Lecture 6. Regression

Spring 2013.

Note Title 4/21/2013

Consider learning the conditional distribution $p(y|x)$. This is often easier than learning the likelihoods $p(x|y)$, the prior $p(y)$, and then compute the posterior $p(y|x)$.

Typically x has many more dimensions than y.

This is considered a regression problem. This is over 260 years old. Gauss finding the planetoid Ceres.

In this lecture, we adress three types of regression problem.

(I) Binary regression. $y \in \{\pm 1\}$

$$p(y|\underline{x};\underline{\lambda}) = \frac{e^{y\,\underline{\lambda}\cdot\underline{\varphi}(\underline{x})}}{e^{\underline{\lambda}\cdot\underline{\varphi}(\underline{x})} + e^{-\underline{\lambda}\cdot\underline{\varphi}(\underline{x})}}, \quad \text{or} \quad p(y|\underline{x};\underline{\lambda}) = \frac{e^{y\,\underline{\lambda}\cdot\underline{\varphi}(\underline{x})}}{Z[\underline{x};\underline{\lambda}]}$$

with $Z[\underline{x};\underline{\lambda}] = e^{\underline{\lambda}\cdot\underline{\varphi}(\underline{x})} + e^{-\underline{\lambda}\cdot\underline{\varphi}(\underline{x})}$

(II) Gaussian regression

$$p(y|\underline{x};\underline{\lambda}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-\underline{\lambda}\cdot\underline{\varphi}(\underline{x}))^2}{2\sigma^2}}$$

This can be extended to vector-valued y.

In both cases, the parameters can be estimated by maximum likelihood (ML).

$$\underset{\text{wrt } \underline{\lambda}}{\text{Minimize}} \quad -\sum_{i=1}^{n} \log P(y_i|\underline{x}_i;\underline{\lambda})$$

This is a convex optimization problem.

We can also include a prior $p(\underline{\lambda})$ and perform MAP optimization.

(III) Multi-Lager Perceptron.

This is an extension of (I).

It leads to non-convex optimization.

Binary Regression:

$$P(y|\underline{x}) = \frac{e^{y\,\underline{\lambda}\cdot\underline{\varphi}(\underline{x})}}{e^{\underline{\lambda}\cdot\underline{\varphi}(\underline{x})} + e^{-\underline{\lambda}\cdot\underline{\varphi}(\underline{x})}}$$

Note: this leads to a decision rule:

classify $\underline{x}$ as $y=+1$, if $\underline{\lambda}\cdot\underline{\varphi}(\underline{x}) > 0$

as $y=-1$, if $\underline{\lambda}\cdot\underline{\varphi}(\underline{x}) < 0$

or $\hat{y}(\underline{x}) = \operatorname{argmin}_y\, y\,\underline{\lambda}\cdot\underline{\varphi}(\underline{x})$

Spring 2013.

Minimize: $-\sum_{i=1}^{N} \log p(y_i | \underline{x}_i; \underline{\lambda})$

$$= -\sum_{i=1}^{N} y_i \underline{\lambda} \cdot \underline{\varphi}(\underline{x}_i) + \sum_{i=1}^{N} \log \{ e^{\underline{\lambda} \cdot \underline{\varphi}(\underline{x}_i)} + e^{-\underline{\lambda} \cdot \underline{\varphi}(\underline{x}_i)} \}$$

This is a convex function of $\underline{\lambda}$

(check → second term can be written as $\sum_i \underline{\lambda} \cdot \underline{\varphi}(\underline{x}_i)$)

$+ \sum_i \log \{ 1 + e^{-2\underline{\lambda} \cdot \underline{\varphi}(\underline{x}_i)} \}$ ... Hessian is pos definite

The gradient of $-\sum_{i=1}^{N} \log p(y_i | \underline{x}_i; \underline{\lambda})$ w.r.t. $\underline{\lambda}$ is $-\sum_{i=1}^{N} y_i \underline{\varphi}(\underline{x}_i) + \sum_{i=1}^{N} \sum_{y \in \{\pm 1\}} y \, \underline{\varphi}(\underline{x}_i) p(y | \underline{x}_i; \underline{\lambda})$

Hence the minimum also balances the statistics: $\hat{\underline{\lambda}}$ s.t.

