23. Computer Vision and Smart Living

金力 Li Jin
li.jin@sjtu.edu.cn
上海交通大学密西根学院
Shanghai Jiao Tong University UM Joint Institute

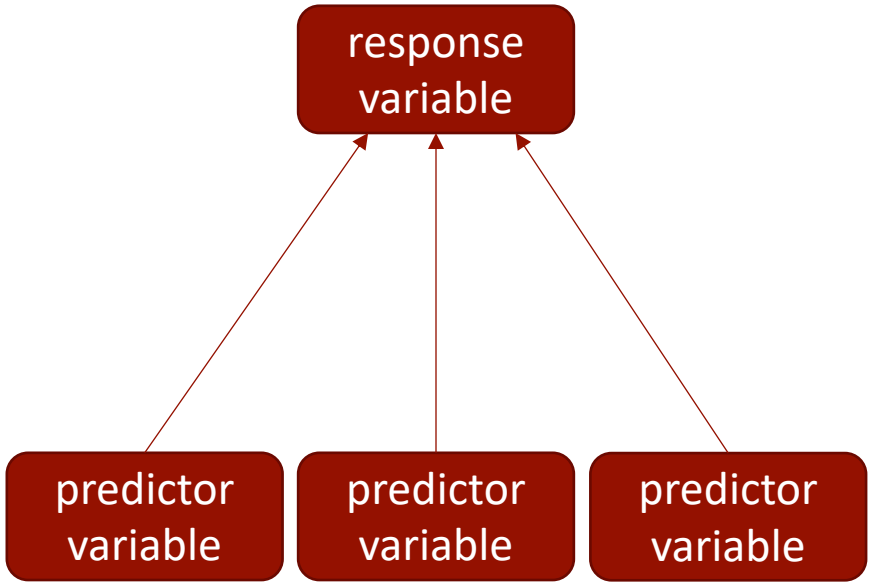
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Outline

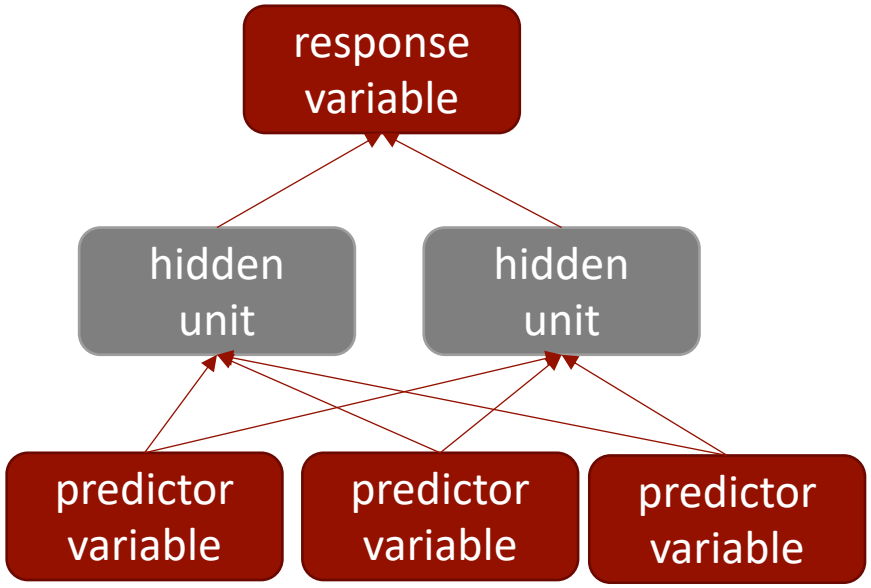
- Training NNs
- NN-based autonomous driving
- Smart living

Review

- A two-stage regression or classification model
- Instead of directly feeding predictor variables into a regression/classification function, we put a set of intermediate derived features or hidden units in between.



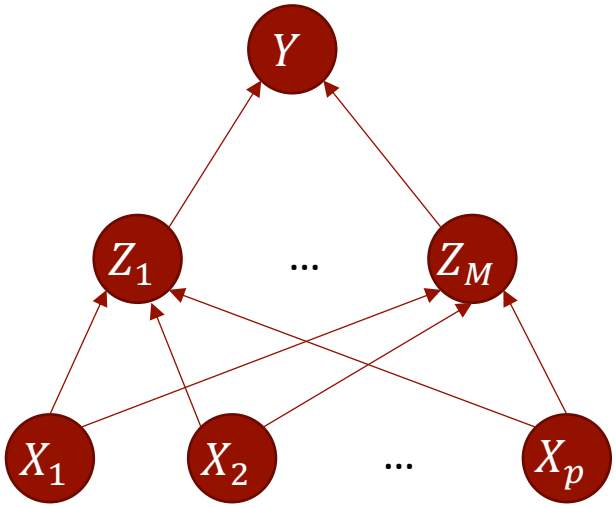
linear regression (LR)



neural network (NN)

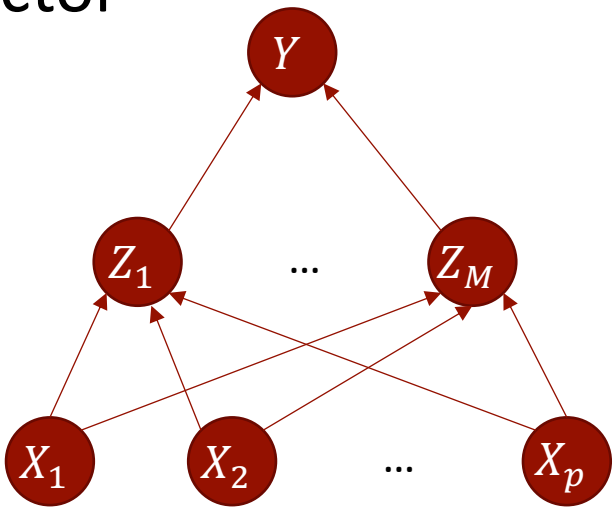
NN regression: 1-dimensional output

- Suppose we have a p-dimensional input vector $X = [X_1\ X_2\ \ldots X_p]^T$
- Our objective is to predict an output scalar Y from X
- Hidden units: an M-dimensional vector $Z = [Z_1\ Z_2\ \ldots\ Z_M]^T$
- Z is given by the sigmoid function:
- $Z_1 = \frac{1}{1+\exp(\alpha_{01}+\alpha_1^T X)}$
- $Z_2 = \frac{1}{1+\exp(\alpha_{02}+\alpha_2^T X)}$
- $Z_m = \frac{1}{1+\exp(\alpha_{0m}+\alpha_m^T X)}$
- $m = 1,2,\ldots,M$



NN regression: 1-dimensional output

- $Z_m = \frac{1}{1+\exp(\alpha_{0m}+\alpha_m^T X)}, \ m = 1,2,\dots,M$
- α_{0m} is a scalar
- α_m is a p-dimensional vector
- Output $Y = \beta_0 + \beta^T Z$
- β_0 is scalar, β is M-dimensional vector
- Thus, we have constructed a neural network.
- The NN is essentially a nonlinear regression.



Weights

- The neural network model has unknown parameters, often called weights, and we seek values for them that make the model fit the training data well.
- We denote the complete set of weights by θ , which consists of

$$\{\alpha_{0m}, \alpha_m;\ m = 1, 2, \ldots, M\}\ \ M(p+1) \text{ weights},$$
$$\{\beta_{0k}, \beta_k;\ k = 1, 2, \ldots, K\}\ \ K(M+1) \text{ weights}.$$

- Main task for training NN: determining weights!

