ml notes

Leonardo Frioli @wikilele

Chapter 1

Intro

1.1 A bunch of stuff to know

In regression the target value is alway stated as:
$d_i = f(\mathbf{x}) + \epsilon$
where ϵ is the error (usually with 0 mean, and variance σ)

The Error is generally stated as:
$R_{emp} = \frac{1}{l}\sum_{p\to1}^{l} L(h(\mathbf{x}_p), d_p)$
L will change according to the task i.e MSE $(d_p - h(\mathbf{x}_p))^2$

Inductive learning Hypotesis
Any h that approximates f well on training examples will also approximate f well on new/unseen instances

Overfitting
A learner overfits the data if:
it outputs an hypotesis h $\in$ H having true error ϵ and empirical error E, but there is anothe h' $\in$ H aving E' $>$ E and ϵ' $<$ ϵ (greater empirical error, but smaller true error).

True error
$R = \int L(d, h(\mathbf{x}))\delta P(x, d)$

- we don't know the probability distribution
- the integral is over all the data

Recall (TP rate) = $\frac{TP}{TP+FN}$
Precision = $\frac{TP}{TP+FP}$

Chapter 2

Concept Learning

2.1 A bunch of stuff to know

For binary input/ouput $|H|^{\#-instances} = 2^{2^n}$ where n is the input dimension.

Let h_j and h_k be boolean valued functions defined over X. Then h_j is **more general then or equal to** h_k ($h_j \geq h_k$) if and only if
$\forall x \in X : (h_k(x) = 1) \rightarrow (h_j(x) = 1)$

G, of version sapce $VS_{H,D}$, summarizes all the negative up to now
S, of version space $VS_{H,D}$, summarizes all the positive up to now

An unbiased learner is unable to generalize because each unobserved instance will be classified positive by precisely half the hypothesis in VS and negative by the other half:
$\forall$ h consistent with $x_i \in TS$, $\exists$ h' identical to h except in x_i but identical on D (TR)

Inductive bias The inductive bias of a concept learning algorithm L is any minimal set of assertions B such that for any target concept c and corresponding training set $D_c = < x_p, c(x_p) >$
$\forall x_p \in X, (B \wedge D_c \wedge x_p) \vdash L(x_p, D_c)$

Chapter 3

Linear and K-nn models

3.1 One variable case

$$\frac{\delta E\left(\mathbf{w}\right)}{\delta w_i} = \frac{\delta\left(y - h_{\mathbf{w}}\left(x\right)\right)^2}{\delta w_i} = 2\left(y - h_{\mathbf{w}}\left(\mathbf{x}\right)\right)\frac{\delta\left(y - h_{\mathbf{w}}\left(x\right)\right)}{\delta w_i} =$$
$$2\left(y - h_{\mathbf{w}}\left(x\right)\right)\frac{\delta\left(y - \left(w_1 x + w_0\right)\right)}{\delta w_i}$$

w_0 derivative

$$\frac{\delta E(\mathbf{w})}{\delta w_0} = -2(y - h_{\mathbf{w}}(x)) = 0 \quad \sum_{p \to 1}^{l} -2(y_p - w_1 x_p - w_0) = 0$$
$$\sum y_p - w_1 \sum x_p - l w_0 = 0 \quad w_0 = \frac{1}{l}\sum y_p - \frac{1}{l} w_1 \sum x_p$$

w_1 derivative

$$\frac{\delta E(\mathbf{w})}{\delta w_1} = -2(y - h_{\mathbf{w}}(x))x = 0 \quad \sum_{p \to 1}^{l} -2(y_p - w_1 x_p - w_0)x_p = 0$$
$$\sum(x_p y_p - w_1 x_p^2 - w_0 x_p) = 0 \quad \sum(x_p y_p) - w_1 \sum x_p^2 - w_0 \sum x_p = 0$$
$$w_1 \sum x_p^2 = \sum x_p y_p - \frac{1}{l}\sum x_p \sum y_p + \frac{1}{l} w_1 \sum x_p \sum y_p$$
$$w_1(\sum x_p^2 - \frac{1}{l}(\sum x_p)^2) = \sum x_p y_p - \frac{1}{l}\sum x_p \sum y_p$$

3.2 Scaling freedom property

Hyperplane equation:
$\mathbf{w}^T\mathbf{x} + w_0 = w_1 x_1 + w_2 x_2 + w_0 = 0$
$x_2 = x_1 \frac{Kw_1}{kw_2} - \frac{Kw_0}{Kw_2}$

3.3 Direct Approach

$$E(\mathbf{w}) = \sum_{p\to1}^{l}(y_p - \sum_{t\to0}^{n} w_t x_{p,t})^2 = \sum_{p\to1}^{l}\delta_p(\mathbf{w})^2$$

