CS475/CS675
Lecture 22: July 14, 2016

Application: regression in machine learning

Machine Learning

- Arthur Samuel (1959): Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed.

- Tom Mitchell (1998): A computer program is said to *learn* from **experience E** with respect to some class of **tasks T** and performance **measure P**, if its performance at tasks in T, as measured by P, improves with experience E.

Supervised Learning

- Inductive learning:
  - Given a **training set** of **examples** of the form $(x, f(x))$
    - x is the input, $f(x)$ is the output
  - Return a function h that **approximates** f
    - h is called the hypothesis

Linear model for regression

- Simplest form of regression
- Picture:

Problem

- Data: $\{(\boldsymbol{x_1}, t_1), (\boldsymbol{x_2}, t_2), \ldots, (\boldsymbol{x_N}, t_N)\}$
  - $\boldsymbol{x} = < x_1, x_2, \ldots, x_D >$: input vector
  - t: target (continuous value)
- Problem: find hypothesis h that maps $\boldsymbol{x}$ to t
  - Assume that h is linear:

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D = \boldsymbol{w^T} \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix}$$

- Objective: minimize some loss function
  - Euclidean loss: $L_2(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(\boldsymbol{x_n}, \boldsymbol{w}) - t_n)^2$

Optimization

- Find best w that minimizes Euclidean loss

$$\boldsymbol{w}^* = argmin_{\boldsymbol{w}} \frac{1}{2} \sum_{n=1}^{N} \left(t_n - \boldsymbol{w}^{\boldsymbol{T}} \begin{pmatrix} 1 \\ \boldsymbol{x_n} \end{pmatrix} \right)^2$$

- Convex optimization problem

  $\Longrightarrow$ unique optimum (global)

Solution

- Let $\overline{\boldsymbol{x}} = \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix}$ then $\min_{\boldsymbol{w}} \frac{1}{2} \sum_{n=1}^{N} \left(t_n - \boldsymbol{w}^{\boldsymbol{T}} \overline{\boldsymbol{x}}_{\boldsymbol{n}} \right)^2$
- Find $\boldsymbol{w}^*$ by setting the derivative to 0

$$\frac{\partial L_2}{\partial_{w_j}} = \sum_{n=1}^{N} \left(t_n - \boldsymbol{w}^{\boldsymbol{T}} \overline{\boldsymbol{x}}_{\boldsymbol{n}} \right) \bar{x}_{nj} = 0 \quad \forall j$$

$$\Longrightarrow \sum_{n=1}^{N} \left(t_n - \boldsymbol{w}^{\boldsymbol{T}} \overline{\boldsymbol{x}}_{\boldsymbol{n}} \right) \overline{\boldsymbol{x}}_{\boldsymbol{n}} = 0$$

- This is a linear system in $\boldsymbol{w}$, therefore we rewrite it as $\boldsymbol{A}\boldsymbol{w} = \boldsymbol{b}$

  where $\boldsymbol{A} = \sum_{n=1}^{N} \overline{\boldsymbol{x}}_{\boldsymbol{n}} \overline{\boldsymbol{x}}_{\boldsymbol{n}}^{\boldsymbol{T}}$ and $\boldsymbol{b} = \sum_{n=1}^{N} t_n \overline{\boldsymbol{x}}_{\boldsymbol{n}}$

Solution

- If training instances span $\Re^{D+1}$ then $\boldsymbol{A}$ is invertible:

$$\boldsymbol{w} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

- In practice it is faster to solve the linear system $\boldsymbol{A}\boldsymbol{w} = \boldsymbol{b}$ directly instead of inverting $\boldsymbol{A}$
  - Gaussian elimination
  - Conjugate gradient
  - Iterative methods

Generalized Linear Models

- How can we do non-linear regression with linear algebra?

- Idea: map inputs to a different space and do linear regression in that space

Example

- Suppose the underlying function is quadratic

Basis functions

- Use non-linear basis functions:
  - Let ϕ_i denote a basis function

$$\phi_0(x) = 1$$
$$\phi_1(x) = x$$
$$\phi_2(x) = x^2$$

  - Let the hypothesis space H be

$$H = \{x \to w_0\phi_0(x) + w_1\phi_1(x) + w_2\phi_2(x) | w_i \in \Re\}$$

- If the basis functions are non-linear in x, then a non-linear hypothesis can still be found by linear regression

Common basis functions

- Polynomial: $\phi_j(x) = x^j$
- Gaussian: $\phi_j(x) = e^{-\frac{\left(x-\mu_j\right)^2}{2s^2}}$
- Sigmoid: $\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$

  where $\sigma(a) = \frac{1}{1+\mathrm{e}^{-\mathrm{a}}}$
- Also Fourier basis functions, wavelets, etc.