Train a NN

- Input: vector of x
- Hidden units: vector of $z = g(x;\theta)$
- Output: scalar $y = f(z;\varphi) = f(g(x;\theta);\varphi)$
- RSS = $\sum\left(y_i - f(g(x_i;\theta);\varphi)\right)^2$ (for regression)
- MCE = $\sum \mathbb{I}_{\{y_i \neq f(g(x_i;\theta);\varphi)\}}$ (for classification)
- Alternatively, cross-entropy

$$\text{CE} = -y_{ik}\log f(g(x_i;\theta);\varphi)$$

- Find parameters θ and φ that minimize RSS or CE
- With the softmax activation function and the cross-entropy error function, the neural network model is exactly a linear logistic regression model in the hidden units, and all the parameters are estimated by maximum likelihood.

How to train NN

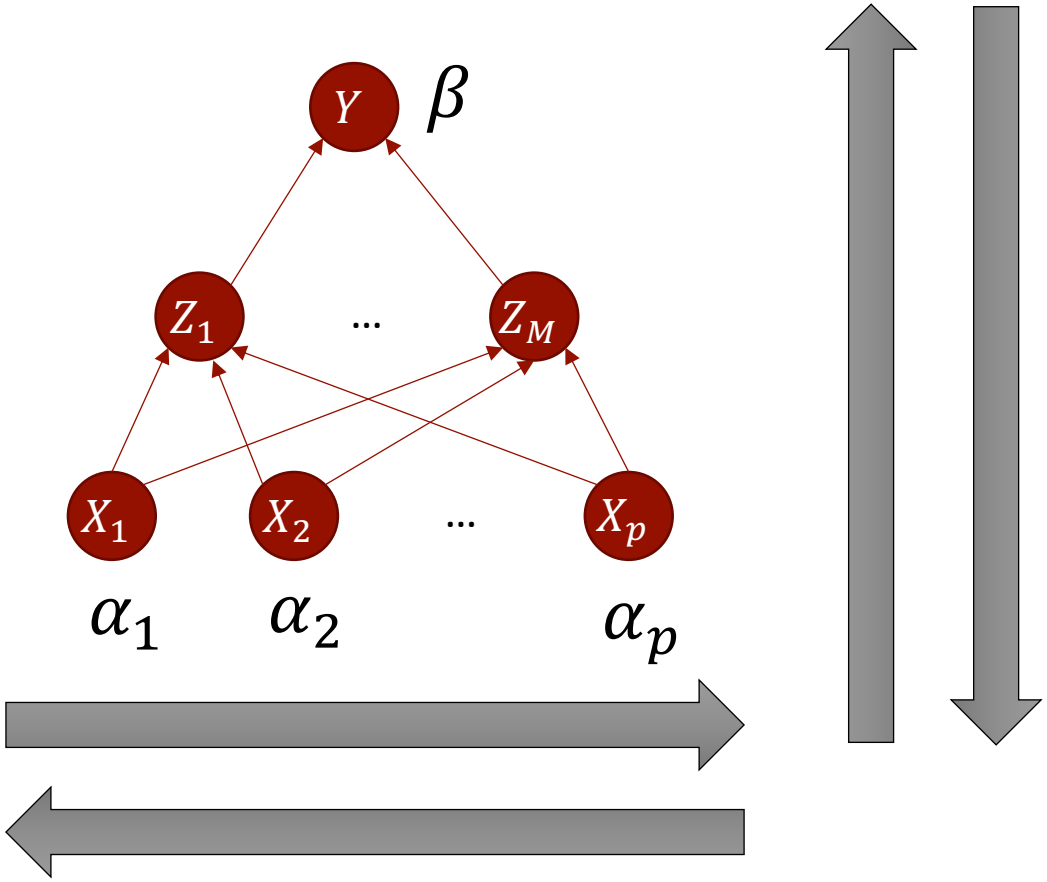
- But how to generate the hidden units themselves?
- Typically we don't want the global minimizer of $R(\theta)$, as this is likely to be an overfit solution.
- Instead some regularization is needed: this is achieved directly through a penalty term, or indirectly by early stopping.
- Penalty term: prediction error, probably plus complexity metric
- Early stopping: no need (and technically impossible) to exactly arrive at the true optimal; instead, can terminate when "close" to the true optimal

Gradient-descent approach

- Gradient-descent approach, also called back-propagation approach particularly for training NNs
- The generic approach to minimizing $R(\theta)$ is by gradient descent, called back-propagation in this setting.
- Because of the compositional form of the model, the gradient can be easily derived using the chain rule for differentiation.
- This can be computed by a forward and backward sweep over the network, keeping track only of quantities local to each unit.

Gradient-descent approach

- "Forward and backward sweep"



Gradient-descent approach

- Back-propagation in detail for squared error loss.
- Let

$$z_{mi} = \sigma(\alpha_{0m} + \alpha_m^T x_i)$$
$$z_i = (z_{1i}, z_{2i}, \ldots, z_{Mi})$$

- Short-handed notation $f(x) = f(g(x;\theta);\varphi)$
- Then, for K-dimensional response y, we have

$$R(\theta) \equiv \sum_{i=1}^{N} R_i$$
$$= \sum_{i=1}^{N}\sum_{k=1}^{K}(y_{ik} - f_k(x_i))^2,$$

Some optimization background

- Suppose we want to minimize a function $v = h(u)$
- v is scalar, u is vector
- Indeed, you can apply some optimality condition to find the optimal solution, say $\nabla_u h(u) = 0$
- However, that involves solving equations. For NN, we involves terms like e^u, which leads to transcendental equations.
- Recall: transcendental equations cannot be analytically solved.
- So, we need some numerical, approximate algorithms
- Answer: gradient-descent approach

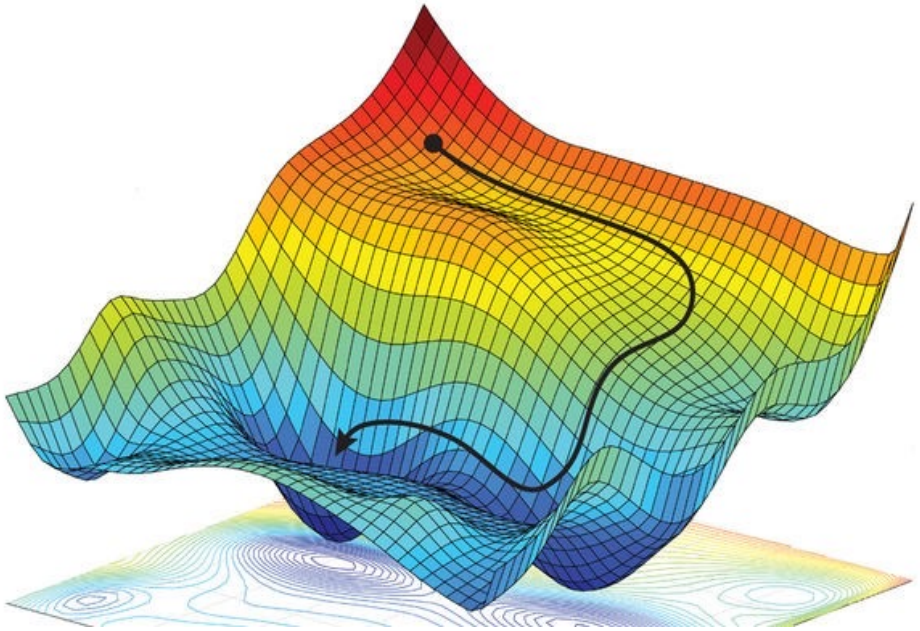
Some optimization background

- Instead of analytically solving the equations, we explore the set of solutions by a heuristic algorithm
- Heuristic: a practical method that is not guaranteed to be optimal, perfect, or rational, but is nevertheless sufficient for reaching an immediate, short-term goal or approximation.
- Objective: minimize $v = h(u)$
- First, we make an initial guess u_0
- Second, evaluate $D(u_0) = \nabla_u h(u)|_{u=u_0}$
- Third, update our solution by $u_1 = u_0 - \gamma D(u_0)$
- Then, keep iterating by $u_i = u_{i-1} - \gamma D(u_{i-1})$

Some optimization background

Terminate when some stopping rule is satisfied

- Rule 1: one-step improvement of objective function is less than some threshold
- Rule 2: magnitude of gradient is less than some threhsold



Gradient-descent approach for NN

- Objective function $R(\theta)$
- Gradient

$$\frac{\partial R_i}{\partial \beta_{km}} = -2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)z_{mi},$$

$$\frac{\partial R_i}{\partial \alpha_{m\ell}} = -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{i\ell}.$$

- Given these derivatives, a gradient descent update at the $(r + 1)$st iteration has the form

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}},$$

$$\alpha_{m\ell}^{(r+1)} = \alpha_{m\ell}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{m\ell}^{(r)}},$$

Passes

- The parameter updates can be implemented with a two-pass algorithm.
- In the forward pass, the current weights are fixed and the predicted values are computed.
- In the backward pass, the errors (essentially the magnitude of gradient) are computed.
- This two-pass procedure is what is known as back-propagation
- More details: Hastie 11.4.