For j → 0 to n

$$\frac{\delta E(\mathbf{w})}{\delta w_j} = 2\sum_{p\to1}^{l}\delta_p(\mathbf{w})\frac{\delta\ \delta_p(\mathbf{w})}{\delta w_j} =$$

$$2\sum_{p}\delta_p(\mathbf{w})\frac{\delta(y_p - \sum_{t\to0}^{n} w_t x_{p,t})}{\delta w_j} = -2\sum_{p} x_{p,j}\delta_p(\mathbf{w})$$

$$-2\sum_{p} x_{p,j}(y_p - \sum_{t\to0}^{n} w_t x_{p,t}) = 0$$

$$\sum_{p\to1}^{l} x_{p,l}y_p = \sum_{p\to1}^{l}\sum_{t\to0}^{n}(x_{p,j}x_{p,t}w_t)$$

$$X^T y = (X^T X)w$$

3.4 Gradient descend as correction rule

$$\Delta\mathbf{w} = -\frac{\delta E(\mathbf{w})}{\delta\mathbf{w}}$$

$$\Delta w_j = \sum_{p\to1}^{l}(x_p)(y_p - \mathbf{x}_p^T\mathbf{w})$$

If input > 0:

If error is positive→ output is too low → increase w_j → increase output → decrease error

If error is negative → output is too high → decrease w_j → decrease output → decrease error

If input < 0:

If error is positive → output is too low → decrease w_j → increase output → decrease error

Riducendo w_j é minore il peso che $input_j$ ha nella somma e di conseguenza maggiore l'ouput dato che l'input é minore di 0

If error is negative → output is too high → increase w_j → decrease output → decrease error

3.5 LBE

$h_{\mathbf{w}}(\mathbf{x}) = \sum_{i \to 0}^{m} w_i \phi_i(\mathbf{x})$
Pay attention to the dimension of the input now it is m and $m > n$

3.6 Tikhonov and STL

Thikhonov regularization

$$E(\mathbf{w}) = \sum_{p \to 1}^{l} (y_p - \mathbf{x}_p^T \mathbf{w})^2 + \lambda \left\|\mathbf{w}\right\|^2$$
$$\left\|\mathbf{w}\right\|^2 = \sum w_i^2$$
$$\mathbf{w}_{new} = \mathbf{w}_{old} + \eta * \Delta\mathbf{w} - 2\lambda \mathbf{w}_{old}$$

SLT

$$\mathbf{R} \leq \mathbf{R}_{emp} + \epsilon(\frac{1}{l}, VC - dim, \frac{1}{\delta})$$

1. Why Tikhonov can help to have a better bound on R?
   Because high VC-dim $\Leftrightarrow$ high $\mathbf{w}$ values
   La regularization diminuisce/controlla i valori dei pesi, quindi la VC, quindi R come mostra la SLT

2. low $\lambda \to$ flexible model $\to$ overfitting
   high $\lambda \to$ rigid model $\to$ underrfitting

3. $\frac{\delta\lambda\|\mathbf{w}\|^2}{\delta w_j} = \frac{\delta\lambda \sum w_i^2}{\delta w_j} = 2\lambda w_j$

3.7 K-nn

1-NN
$i(\mathbf{x}_{new}) = arg\, min_j\ d(\mathbf{x}_{new}, \mathbf{x}_j)$
Euclidian distance : $d(\mathbf{x}, \mathbf{x}_j) = \sqrt{\sum_{i \to 1}^{n} (x_i - x_{ji})^2} = \|\mathbf{x} - \mathbf{x}_j\|$

$\frac{N}{k}$ effective number of parameters:
if k is low $\to \frac{N}{k}$ is high $\to$ overfitting
if k is high $\to \frac{N}{k}$ is low $\to$ underfitting
where $k \to 1..N$

- variable scaling and different input ranges have high impact
- computationally expensive
- metric dependent

- **Curse of Dimensionality**
  The volume of the problem space increases so fast that the available data becomes sparse.
  In ten dimension (n = 10) we need to cover 80% of the range of each coordinate to capture 10% of data.
  Estimates are no longer local.
  If we want local estimates we should reduce k → overfitting

- **Sampling density** $\alpha \, l^{\frac{1}{dim}}$
  if 100 point are sufficient to estimate a function in $\mathbb{R}^1$
  100^{10} are needed to achieve similer accuracy in $\mathbb{R}^{10}$

- curse of noisy (irrelevant features)

Chapter 4

Neural Networks

4.1 Perceptron learning algorithm

$$if\ out \neq d \rightarrow \mathbf{w}_{new} = \mathbf{w}_{old} + \eta d\mathbf{x}$$
$$else \rightarrow do\ nothing$$

4.2 Perceptron Convergence Theorem

The perceptron learning algorithm is guarenteed to converge (calssifying correctly all the input patterns) in a finite number of steps is the patterns are linearly separable
Two groups are **linearly separable** in a n-dimensional space if they can be separated by a (n-1)-dimensional hyperplane.