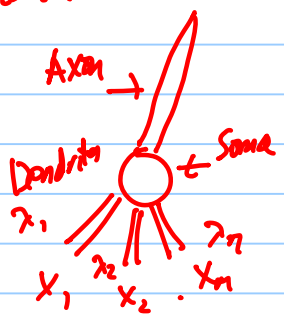
$$\sum_{i=1}^{N} y_i \underline{\varphi}(\underline{x}_i) = \sum_{i=1}^{N} \sum_{y \in \{\pm 1\}} y \, \underline{\varphi}(\underline{x}_i) p(y | \underline{x}_i; \hat{\underline{\lambda}}).$$

Note: also can use a prior $p(\underline{\lambda})$. (later lecture).

Algorithms – steepest descent, or variants –

Can be specified to minimize this

Special case – idealized "neuron"



$\underline{\varphi}(\underline{x}) = (x_1, x_2 \ldots x_n)$

$\underline{\lambda} = (\lambda_1, \lambda_2, \ldots \lambda_n)$

$\underline{\lambda} \cdot \underline{\varphi}(\underline{x}) = \lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_n x_n$.

Integrate-And-Fire: If $\underline{\lambda} \cdot \underline{\varphi}(\underline{x}) > 0$, then the neuron will fire with probability.

or threshold $\underline{\lambda} = (\lambda_0, \lambda_1, \ldots \lambda_n)$

$\underline{\varphi}(\underline{x}) = (1, x_1, \ldots x_n)$

$\underline{\lambda} \cdot \underline{\varphi}(\underline{x}) > 0$, $\sum_{i=1}^{n} \lambda_i x_i > -\lambda_0$.

Spring 2013.

Now Consider continuous regression with Gaussian model.

$$y = \underline{\lambda} \cdot \underline{\varphi}(\underline{x}) + \epsilon \qquad P(\epsilon) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\epsilon^2/2\sigma^2}$$

ϵ zero mean Gaussian

$$P(y \mid \underline{x}, \underline{\lambda}, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y - \underline{\lambda}\cdot\underline{\varphi}(\underline{x}))^2}$$

ML: minimize $-\sum_{i=1}^{N} \log P(y_i \mid \underline{x}_i, \underline{\lambda}, \sigma)$ dataset $\mathcal{X} = \{(\underline{x}_i, y_i) : i = 1, \ldots, N\}$

$$= \sum_{i=1}^{N} \frac{1}{2\sigma^2}(y_i - \underline{\lambda}\cdot\underline{\varphi}(\underline{x}_i))^2 - N \log\{\sqrt{2\pi}\sigma\}.$$

Minimize w.r.t. $\underline{\lambda}$

$$-\frac{1}{\sigma^2}\sum_{i=1}^{N}(y_i - \underline{\lambda}\cdot\underline{\varphi}(\underline{x}_i))\,\underline{\varphi}(\underline{x}_i) = 0$$

solution $$\hat{\underline{\lambda}} = \left\{\sum_{i=1}^{N} \underline{\varphi}(\underline{x}_i)\,\underline{\varphi}(\underline{x}_i)^T\right\}^{-1} \sum_{i=1}^{N} y_i\,\underline{\varphi}(\underline{x}_i)$$

Minimize w.r.t. σ $\quad -\frac{1}{\sigma^3}\sum_{i=1}^{N}(y_i - \underline{\lambda}\cdot\underline{\varphi}(\underline{x}_i))^2 - \frac{N}{\sigma}$

solution $$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{\underline{\lambda}}\cdot\underline{\varphi}(\underline{x}_i))^2$$ //

This estimates the regression coefficients $\underline{\lambda}$ and also the variance.

This can be generalized to allow for vector-valued output.

Similarly binary regression can be extended to multiclass: replace y $\underline{\lambda}\cdot\underline{\varphi}(\underline{x})$ by $\underline{\lambda}\cdot\underline{\varphi}(y, \underline{x})$.

Note: in terms of coordinates

$$\frac{\partial}{\partial \lambda_a} = \sum_{i=1}^{N}\left(y_i - \sum_a \lambda_a \varphi_a(\underline{x}_i)\right)\left(y_i - \sum_b \lambda_b \varphi_b(\underline{x}_i)\right)$$

$$2(-1)\sum_{i=1}^{N} \varphi_a(\underline{x}_i)\left(y_i - \sum_b \lambda_b \varphi_b(\underline{x}_i)\right)$$

Then the minimum occurs at

$$\sum_b \left\{\sum_{i=1}^{N} \varphi_a(\underline{x}_i)\,\varphi_b(\underline{x}_i)\right\} \lambda_b = \sum_{i=1}^{N} y_i\,\varphi_a(\underline{x}_i).$$

$\sum_{i=1}^{N} \varphi_a(\underline{x}_i)\,\varphi_b(\underline{x}_i)$ are the coefficients of the a^{th} row and b^{th} column of a matrix.