Some issues in training NNs

- Starting values
- Overfitting
- Scaling of the inputs
- Number of hidden units and layers
- Multiple minima

Starting values

- $y = f(z;\varphi) = f(g(x;\theta);\varphi)$
- RSS = $\sum(y_i - f(g(x;\theta);\varphi))^2$
- Find parameters θ and φ that minimize RSS
- Usually starting values for parameters are chosen to be random values near zero.
- Note: small weights in sigmoid function -> almost linear model
- However, cannot be exactly zero; otherwise iteration will not move
- Starting instead with large weights often leads to poor solutions.

Overfitting

- Often neural networks have too many weights and will overfit the data at the global minimum of RSS.
- In early developments of neural networks, either by design or by accident, an early stopping rule was used to avoid overfitting.
- That is, we train the model only for a while, and stop well before we approach the global minimum.
- Since the weights start at a highly regularized (linear) solution, this has the effect of shrinking the final model toward a linear model.
- A validation dataset is useful for determining when to stop, since we expect the validation error to start increasing.

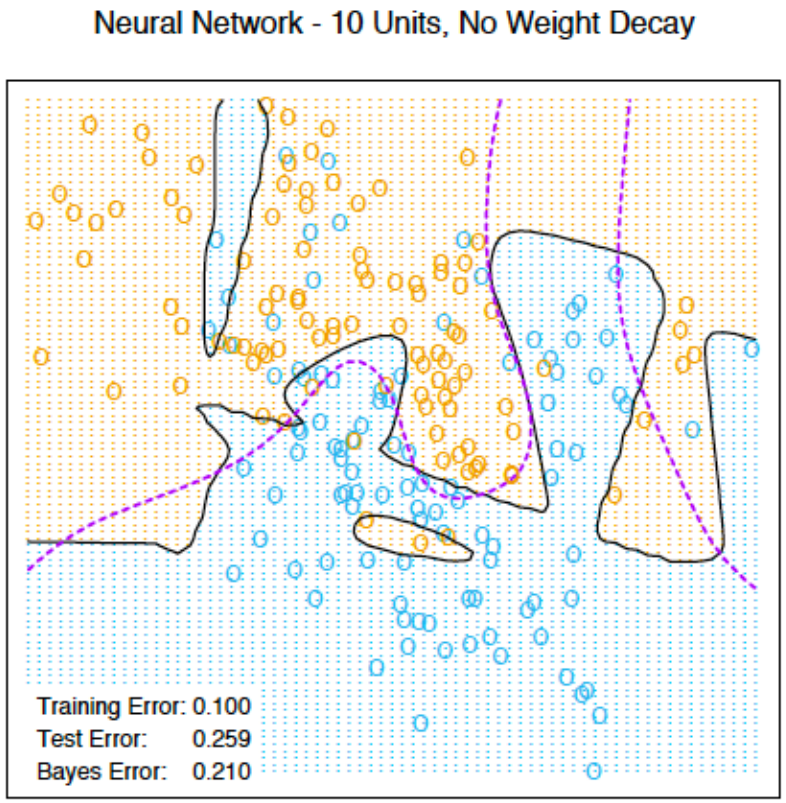
Weight decay

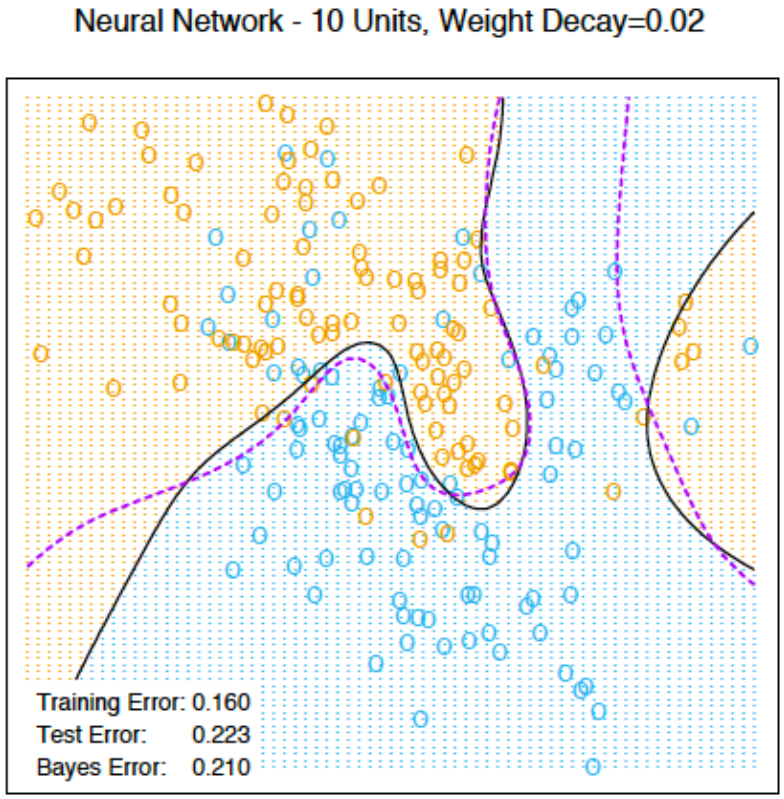
- One way of avoiding overfitting
- Analogous to ridge regression
- Recall ridge regression: $\min R(\beta) + \lambda \beta^T \beta$
- Weight decay for NN:

$$\min R(\alpha, \beta) + \lambda(\alpha^T \alpha + \beta^T \beta)$$

- λ is a tuning parameter
- Larger values of λ will tend to shrink the weights toward zero
- Typically cross-validation is used to estimate λ.

Weight decay





Scaling of the inputs

- Scaling of the inputs determines the effective scaling of the weights in the bottom layer
- Can have a large effect on the quality of the final solution.
- At the outset it is best to standardize all inputs to have mean zero and standard deviation one.
- This ensures all inputs are treated equally in the regularization process
- Also allows one to choose a meaningful range for the random starting weights
- With standardized inputs, it is typical to take random uniform weights over the range [– 0. 7, +0. 7].