Proof.
We can focus only on positive patterns:
Assume $(x_i,\, d_i)$ where $d_i = \pm 1$
Linear separable $\rightarrow \exists \mathbf{w}^*$ solution such that $d_i(\mathbf{w}^*\mathbf{x}_i) \geq \alpha$
where $\alpha = min_i d_i(\mathbf{w}^*\mathbf{x}_i) > 0$
given $\mathbf{x'}_i = (d_i\mathbf{x}_i)$
$\mathbf{w}^*$ is a solution $\Leftrightarrow$ $\mathbf{w}^*$ is a solution for $(\mathbf{x'}_i, +1)$
if $\mathbf{w}^*$ is a solution for $\mathbf{x}_i \rightarrow d_i(\mathbf{w}^*\mathbf{x}_i) \geq \alpha \rightarrow (\mathbf{w}^* d_i \mathbf{x}_i) \geq \alpha \rightarrow 1(\mathbf{w}^*\mathbf{x'}_i) \geq \alpha \rightarrow$ $\mathbf{w}^*$ is a solution for $\mathbf{x'}_i$
if $\mathbf{w}^*$ is a solution for $\mathbf{x'}_i \rightarrow\rightarrow (\mathbf{w}^* d_i \mathbf{x}_i) \geq \alpha \rightarrow d_i(\mathbf{w}^*\mathbf{x}_i) \geq \alpha \rightarrow \mathbf{w}^*$ is a solution for $\mathbf{x}_i$

Assume $\mathbf{w}(0) = 0$, $\eta = 1$ and $\beta = max_i|\mathbf{x}_i|^2$
After q misclassifications (all false negative)
$\mathbf{w}(q) = \sum_{j \rightarrow 1}^{q} \mathbf{x}_{(i_j)}$
beacuse $\mathbf{w}(j) = \mathbf{w}(j-1) + \mathbf{x}_{i_j}$

Lower bound on $\|\mathbf{w}(q)\|$

$$\mathbf{w}^*\mathbf{w}(q) = \mathbf{w}^* \sum_j^q \mathbf{x}_{i_j} \geq q\alpha$$

$$(\mathbf{w}\mathbf{v})^2 \leq \|\mathbf{w}\|^2 \|\mathbf{v}\|^2 \, (Cauchy - Swartz\; inequality)$$

$$\|\mathbf{w}^*\|^2 \|\mathbf{w}(q)\|^2 \geq (\mathbf{w}^*\mathbf{w}(q))^2 \geq (q\alpha)^2$$

$$\|\mathbf{w}(q)\|^2 \geq \frac{(q\alpha)^2}{\|\mathbf{w}^*\|^2}$$

Upper bound on $\|\mathbf{w}(q)\|$

$$\|\mathbf{w}(q)\|^2 = \left\|\mathbf{w}(q-1) + \mathbf{x}_{i_q}\right\|^2 = \|\mathbf{w}(q-1)\|^2 + 2\mathbf{w}(q-1)\mathbf{x}_{i_q} + \left\|\mathbf{x}_{i_q}\right\|^2$$

$$2\mathbf{w}(q-1)\mathbf{x}_{i_q} < 0$$

$$\|\mathbf{w}(q)\|^2 \leq \|\mathbf{w}(q-1)\|^2 + \left\|\mathbf{x}_{i_q}\right\|^2$$

$$\|\mathbf{w}(q)\|^2 \leq \sum_j^q \left\|\mathbf{x}_{i_j}\right\|^2 \leq q\beta$$

$$\|\mathbf{w}(q)\|^2 \leq q\beta$$

4.3 Cybenko's Theorem

A single hidden-layer network (with logistic activation functions) can approximate (arbitrarly well) every continuous function (on hyper cubes) provided enough units in the hidden layer.

$$\forall x \in \; hypercube, \exists h \; such \; that \; |f(x) - h(x)| < \epsilon$$

4.4 Sigmoid activation function

First derivative

$$\frac{\delta}{\delta x}\frac{1}{g(x)} = -\frac{g(x)'}{g(x)^2}$$

$$\frac{\delta}{\delta x}e^{f(x)} = e^{f(x)}\frac{\delta}{\delta x}f(x)$$

$$f_\sigma(x) = \frac{1}{1+e^{-ax}}$$

Assume a == 1

$$f'_\sigma(x) = -\frac{(1+e^{-x})'}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1+e^{-x}-1}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}}[\frac{(1+e^{-x})-1}{1+e^{-x}}] = \frac{1}{1+e^{-x}}(1-\frac{1}{1+e^{-x}})$$