Non-linear regression

- Same as linear regression, but replace $\overline{\boldsymbol{x}}$ by $\phi(\boldsymbol{x})$
- Problem:

$$\boldsymbol{w}^* = argmin_{\boldsymbol{w}} \frac{1}{2} \sum_{n=1}^{N} \left(t_n - \boldsymbol{w}^{\boldsymbol{T}} \phi(\boldsymbol{x})\right)^2$$

- Solution: $\boldsymbol{Aw} = \boldsymbol{b}$

  where $\boldsymbol{A} = \sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x_n}) \boldsymbol{\phi}(\boldsymbol{x_n})^{\boldsymbol{T}}$ and $\boldsymbol{b} = \sum_{n=1}^{N} t_n \boldsymbol{\phi}(\boldsymbol{x_n})$

Case Study: AIBO Gait Optimization

Gait Optimization

- Problem: find best parameter setting of the gait controller to maximize walking speed
  - Why?: Fast robots have a better chance of winning in robotic soccer

- Solutions:
  - Stochastic hill climbing
  - **Gaussian Process Regression**
    - Lizotte, Wang, Bowling, Schuurmans (2007) Automatic Gait Optimization with Gaussian Processes, *International Joint Conferences on Artificial Intelligence (IJCAI)*.

Search Problem

- Let $\boldsymbol{x} \in \Re^{15}$, be a vector of 15 parameters that defines a controller for gait
- Let $f: \boldsymbol{x} \rightarrow \Re$ be a mapping from controller parameters to gait speed

- Problem: find parameters $\boldsymbol{x}^*$ that yield highest speed.

$$\boldsymbol{x}^* \leftarrow argmax_{\boldsymbol{x}} f(\boldsymbol{x})$$

But f is unknown…

Approach

- Picture

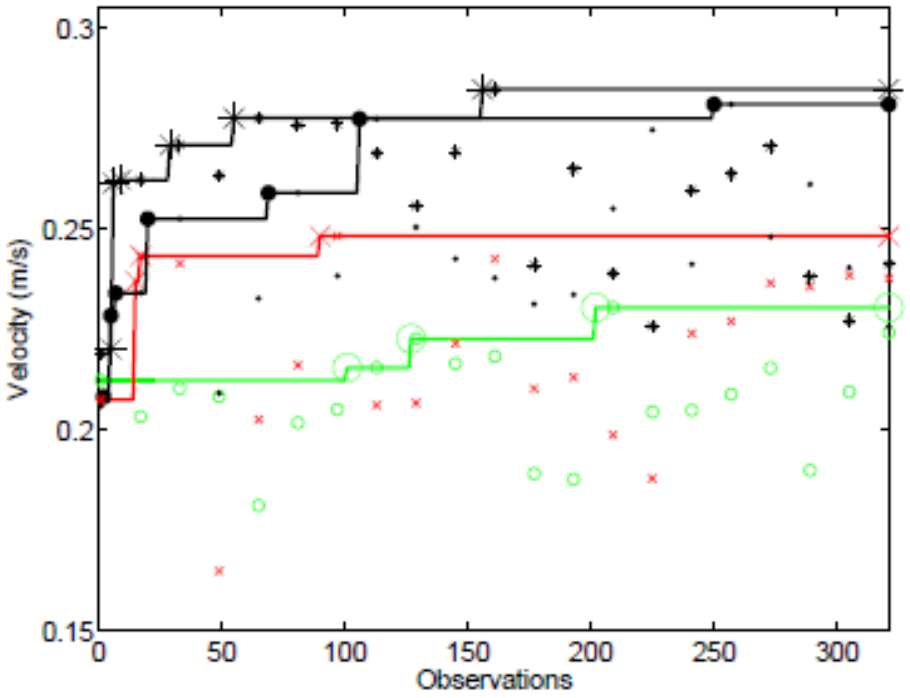
Approach

- Initialize $f(\cdot) \sim GP(m(\cdot), k(\cdot,\cdot))$
- Repeat:
  - Select new $\boldsymbol{x}$:

$$\boldsymbol{x}_{new} \leftarrow argmax_{\boldsymbol{x}} \frac{k(\boldsymbol{x},\boldsymbol{x})}{\max_{\boldsymbol{x}' \in X} f(\boldsymbol{x}') - m(\boldsymbol{x})}$$

  - Evaluate $f(\boldsymbol{x_{new}})$ by observing speed of robot with parameters set to $\boldsymbol{x}_{new}$
  - Update Gaussian process:
    - $\boldsymbol{X} \leftarrow \boldsymbol{X} \cup \{\boldsymbol{x_{new}}\}$ and $\boldsymbol{y} \leftarrow \boldsymbol{y} \cup f(\boldsymbol{x_{new}})$
    - $m(\cdot) \leftarrow k(\cdot, \boldsymbol{X})(\boldsymbol{K} + \sigma^{-2}\boldsymbol{I})^{-1}\boldsymbol{y}$
    - $k(\cdot,\cdot) \leftarrow k(\cdot,\cdot) - k(\cdot, \boldsymbol{X})(\boldsymbol{K} + \sigma^{2}\boldsymbol{I})^{-1}k(\boldsymbol{X},\cdot)$

Results



(•) GP w/MPI	(∗) GP w/MPI	(×) H.Clmb	(○) U.Rand
0.281 m/s	0.285 m/s	0.248 m/s	0.230 m/s
$\sigma_f^2 = 0.06$	$\sigma_f^2 = 0.6$		

Kernel (covariance function):

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^T S(\boldsymbol{x}-\boldsymbol{x}')}$$