Number of hidden units and layers

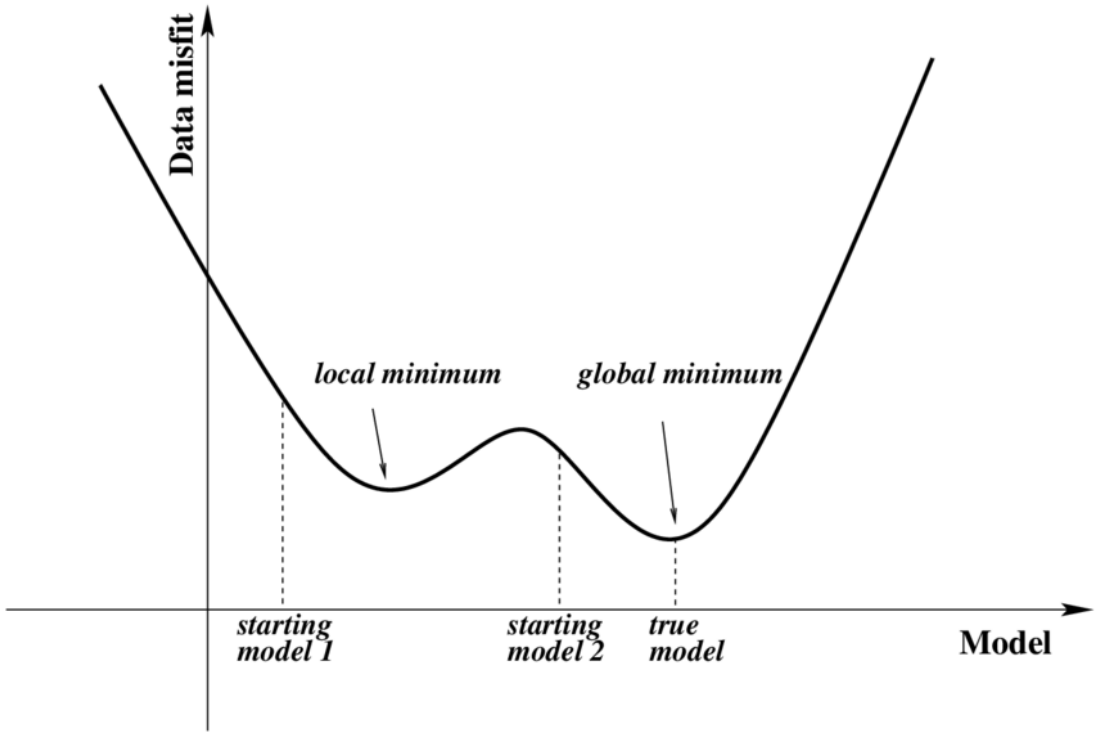
- Generally speaking it is better to have too many hidden units than too few.
- With too few hidden units, the model might not have enough flexibility to capture the nonlinearities in the data
- With too many hidden units, the extra weights can be shrunk toward zero if appropriate regularization is used.
- Typically the number of hidden units is somewhere in the range of 5 to 100
- This number increasing with the number of inputs and number of training cases.

Number of hidden units and layers

- Common practice: put down a reasonably large number of units and train them with regularization
- Some researchers use cross-validation to estimate the optimal number, but this seems unnecessary if cross-validation is used to estimate the regularization parameter
- Choice of the number of hidden layers is guided by background knowledge and experimentation
- Each layer extracts features of the input for regression or classification
- Use of multiple hidden layers allows construction of hierarchical features at different levels of resolution

Multiple Minima

- The error function $R(\theta)$ (i.e. the RSS) is nonconvex, possessing many local minima
- As a result, the final solution obtained is quite dependent on the choice of starting weights



Multiple Minima

- One must at least try a number of random starting configurations, and choose the solution giving lowest (penalized) error
- Probably a better approach is to use the average weights over the collection of networks as the final prediction
- Another approach is via bagging, which averages the predictions of networks training from randomly perturbed versions of the training data
- For non-convex problems, global optimality is hardly theoretically guaranteed.

Example: autonomous driving

Motivation

- Autonomous navigation has been a difficult problem for traditional vision and robotic techniques, primarily because of the noise and variability associated with real world scenes.
- Autonomous navigation systems based on traditional image processing and pattern recognition techniques often perform well under certain conditions but have problems with others.
- Part of the difficulty stems from the fact that the processing performed by these systems remains fixed across various driving situations.

Why NN

- Artificial neural networks have displayed promising performance and flexibility in other domains characterized by high degrees of noise and variability
- ALVINN (Autonomous Land Vehicle In a Neural Network) is a connectionist approach to the navigational task of road following.
- Specifically, ALVINN is an artificial neural network designed to control the NAVLAB, the Carnegie Mellon autonomous navigation test vehicle.

Outline

- Training NNs
- NN-based autonomous driving
- Smart living

Architecture

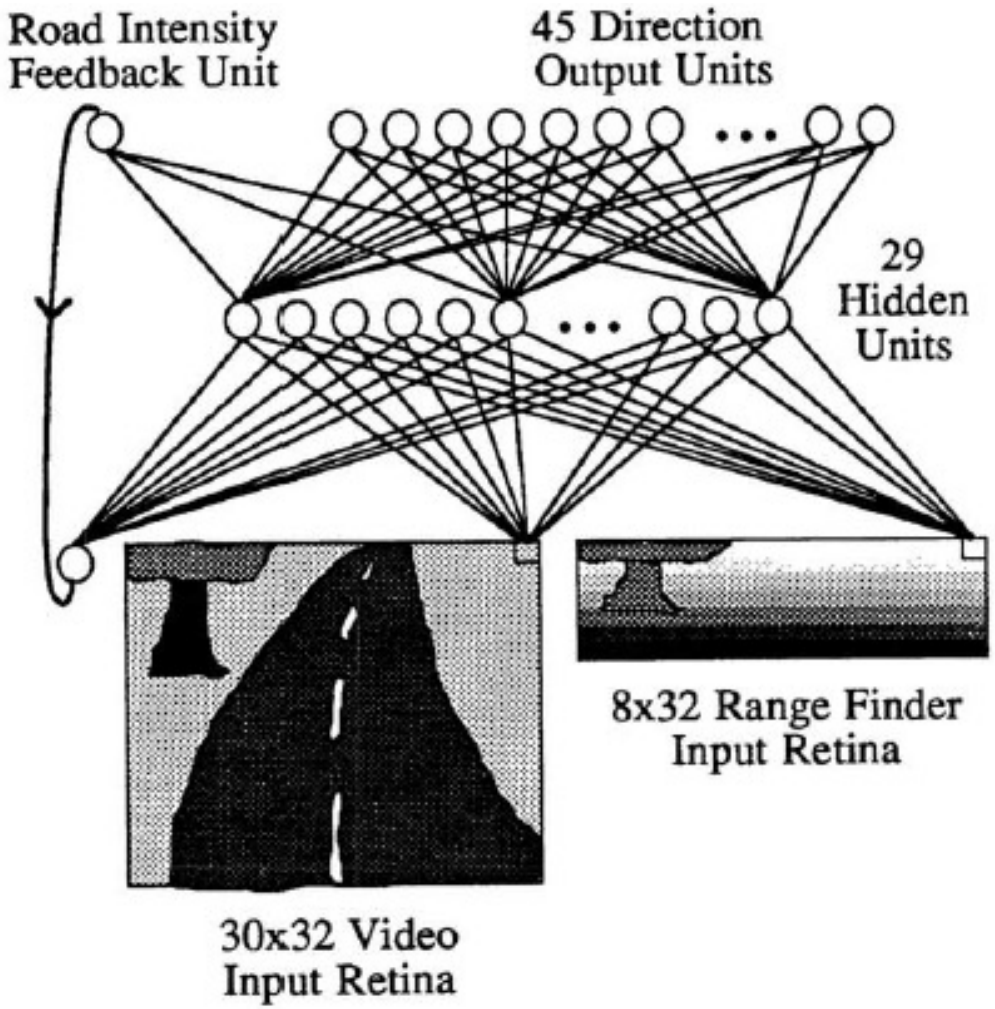


Figure 1: ALVINN Architecture

Input: video

- Input: sensory input available on the autonomous vehicle; video and range information.
- Camera sensor receives video camera input from a road scene.
- The activation level of each is proportional to the intensity in the blue color band of the corresponding patch of the image.
- The blue band of the color image is used because it provides the highest contrast between the road and the non-road