Spring 2013.

We can get a variant by specifying:

$$y = \underline{\lambda} \cdot \underline{\varphi}(\underline{x}) + \epsilon \quad , \text{ where } \quad P(\epsilon) = \frac{1}{2\sigma} e^{-|\epsilon|/\sigma}$$

$$\left(\int_0^\infty e^{-\epsilon/\sigma} d\epsilon = \left[-\sigma e^{-\frac{\epsilon}{\sigma}} \right]_0^\infty = \sigma \right).$$

$$P(y | \underline{x}, \underline{\lambda}, \sigma) = \frac{1}{2\sigma} e^{-\frac{1}{\sigma} | y - \underline{\lambda} \cdot \underline{\varphi}(\underline{x}) |}.$$

Estimate $\underline{\lambda}, \sigma$ by ML

$$\frac{1}{\sigma} \sum_{i=1}^{N} | y_i - \underline{\lambda} \cdot \underline{\varphi}(\underline{x}_i) | + N \log(2\sigma)$$

$$\hat{\underline{\lambda}} = \underset{\underline{\lambda}}{\text{ARG MIN}} \sum_{i=1}^{N} | y_i - \underline{\lambda} \cdot \underline{\varphi}(\underline{x}_i) |$$

This is a convex optimization problem - steepest descent and other algorithms will get $\hat{\underline{\lambda}}$

$$\hat{\sigma} = \frac{1}{N} \sum_{i=1}^{N} | y_i - \hat{\underline{\lambda}} \cdot \underline{\varphi}(\underline{x}_i) |.$$

This is more robust than the previous (Gaussian) method. It is L_1 norm instead of L_2.

Next we consider some specific examples from Alpaydin's book.

Spring 2013

(5) Alpaydin's notation is different.
He uses a function $g(\underline{x}) = \underline{\lambda} \cdot \underline{\varphi}(\underline{x})$
(in our notation).

For example,

$$\underline{\lambda} = (w_0, w_1, \ldots w_d)$$

$$g(\underline{x}) = w_1 x_1 + \ldots + w_d x_d + w_0 = \sum_{j=1}^{d} w_j x_j + w_0$$

Back to simple case — single attribute x_1

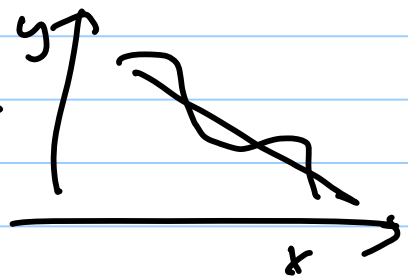
Dataset $\mathcal{X} = \{(x^i, y^i) : i = 1, N\}$

$$E(w_1, w_0 | \mathcal{X}) = \sum_{i=1}^{N} \left[y^i - (w_1 x^i + w_0) \right]^2$$

Minimize: $\frac{\partial E}{\partial w_1} = 0$ & $\frac{\partial E}{\partial w_0} = 0$

solution
$$\begin{cases} \hat{w}_1 = \dfrac{\sum_{i=1}^{N} x^i y^i - \bar{x}\bar{y}N}{\sum (x^i)^2 - N\bar{x}^2} \\ \hat{w}_0 = \bar{y} - w_1 \bar{x} \end{cases}$$

where $\bar{x} = \sum_{i=1}^{N} x^i / N$, $\bar{y} = \sum_{i=1}^{N} y^i / N$.



A "richer" model can be used.
→ eg $g(x) = w_2 x^2 + w_1 x + w_0$
Too high an order follows the data too closely.

Spring 2023

More Abstractly,

Linear regression: $g(x^i | w_1, w_0) = w_1 x^i + w_0$.

Differentiate energy wrt w_1, w_0 gives two equations

$$\sum_i y^i = N w_0 + w_1 \sum_i x^i$$

$$\sum_i y^i x^i = w_0 \sum_i x^i + w_1 \sum_i (x^i)^2.$$

Expressed in linear algebra form as $\underline{A}\,\underline{w} = \underline{y}$

$$\underline{A} = \begin{bmatrix} N & \sum_i x^i \\ \sum_i y^i & \sum_i (x^i)^2 \end{bmatrix}, \quad \underline{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \quad \underline{y} = \begin{bmatrix} \sum_i y^i \\ \sum_i x^i y^i \end{bmatrix}$$

solved to give $\underline{w} = \underline{A}^{-1}\underline{y}$.

