vec{m}_h||^2}{2s_h^2} \right] \]

\[ y^t = \sum_{h=1}^{H} w_hp_h^t + w_0 \]

Node that the \( p_h \) are taking the usual place of the \( x_i \)

\[ x_1 \quad x_j \quad x_d \]

\[ y_i \]

\[ w_{ih} \]

\[ m_{hj}, s_h \]

\[ p_0 = +1 \]
Using RBFs as a New Basis

Local representation in the space of \((p_1, p_2, p_3)\)
- \(x^a\) : (1.0, 0.0, 0.0)
- \(x^b\) : (0.0, 0.0, 1.0)
- \(x^c\) : (1.0, 1.0, 0.0)

Distributed representation in the space of \((h_1, h_2)\)
- \(x^a\) : (1.0, 1.0)
- \(x^b\) : (0.0, 1.0)
- \(x^c\) : (1.0, 0.0)
Obtaining RBFs

- Unsupervised:
  - Use any prior technique to compute the means (e.g., k-means)
  - Set the spread to cover the cluster
    - Find the $\bar{x}^t$ belonging to cluster $h$ but farthest from $\bar{m}_h$
    - Set $s_h$ so that $p^t_h \approx 0.5$

- Supervised: Because $p_h$ are differentiable, can combine with training of overall function
Falling Between the Cracks

- With RBFs it is possible for some $\vec{x}$ to fall outside region of influence of all clusters.
- May be useful to train an “overall” model and then train local exceptions

\[
y^t = \sum_{h=1}^{H} \mathbf{w}^h p_h^t + \vec{v}^T \vec{x}^t + v_0
\]

exceptions  \hspace{2cm} default rule
Rule with Exceptions
Normalized Basis Functions

- Alternatively, normalize the basis functions so that their sum is 1.0
- Do cooperative calculation
Prior rules often give localized solutions
E.g., IF \(((x_1 \approx a \text{ AND } x_2 \approx b)) \text{ OR } (x_3 \approx c)\) THEN y=0.1

\[ p_1 = \exp \left( -\frac{(x_1 - a)^2}{2s_1^2} \right) \exp \left( -\frac{(x_2 - b)^2}{2s_2^2} \right) \text{ with } w_1 = 0.1 \]

\[ p_2 = \exp \left( -\frac{(x_3 - c)^2}{2s_3^2} \right) \text{ with } w_2 = 0.1 \]
Localizing

Learning after Localizing

1. Localizing
   - Competitive Learning
     - Online k-Means
     - Adaptive Resonance Theory
     - Self-Organizing Maps
   - Learning Vector Quantization
   - Radial-Basis Functions
     - Falling Between the Cracks
     - Rule-Based Knowledge

2. Learning after Localizing
   - Hybrid Learning
   - Competitive Basis Functions
   - Mixture of Experts (MoE)
Hybrid Learning

- Use unsupervised techniques to learn centers (and spreads)
- Learn 2nd layer weight by supervised gradient-descent
Fully Supervised

Training both levels at once

\[
E (\{\vec{m}_h, s_h, w_{ih}\}_{i,h} | \mathcal{X}) = \frac{1}{2} \sum_t \sum_i (r_i^t - y_i^t)^2
\]

\[
y_i^t = \sum_{h=1}^{H} w_{ih} p_h^t + w_{i0}
\]

\[
\Delta w_{ih} = \eta \sum_t (r_i^t - y_i^t) p_h^t
\]

\[
\Delta m_{hj} = \eta \sum_t \left[ \sum_i (r_i^t - y_i^t) w_{ih} \right] p_h^t \frac{(x_j^t - m_{hj})}{s_h^2}
\]

\[
\Delta s_h = \eta \sum_t \left[ \sum_i (r_i^t - y_i^t) w_{ih} \right] p_h^t \frac{||\vec{x}^t - \vec{m}_h||^2}{s_h^3}
\]
Mixture of Experts

- In RBF, each local fit is a constant, $w_{ih}$.
- In MoE, each local fit is a linear function of $\vec{x}$, a “local expert”:
  \[ w_{ih}^t = \vec{v}_{ih}^T \vec{x}^t \]
- The $g_h$ form a gating network.
Gating

The gating network selects a mixture of models from the local experts \((w_h)\)

- **Radial gating**

  \[ g^t_h = \frac{\exp \left[ - \frac{||\vec{x}^t - \vec{m}_h||^2}{2s^2_h} \right]}{\sum_j \exp \left[ - \frac{||\vec{x}^t - \vec{m}_j||^2}{2s^2_j} \right]} \]

- **Softmax gating**

  \[ g^t_h = \frac{\exp \left[ \vec{m}_h^T \vec{x}^t \right]}{\sum_j \exp \left[ \vec{m}_j^T \vec{x}^t \right]} \]
Cooperative MoE

\[ E (\{\vec{m}_h, s_h, w_{ih}\}_{i,h} | \mathcal{X}) = \frac{1}{2} \sum_t \sum_i (r_i^t - y_i^t)^2 \]

\[ \Delta \vec{v}_{ih} = \eta \sum_t (r_i^t - y_i^t) g_h^t \vec{x}_t \]

\[ \Delta \vec{m}_{hj} = \eta \sum_t (r_i^t - y_i^t)(w_{ih}^t - y_i^t) g_h^t x_j^t \]
Cooperative & Competitive MoE

**Cooperative**

\[ \Delta \vec{v}_{ih} = \eta \sum_t (r_i^t - y_i^t) g_h^t \vec{x}_t^t \]

\[ \Delta \vec{m}_{hj} = \eta \sum_t (r_i^t - y_i^t)(w_{ih}^t - y_i^t) g_h^t \vec{x}_j^t \]

**Competitive**

\[ \Delta \vec{v}_{ih} = \eta \sum_t (r_i^t - y_i^t) f_h^t \vec{x}_t^t \]

\[ \Delta \vec{m}_h = \eta \sum_t (f_i^t - g_i^t) \vec{x}_t^t \]

\( f_h \) is the posterior prob. of unit \( h \) taking both the input and output into account.
Cooperative vs. Competitive

- Cooperative is generally more accurate.
  - Models overlap, giving smoother fit
- Competitive generally learns faster.
  - Generally only one expert at a time is active