Input: range

- Also receives input from a laser range finder.
- The activation level of each unit is proportional to the proximity of the corresponding area in the image.
- The road intensity feedback unit indicates whether the road is lighter or darker than the non-road in the previous image.
- 1217 input units fully connected to the hidden layer of 29 units, which is in tum fully connected to the output layer

Output

- The output layer consists of 46 units, divided into two groups.
- The first set of 45 units is a linear representation of the turn curvature along which the vehicle should travel in order to head towards the road center.
- The middle unit represents the "travel straight ahead" condition while units to the left and right of the center represent successively sharper left and right turns.
- The final output unit is a road intensity feedback unit which indicates whether the road is lighter or darker than the non-road in the current image
- Parameters to determine: the activation levels in the hidden units

Test

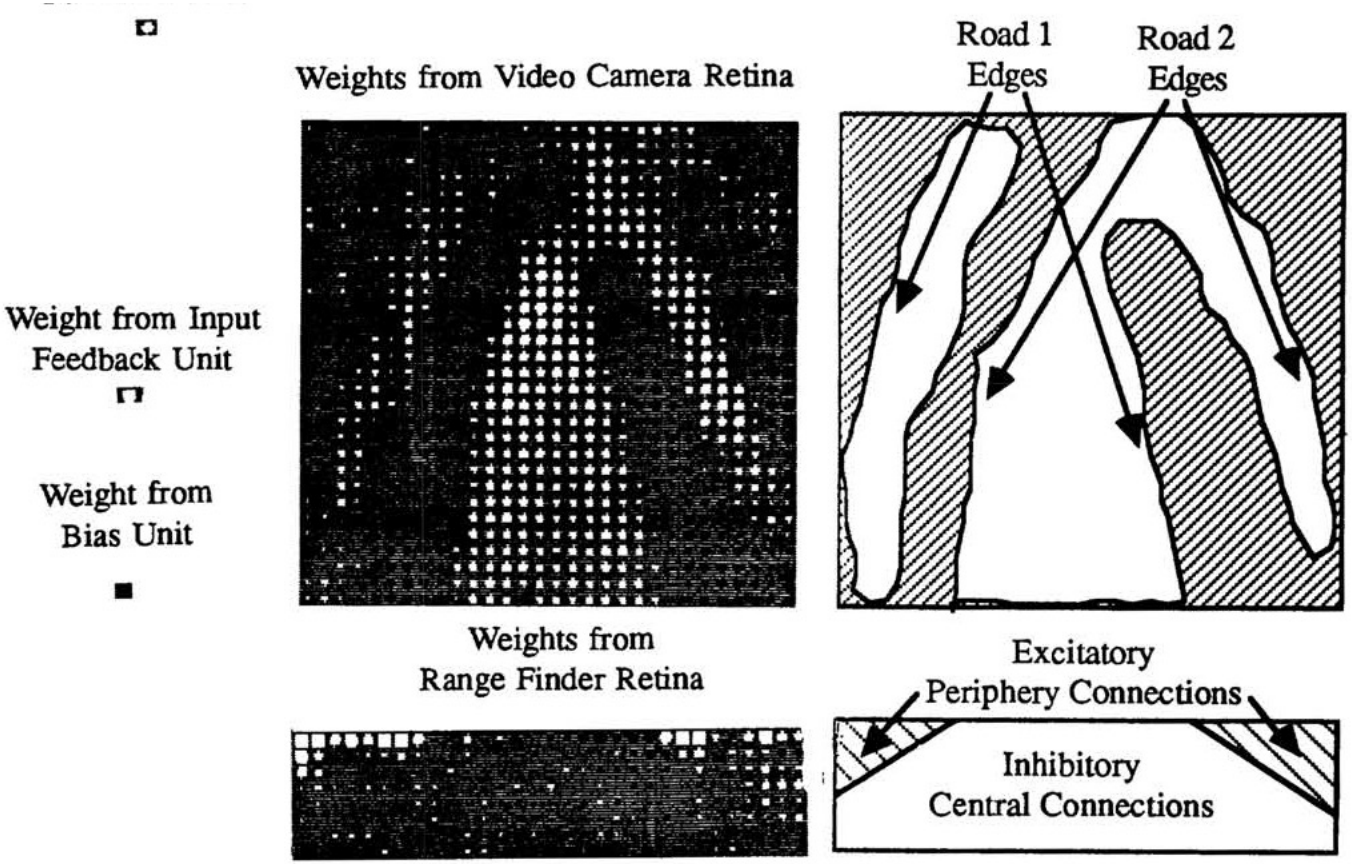


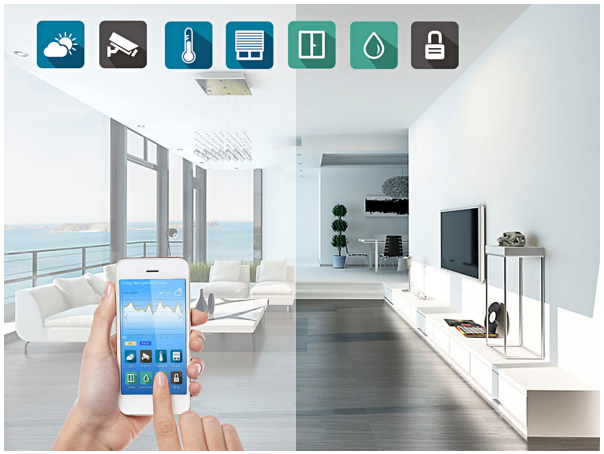
Figure 4: Diagram of weights projecting to and from a typical hidden unit in a network trained on roads with a fixed width. The schematics on the right are aids for interpretation.

Outline

- Training NNs
- NN-based autonomous driving
- Smart living

Background

- Nowadays, smartphones are becoming more powerful with reinforced processors, larger storage capabilities, richer entertainment functions and more communication methods.
- Bluetooth, which is mainly used for data exchange, add new features to smartphones.
- Bluetooth technology, created by telecom vendor Ericsson in 1994, shows its advantage by integrating with smartphones.



Bluetooth

- It has changed how people use digital devices at home or office, and has transferred traditional wired digital devices into wireless devices.
- A host Bluetooth device is capable of communicating with up to seven Bluetooth modules at the same time through one link.

Smart home

- Considering its normal working area of within eight meters, it is especially useful in a home environment.
- Thanks to Bluetooth technology and other similar techniques, the concept of Smart Living has offered better opportunity in convenience, comfort and security which
- With dramatic increase in smartphone users, smartphones have gradually turned into an all-purpose portable device and provided people for their daily use

Remote health

- Sensing technology and machine learning can be used for remote health
- The need for the development of such technologies is underscored by the aging of the population, the cost of formal health care, and the importance that individuals place on remaining independent in their own homes.

Activities of Daily Living

- To function independently at home, individuals need to be able to complete Activities of Daily Living (ADLs) such as eating, dressing, cooking, drinking, and taking medicine.
- Automating the recognition of activities is an important step toward monitoring the functional health of a smart home resident.

Activity recognition

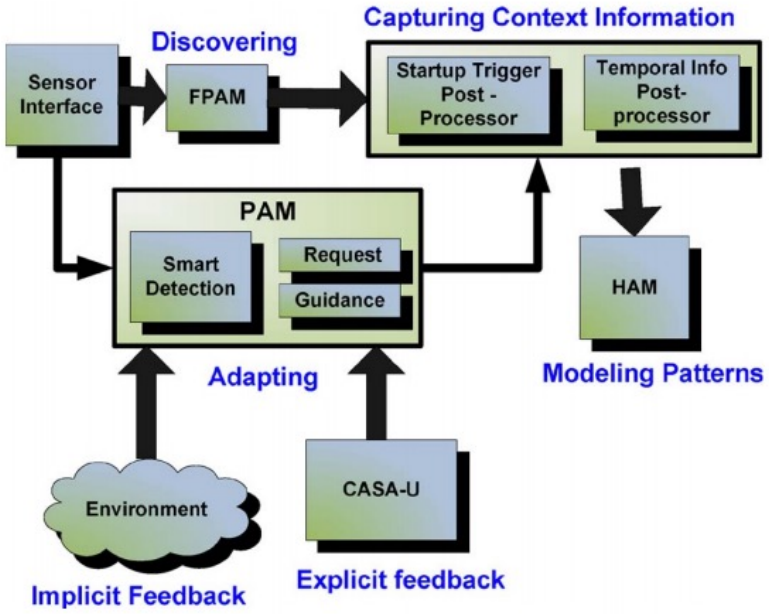
- When surveyed about assistive technologies, family caregivers of Alzheimer's patients ranked activity identification and tracking at the top of their list of needs
- In response to this recognized need, researchers have designed a variety of approaches to model and recognize activities.