Polynomial Regression.

$$g(x^i | w_k, \ldots, w_2, w_1, w_0) = w_k (x^i)^k + \ldots + w_1 x^i + w_0$$

$k+1$ parameters $w_k \ldots w_0$

Diff. energy - gives $k+1$ linear eqn's in $k+1$ variables.

$$\underline{A}\,\underline{w} = \underline{y}.$$

Can write $\underline{A} = \underline{D}^T \underline{D}$, $\underline{y} = \underline{D}^T \underline{r}$ $\quad \underline{D} = \begin{bmatrix} 1 & x_1 & x_1^k \\ 1 & x_2 & x_2^k \\ 1 & \cdot & \cdot \end{bmatrix}$

Solve to get $\underline{w} = (D^T D)^{-1} D^T r$ $\quad r = \begin{bmatrix} r^1 \\ r^2 \\ \vdots \\ r^N \end{bmatrix}$

Must adjust the complexity of the model to the amount of data available

Complexity of poly regression is no. parameters k.

Need to pick k to give best generalization error

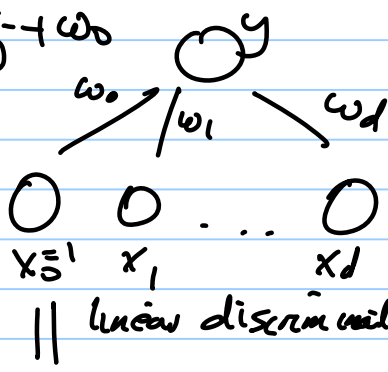
(7) Spring 2013.

Note Title 4/27/2008

Multilayer Perceptrons

Analogy to Neural Networks in the Brain.
- over-simplified.

Perceptron. $y = \sum_{j=1}^{d} w_j x_j + w_0$



idealized neuron

Threshold function.

hard $s(a) = \begin{cases} 1, & \text{if } a > 0 \\ 0, & \text{otherwise.} \end{cases}$

$y = s(w^T x)$ || linear discriminant.

soft $y = \sigma(w^T x) = \frac{1}{1 + e^{-w^T x}}$. $\sigma(\cdot)$ sigmoid function.

There are a variety of different algorithms to train a perceptron from labelled examples

Example: Quadratic error.

$E(w \mid x^t, y^t) = \frac{1}{2}(y^t - w x^t)^2$

update rule $\Delta w_j^t = -\eta \frac{\partial E^t}{\partial w_j} = +\eta (y^t - w x^t) x_j^t$

$E(\{w_i\}_i \mid x^t, y^t) = -\sum_i \left[r_i^t \log y_i^t + (1 - r_i^t) \log (1 - y_i^t) \right]$

$r^t = \text{sigmoid}(w_i^T x^t)$.

update rule $\Delta w_j^t = -\eta (r^t - y^t) x_j^t$.

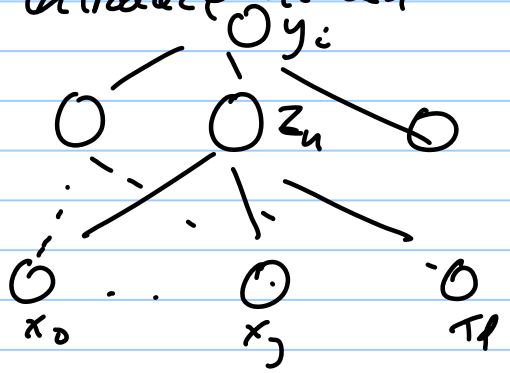
Update = Learning Factor. (Desired Output − Actual Output) × Input.

Spring 2013

Multilayer Perceptron.

A single layer perceptron can only approximate linear functions of the input — i.e. the set of perceptrons has limited capacity.

Multilayer perceptrons were invented to increase the capacity — introduce hidden units (nodes)



$$z_h = \sigma(w_h^T x) = \frac{1}{1+\exp\{-\sum_{j=1}^{d} w_{hj} x_j + w_{i0}\}}$$

$h = 1, \ldots H.$

$$\text{Output } y_i = v_i^T z = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}.$$

Other output function — e.g. $y_i = \sigma(v_i^T z)$.

Many levels can be specified.

What do the hidden units represent?

Unclear, but many people have tried to explain them

Any input-output function can be represented as a multilayer perceptron with enough hidden units — infinite capacity.

How to train a multilayer perceptron?

Unknown parameters – the weights ω_{hj}, v_{ij}.

