Summary of lecture 3 (I/III)

Investigated one linear discriminant (a function that takes an input and assigns it to one of \(K\) classes) method in detail (least squares).

Modelled each class as \(y_k(x, w) = w_k^T x + w_k,0\) and solved the LS problem, resulting in \(\hat{w} = (X^T X)^{-1} X^T T\).

Showed how probabilistic generative models could be built for classification using the strategy,

1. Model \(p(x \mid C_k)\) (a.k.a. class-conditional density).
2. Model \(p(C_k)\).
3. Use ML to find the parameters in \(p(x \mid C_k)\) and \(p(C_k)\).
4. Use Bayes’ rule to find \(p(C_k \mid x)\).

Summary of lecture 3 (II/III)

The “direct” method called logistic regression was introduced. Start by stating the model

\[
p(C_1 \mid \phi) = \sigma(w^T \phi) = \frac{1}{1 + e^{-w^T \phi}},
\]

which results in a log-likelihood function according to

\[
L(w) = -\ln p(T \mid w) = -\sum_{n=1}^{N} \left(t_n \ln(y_n) + (1 - t_n) \ln(1 - y_n)\right),
\]

where \(y_n = p(C_1 \mid \phi) = \sigma(w^T \phi)\). Note that this is a nonlinear, but concave function of \(w\).

Hence, we can easily find the global minimum using Newton’s method (resulting in an algorithm known as IRLS).

Summary of lecture 3 (III/III)

The likelihood function for logistic regression is

\[
p(T \mid w) = \prod_{n=1}^{N} \sigma(w^T \phi_n)^{t_n} (1 - \sigma(w^T \phi_n))^{1-t_n},
\]

Hence, computing the posterior density \(p(w \mid T) = \frac{p(T \mid w)p(w)}{p(T)}\) is intractable and we considered the Laplace approximation.

The Laplace appr. is a simple (local) appr. obtained by fitting a Gaussian centered around the (MAP) mode of the distribution.
Outline – Lecture 4

**Aim:** Introduce the neural network model and derive backpropagation.

Outline:
1. Summary of lecture 3
2. Generalize the linear model to a nonlinear function expansion
3. Successful examples of NN in real life examples
   - System identification
   - Handwritten digit classification
4. Training neural networks

---

### NN for regression – an example

1. Form $M$ linear combinations of the input $x \in \mathbb{R}^D$
   
   
   \[
   a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \quad j = 1, \ldots, M.
   \]

2. Apply a nonlinear transformation
   
   \[
   z_j = h^{(0)} \left( a_j^{(1)} \right), \quad j = 1, \ldots, M.
   \]

3. Form $M_y$ linear combinations of $z$
   
   \[
   a_j^{(2)} = \sum_{i=1}^{M} w_{ji}^{(2)} z_i + w_{j0}^{(2)}, \quad j = 1, \ldots, M_y.
   \]

4. Apply a nonlinear transformation
   
   \[
   y_j = h^{(1)} \left( a_j^{(2)} \right), \quad j = 1, \ldots, M_y.
   \]

---

### Learning from large-scale data

In many situations we have so much data that we cannot evaluate the cost function $E$ on the entire dataset.

Gives rise to a **stochastic optimization** problem.

Lots of research activity on this problem within the machine learning and optimization world right now!

A nice up to date summary is provided by

Ultrabrief current research snapshot

Key innovations

- Derived fast Cholesky routines to update a matrix containing curvature information.
- Exploit a receding history of iterates and gradients akin to L-BFGS.
- An auxiliary variable Markov chain construction.

Training a deep CNN for MNIST data.

Logistic loss function with an L2 regularizer, gisette, 6 000 observations and 5 000 unknown variables.


Two examples of neural networks in use

1. System identification
2. Handwritten digit classification

These examples will provide a glimpse into a few real life applications of models based in nonlinear function expansions (i.e., neural networks) both for regression and classification problems.

NN example 1 – system identification (I/V)

Neural networks are one of the standard models used in nonlinear system identification.

Problem background: The task here is to identify a dynamical model of a Magnetorheological (MR) fluid damper. The MR fluid (typically some kind of oil) will greatly increase its so called apparent viscosity when the fluid is subjected to a magnetic field.