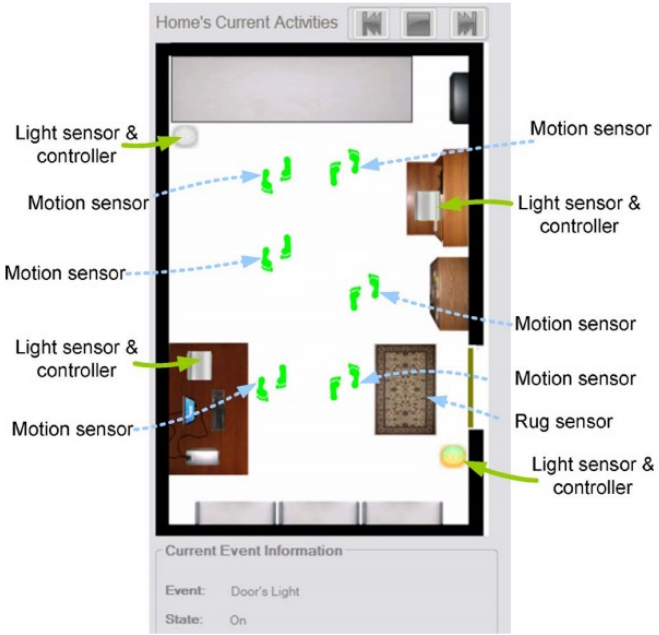
Mapping from activity to health

- The generally accepted approach is to model and recognize those activities that are frequently used to measure the functional health of an individual
- However, a number of difficulties arise with this approach...
  1. Irregular activities may be perfectly normal
  2. Irregular activities may also indicate health condition
  3. Lack of training data

Technology basis

- Center for Advanced Studies on Adaptive Systems (CASAS)
- Utilizes machine learning techniques to discover patterns in resident's daily activities and to generate automation polices that mimic these patterns





Methodology

- We introduce a state-of-the-art approach in the context of the CASAS Smart Home project by using sensor data that are collected in the CASAS smart apartment testbed.
- Unsupervised learning -> a more automated approach for activity recognition than is offered by previous approaches, which take a supervised approach and annotate the available data for training.
- Algorithmic basis: hidden Markov models (HMMs)
- Two major steps:
  1. Discover activities
  2. Recognize activities

Discover activities

- First step: how to identify the frequent and repeatable sequences of sensor events that comprise our smart environment's notion of an activity.
- How to do this? -> By applying frequent sequential pattern mining techniques, we can identify contiguous, consistent sensor event sequences that might indicate an activity of interest. (Recall smart meters)
- Core idea: frequent sequence mining technique called Varied-Order Sequential Miner (DVSM)
- DVSM can extract the pattern (a, b) from instances $\{b, x, c, a\}$, $\{a, u, b\}$ and $\{a, b, q\}$ despite the fact that the events are discontinuous and have varied orders.

Discovering Frequent Discontinuous Sequences

- DVSM first creates a reduced data set D_r containing the topmost frequent events.
- Next, DVSM slides a window of size 2 across D_r to find patterns of length 2.
- After this first iteration, the whole data set does not need to be scanned again.
- Instead, DVSM extends the patterns discovered in the previous iteration by their prefix and suffix events, and will match the extended pattern against the already discovered patterns (in the same iteration) to see if it is a variation of a previous pattern, or if it is a new pattern.

Similarity measure

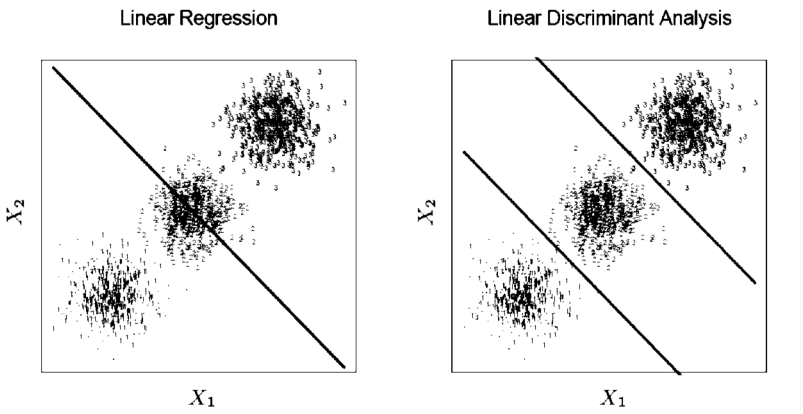
- To see if two patterns should be considered as variations of the same pattern, we use the Levenshtein (edit) distance to define a similarity measure $sim(A, B)$. between the two patterns.
- The edit distance, $e(A, B)$, is the number of edits (insertions, deletions, and substitutions) required to transform an event sequence A into another event sequence B.
- Similarity measure

$$sim(A, B) = 1 - \frac{e(A, B)}{\max\{|A|, |B|\}}$$

- Small $sim(A, B)$ -> variations of same pattern

Clustering Sequences into Groups of Activities

- We need to group the set of discovered patterns into a set of clusters
- The resulting set of clusters centroids represents the activities that we will model, recognize, and track
- Clustering: classify data points close to each other into one category
- Key: measure distance between data points



Clustering Sequences into Groups of Activities

- The patterns discovered by DVSM were composed of sensor events.
- In the clustering algorithm, the pattern is composed of states. (Recall “signatures” in smart meters)
- States correspond to the pattern’s events, but are enhanced to include additional information such as the type and duration of the sensor events.
- In addition, we can combine several states together to form a new state (we call it an extended state).
- For example, if a motion sensor in the kitchen is triggered several times in a row without another sensor event interrupting the sequence, the series of identical motion sensor events will be combined into one event with a longer duration.

Clustering Sequences into Groups of Activities

- How to measure distance between activities? -> Edit distance
- Compute the number of edit operations that are required to make activity X the same as activity Y.
- The edit operations include adding a step or deleting a step (traditional edit distance), reordering a step (order distance), or changing the attributes of a step (for this application, step attributes include the event duration and event frequencies).
- The general edit distance gives us a measure to compare activities and also to define cluster centroids.

Recognize activities

- Once the activities are discovered for a particular individual, we want to build a model that will recognize future executions of the activity.
- This will allow the smart environment to track each activity and determine if an individual's routine is being maintained.