Define an error function:

e.g. $E[\omega, v] = \sum_i \left(y_i - \sum_h v_{ih}\, \sigma\left(\sum_j \omega_{hj} x_j\right)\right)^2$

update $\Delta \omega_{hj} = -\dfrac{\partial E}{\partial \omega_{hj}}$ computed by the chain rule.

$\Delta v_{ik} = -\dfrac{\partial E}{\partial v_{ik}}$ computed directly

Define $r_k = \sigma\left(\sum_j \omega_{kj} x_j\right)$, $E = \sum_i \left(y_i - \sum_k v_{ik} r_k\right)^2$

$$\frac{\partial E}{\partial \omega_{kj}} = \frac{\partial E}{\partial r_k} \cdot \frac{\partial r_k}{\partial \omega_{kj}}$$

$$\frac{\partial E}{\partial r_k} = -2 \sum_i \left(y_i - \sum_l v_{il} r_l\right) v_{ik}$$

$$\frac{\partial r_k}{\partial \omega_{kj}} = x_j\, \sigma'\left(\sum_j \omega_{kj} x_j\right)$$

$$\sigma'(z) = \frac{d}{dz}\sigma(z) = \sigma(z)\left(1 - \sigma(z)\right).$$

Hence $$\frac{\partial E}{\partial \omega_{hj}} = -2 \sum_i \left(y_i - \sum_l v_{il} r_l\right) v_{ik}\, x_j\, \sigma\left(\sum_j \omega_{kj} x_j\right)\left\{1 - \sigma\left(\sum_j \omega_{kj} x_j\right)\right\}$$

($\left(y_i - \sum_l v_{il} r_l\right)$: error at output layer; v_{ik}: weights from middle layer to output layer.)

This is called backpropagation. The error at the output layer is propagated back to the nodes at the middle layer

$\sum_i \left(y_i - \sum_l v_{il} r_l\right) v_{ik}$

where it is multiplied by the activity $r_k(1 - r_k)$ at that node, and by the activity x_j at the input.

errors $\sum_i \left(y_i - \sum_l v_{il} r_l\right) v_{ik}$ multiply by $r_k(1-r_k)$ $\to$ r_k (middle layer) $\to$ ω_{kj} $\to$ x_j (input layer)

Spring 2013.

Learning in Batch Mode:

Put all data into an energy function - ie. sum the errors over all the training data. Update the weights - equations above - by summing over all the data.

Alternatively, online learning.

At time t, select one element $(\underline{x}^t, y^t)$ from the training set at random.
Perform one iteration of steepest descent only.
Then select another element at random.

Online learning can be shown to converge and may be better at avoiding local minima in the energy function than batch methods.

Also, online is suitable if we keep getting new input over time - this happens in many real world applications.

Spring 2013

Critical Issues for multilayer perceptrons

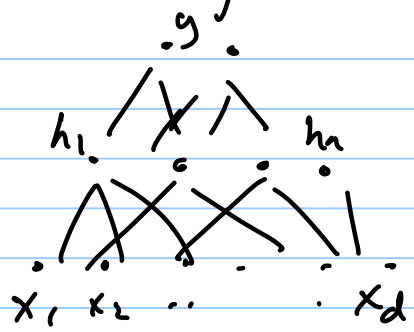
— how many hidden units to use?

Book describes several techniques for dealing with this → for example, having more hidden units than you need and then penalizing the weights.

Eg. Add term $\sum_{hj}(w_{hj}^2) + \sum_{ih}(v_{ih}^2)$ to the cost function → intuition, if the weights are small to a hidden unit, then the hidden unit is not used.

In practice, some of the most effective multilayer perceptrons are those which the structure was hand designed.

(12) Multilayer Perceptrons / SVM / AdaBoost Spring 2017
(next two lectures)



$$y_i = \sum_h v_{ih} h_h$$
$$h_h = \sum_j \sigma(\sum w_{hj} x_j).$$

SVM can also be represented in this way

$$y = \text{sign}\left(\sum_\mu \alpha_\mu y_\mu \underline{x}_\mu \cdot \underline{x}\right)$$

hidden units response $\underline{x}_\mu \cdot \underline{x} = h_\mu$.

$$y = \text{sign}\left(\sum_\mu \alpha_\mu y_\mu h_\mu\right)$$

Advantage of SVM — number of hidden units is given by the no. of support vectors.
→ $\{\alpha_\mu\}$ specified by maximizing the primal problem (well defined algorithm to perform this minimization).