MR fluid dampers are semi-active control devices which are used to reduce vibrations.

Input signal: velocity \( v(t) \) [cm/s] of the damper
Output signal: Damping force \( f(t) \) [N].

NN example 1 – system identification (II/V)

Have a look at the data
**NN example 1 – system identification (III/V)**

As usual, we try simple things first, that is a linear model. The best linear model turns out to be an output error (OE) model which gives 51% fit on validation data.

\[
\text{LinMod2} = \text{oe}(\text{ze}, [4 2 1]); \quad \% \text{OE model } y = B/F u + e
\]

Try a sigmoidal neural network using 10 hidden units

\[
\text{Options} = \{'\text{MaxIter}',50, '\text{SearchMethod}', '\text{LM}'\};
\]

\[
\text{Narx1} = \text{nlarx}(\text{ze}, [2 4 1], '\text{sigmoidnet}',\text{Options}{:});
\]

This model already gives a 72% fit on test/validation data.

\[
\text{compare}(\text{zv}, \text{Narx1});
\]

**NN example 1 – system identification (IV/V)**

Using 12 hidden units and only making use of some of the regressors,

\[
\text{Sig} = \text{sigmoidnet}(\text{'NumberOfUnits',12}); \quad \% \text{create Sigmoidnet object}
\]

\[
\text{Narx5} = \text{nlarx}(\text{ze}, [2 3 1], \text{Sig}, '\text{NonlinearRegressors}', [1 3 4],...
\]

\[
\text{Options}{:});
\]

the performance can be increased to a 85% fit on validation data.

Of course, this model need further validation, but the improvement from 51% fit for the best linear model is substantial.

This example if borrowed from


and it is used as one example in illustrating Mathwork’s toolbox,

se.mathworks.com/help/ident/examples/nonlinear-modeling-of-a-magneto-rheological-fluid-damper.html

**NN example 2 – digit classification (I/IV)**

You have tried solving this problem using linear methods before. Let us see what can be done if we generalize to nonlinear function expansions (neural networks) instead.

Let us now investigate 4 nonlinear models and one linear model solving the same task,

- **Net-1**: No hidden layer (equivalent to logistic regression).
- **Net-2**: One hidden layer, 12 hidden units fully connected.
- **Net-3**: Two hidden layers locally connected.
- **Net-4**: Two hidden layers, locally connected with weight sharing.
- **Net-5**: Two hidden layers, locally connected with two levels of weight sharing.

**NN example 2 – digit classification (II/IV)**

Local connectivity (Net3-Net5) means that each hidden unit is connected only to a small number of units in the layer before. It makes use of the key property that nearby pixels are more strongly correlated than distant pixels.
**The backpropagation algorithm**

1. **Input layer:** calculate the hidden units $z^1_j$ in the input layer.

2. **Feedforward:** starting with the first layer, compute

   $$z^l_j = \sum_{i=1}^{M} w^l_{ij} z^{l-1}_i + w^l_{0j}, \quad z^L_j = h(z^L_j),$$

   for all layers $l = 1, \ldots, L$.

3. **Error at top level:** calculate the error at the top level according to

   $$\delta^L_j = \frac{\partial E}{\partial a^L_j} h'(a^L_j).$$

4. **Backpropagate the error:** Use

   $$\delta^l_j = \left( \sum_{k=1}^{M} \delta^{l+1}_k w^{l+1}_{kj} \right) h'(a^l_j)$$

   to propagate the error backwards and calculate $\delta^l_j$ for all layers.

5. **Calculate the gradient:** by using

   $$\frac{\partial E}{\partial w^l_{ij}} = \delta^l_j z^{l-1}_i.$$
A few concepts to summarize lecture 4

**Neural networks:** A nonlinear function (as a function expansion) from a set of input variables \( \{x_i\} \) to a set of output variables \( \{y_k\} \) controlled by a vector \( w \) of parameters.

**Backpropagation:** Computing the gradients via the chain rule, combined with clever reuse of information that is needed for more than one gradient.

**Convolutional neural networks:** The hidden units takes their inputs from a small part of the available inputs and all units have the same weights (called weight sharing).

**Deep learning:** Models that automatically learns representations of data (features) with multiple layers of abstraction.