Hidden Markov model

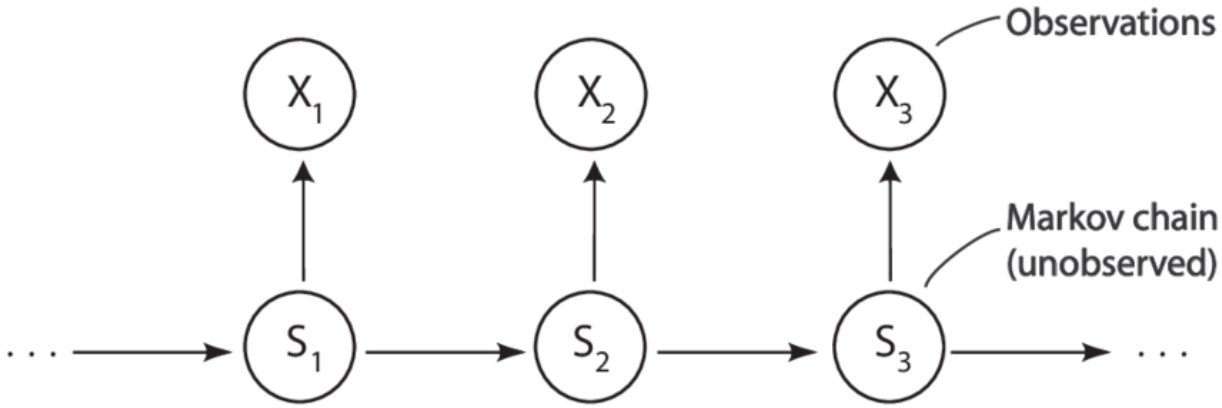
- We use a hidden Markov model (HMM) to recognize activities from sensor data as they are being performed.
- Each model is trained to recognize the patterns that correspond to the cluster representatives found in the previous step.
- A Markov Model is a statistical model of a dynamic system, which models the system using a finite set of states, each of which is associated with a multidimensional probability distribution over a set of parameters.
- The system is assumed to have a Markovian property, such that the current state depends on a finite history of previous states. (Recall queuing model)

Hidden Markov model

- A hidden Markov model is a statistical model in which the underlying data are generated by a stochastic process that is not observable.
- The process is assumed to be Markovian and can be observed through another set of stochastic processes that produce the sequence of observed features.
- HMMs traditionally perform well in the cases where temporal patterns need to be recognized
- This aligns with our requirement to recognize possibly interleaved activities.

Hidden Markov model

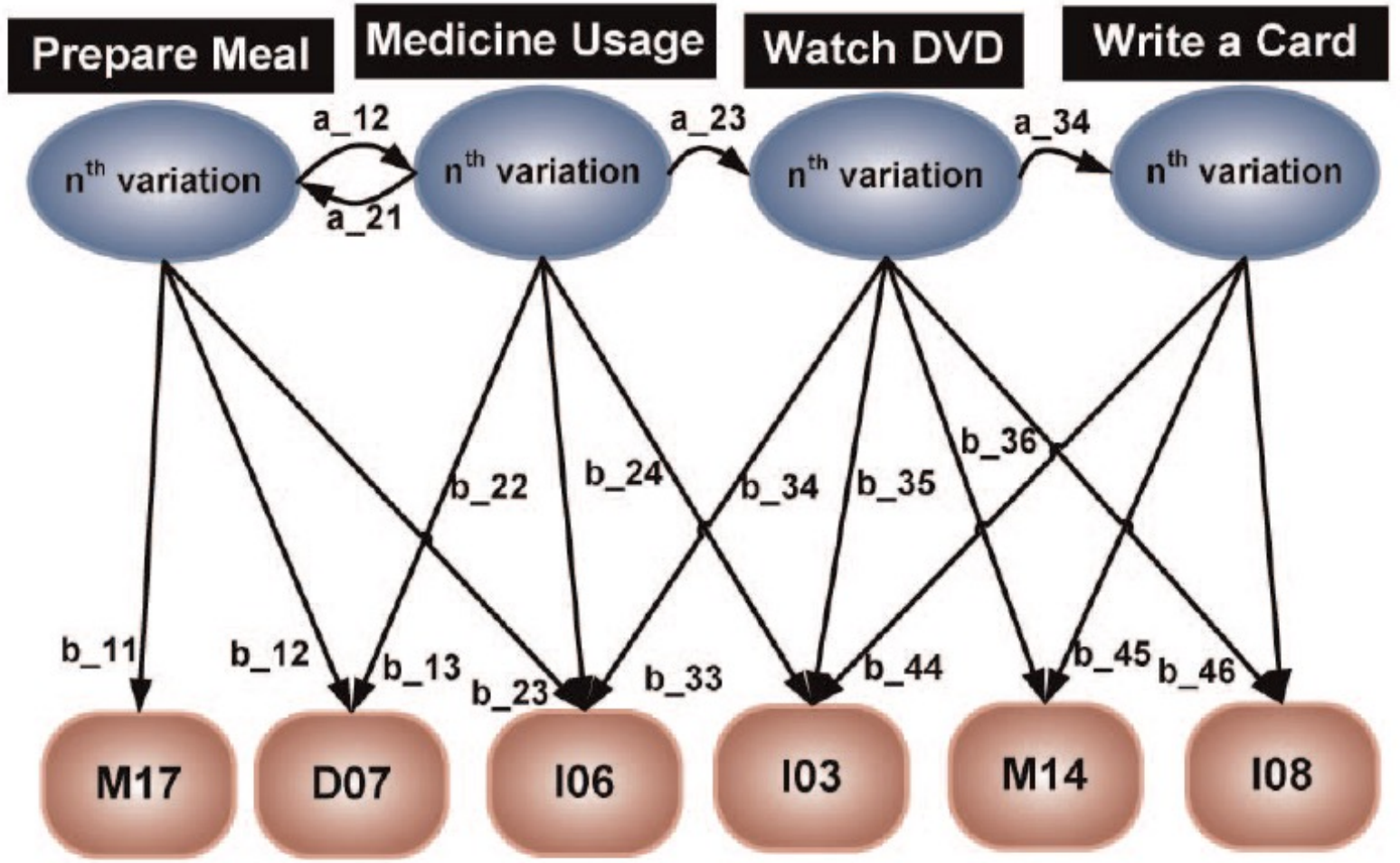
- As with a Markov chain, the conditional probability distribution of any hidden state depends only on the value of a finite number of preceding hidden states.
- The observable variable at time t, namely $x(t)$, depends only on the hidden variable $y(t)$ at that time slice.



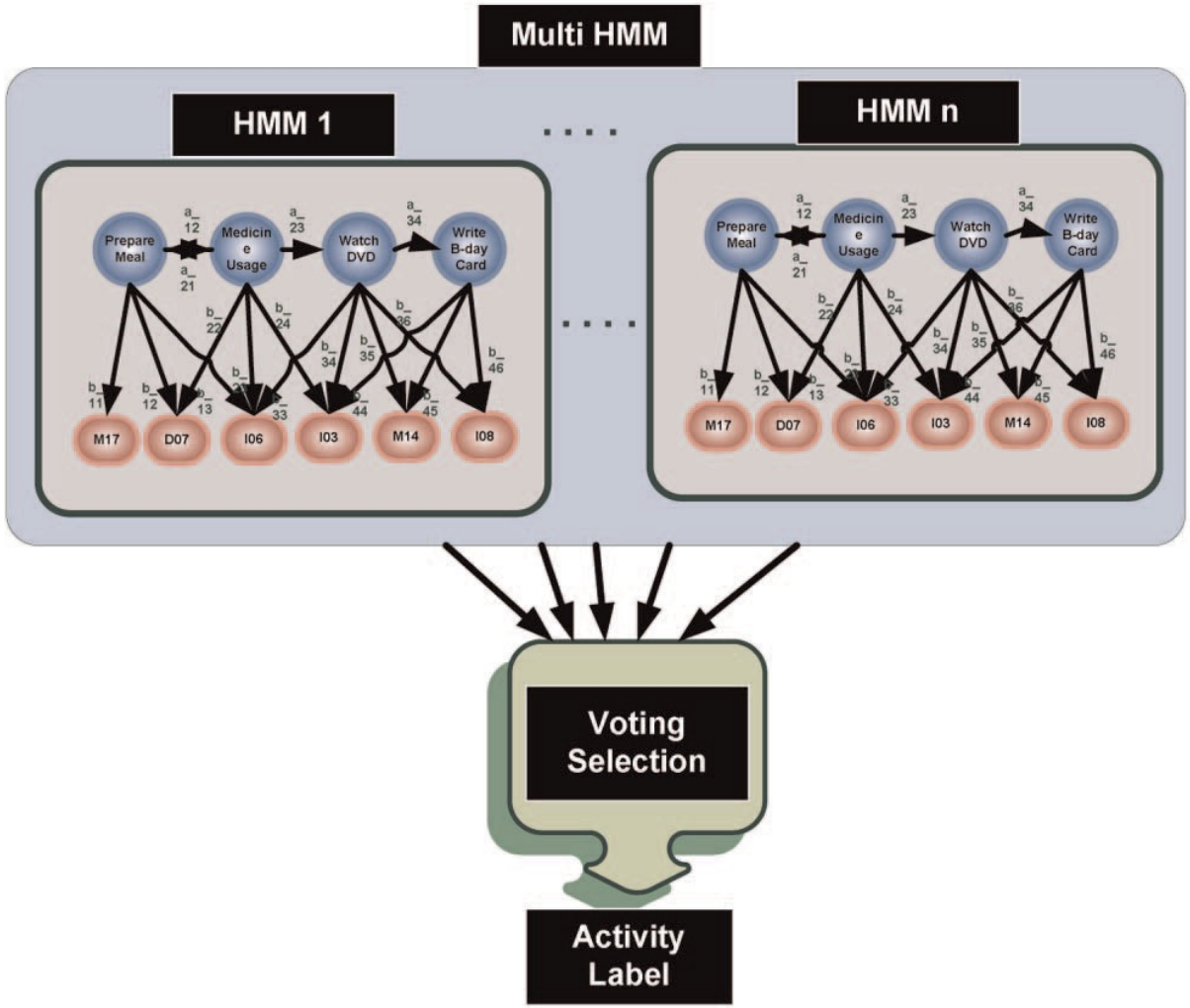
Hidden Markov model

- We can specify an HMM using three probability distributions:
  1. the distribution over initial states $\Pi = \{\pi_k\}$
  2. the state transition probability distribution $A = \{a_{kl}\}$, where
  $$a_{kl} = \Pr\{Y(t+1) = l | Y(t) = k\}$$
  is the probability of transitioning from state k to state l; and
  3. the observation distribution $B = \{b_{il}\}$, with
  $$b_{il} = \Pr\{X(t) = i | Y(t) = l\}$$
  indicating the probability that the state l would generate observation $X(t) = i$.
- The HMM can be trained using the maximum likelihood method

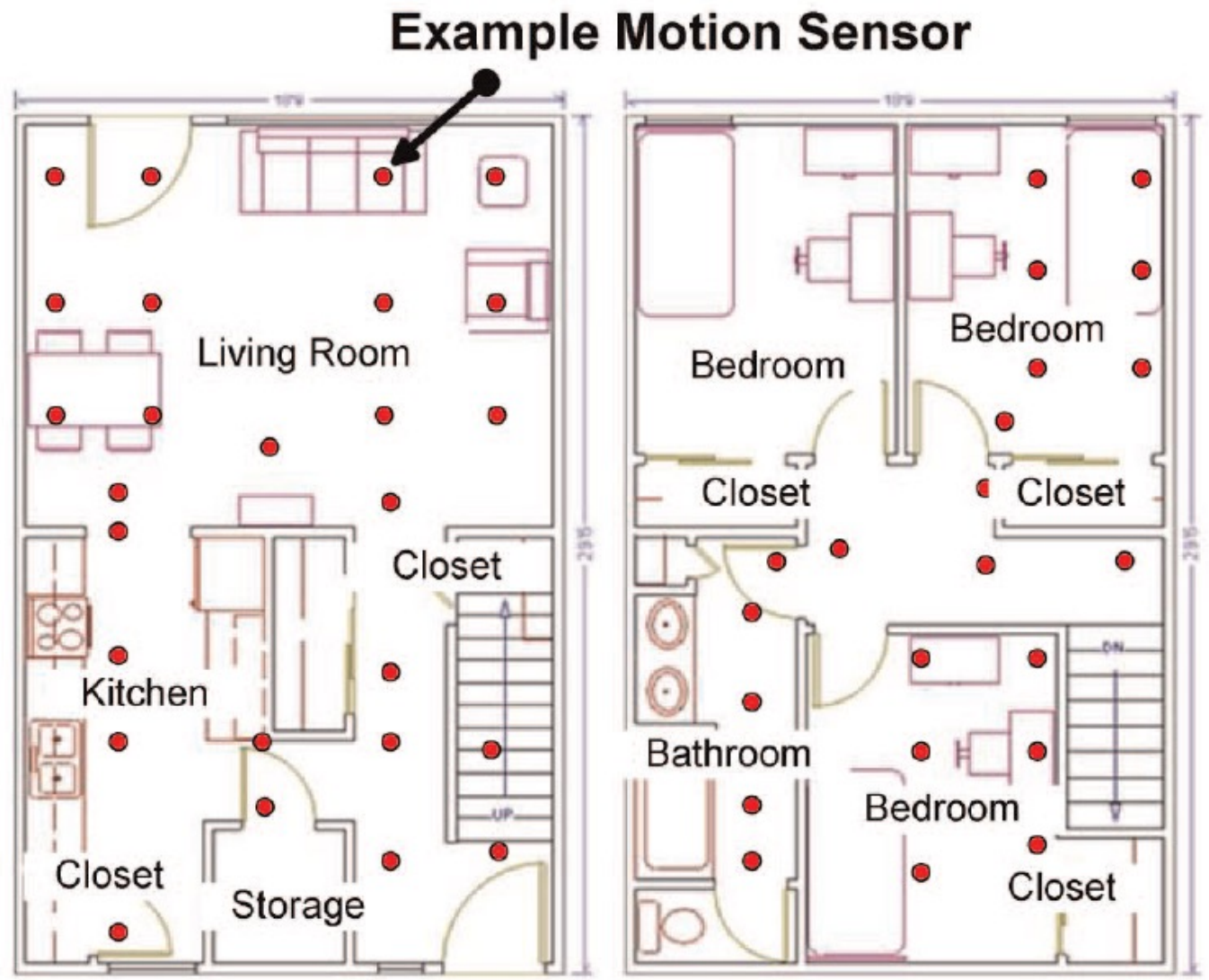
Hidden Markov model



Implementation



Implementation



Implementation

- The smart apartment testbed includes three bedrooms, one bathroom, a kitchen, and a living/dining room.
- The apartment is equipped with motion sensors positioned on the ceiling approximately 1 meter apart throughout the space.
- In addition, we have installed sensors to provide ambient temperature readings, and custom-built analog sensors to provide readings for hot water, cold water, and stove burner use.

motion sensor

Implementation

- Voice over IP using the Asterisk software captures phone usage, contact switch sensors monitoring the open/closed status of doors and cabinets, and pressure sensors monitor usage of key items such as the medicine container, cooking pot, and phone book.
- Sensor data are captured using a sensor network that was designed in house and is stored in an SQL database.

Sensor ID	time	value
8146000000B	12 10:50:45.673225	ON
E460000000D	12 10:50:48.903745	ON
A360000000F	12 12:30:09.56483	1.98

Resident performing “hand washing” activity. This activity triggers motion sensor ON/OFF events as well as water flow sensor values.

Implementation

- 20 Washington State University undergraduate students recruited from the psychology subject pool perform the following five activities:
  - Telephone Use: Look up a specified number in a phone book, call the number, and write down the cooking directions given on the recorded message.
  - Hand Washing: Wash hands in the kitchen sink.
  - Meal Preparation: Cook oatmeal on the stove according to the recorded directions, adding brown sugar and raisins (from the kitchen cabinet) once done.
  - Eating and Medication Use: Eat the oatmeal together with a glass of water and medicine (a piece of candy).
  - Cleaning: Clean and put away the dishes and ingredients.

Implementation