4.5 Δw_j

$$\frac{\delta E(\mathbf{w})}{\delta w_j} = \frac{\delta \sum_p \frac{1}{2}(d_p - f_\sigma(\mathbf{x}_p^T\mathbf{w}))^2}{\delta w_j} = \sum_p (\frac{\delta(d_p - f_\sigma(net))^2}{\delta net}\frac{\delta net}{\delta w_j}) =$$

$$\sum_p \frac{1}{2}(2(d_p - f_\sigma(net))\frac{(d_p - f_\sigma(net))}{\delta net}\frac{\mathbf{x}_p^T\mathbf{w}}{\delta w_j}) =$$

$$\sum_p (d_p - f_\sigma(net))(-1)\frac{f_\sigma(net)}{\delta net}(x_p)_j$$

4.6 NN with only linear activation functions

$$h(\mathbf{x}) = \sum_j w_{kj}(\sum_i w_{ij}x_i) = \sum_i(\sum_j w_{kj}w_{ij})x_i$$

Example with two hidden units:

$$w_{k1}(w_{11}x_1 + w_{12}x_2) + w_{k2}(w_{12}x_1 + w_{22}x_2)$$
$$(w_{k1}w_{11} + w_{k2}w_{12})x_1 + (w_{k1}w_{21} + w_{k2}w_{22})x_2$$

4.7 Cascade Correlation

Derivative of absolute value

$$\frac{\delta|f(x)|}{\delta x} = sign(f(x))\frac{\delta f(x)}{\delta x}$$

$$S = \sum_k |S_k| \quad \frac{\delta S}{\delta w_j} = \sum_k signS_k \frac{\delta S_k}{\delta w_j}$$
$$\frac{\delta S_k}{\delta w_j} = \frac{\delta \sum_p (O_p - mean_p(O))err_{k,p}}{\delta w_j}$$
$$err_{k,p} = E_{k,p} - mean_p(E_k)$$
$$\frac{\delta \sum_p (O_p - mean_p(O))err_{k,p}}{\delta O_p}\frac{\delta O_p}{\delta net_p}\frac{\delta net_p}{\delta w_j}$$
$$err_{k,p}f'(net_p)(I_j)_p = \frac{\delta S_k}{\delta w_j}$$

Chapter 5

Backpropagation

$$E_{TOT} = \sum_p E_p$$

$$E_p = \sum_k \frac{1}{2}(y_k - o_k)^2$$

$$\Delta_p w_{ti} = -\frac{\delta E_p}{\delta w_{ti}} = -\frac{\delta E_p}{\delta net_t}\frac{\delta net_t}{\delta w_{ti}} = \delta_t o_i$$

$$\frac{\delta net_t}{\delta w_{ti}} = \frac{\delta \sum_r w_{tr} o_r}{\delta w_{ti}} = o_i$$

$$\delta_t = -\frac{\delta E_p}{\delta net_t} = -\frac{\delta E_p}{\delta o_t}\frac{\delta o_t}{\delta net_t}$$

$$\frac{\delta o_t}{\delta net_t} = \frac{\delta f_t(net_t)}{\delta net_t} = f'_t(net_t)$$

If t == k and k is an output unit:

$$-\frac{\delta E_p}{\delta o_k} = -\frac{\delta \frac{1}{2}\sum_r (y_r - o_r)^2}{\delta o_k} = (y_k - o_k)$$

$$\delta_k = (y_k - o_k) f'_k(net_k) = -\frac{\delta E_p}{\delta net_k}$$

If t == j and j is and hidden unit:

$$-\frac{\delta E_p}{\delta o_j} = \sum_k -\frac{\delta E_p}{\delta net_k}\frac{\delta net_k}{\delta o_j} = \sum_k \delta_k w_{kj}$$

$$\frac{\delta net_k}{\delta o_j} = \frac{\sum_r w_{kr} o_r}{\delta o_j} = w_{kj}$$

$$\delta_j = (\sum_k \delta_k w_{kj}) f_j'(net_j)$$

Since training examples provide target values t_k only for network outputs, no target values are directly available to indicate the error of the hidden units' values. Instead, the error term for hidden unit j is calculates by summing the error terms δ_k for each output unit influenced by j, weighting each of the δ_k's by w_{kj}, the weight from hidden unit j to output unit k. This weight characterizes the degree to which hidden unit j is "responsible for" the error output uni k. [Mitchell]

Chapter 6

SVM

6.1 Vapnik's Theorem

Let D denote the diameter of the smallest ball containing all the input vectors $\mathbf{x}_1$, ... $\mathbf{x}_n$. The set of optimal hyperplanes described by the equation

$$\mathbf{w}_o^T\mathbf{x} + b_o = 0$$

has VC dimension h bounded from above as

$$h \leq min\{\left\lceil \frac{D^2}{\rho^2} \right\rceil, m_0\} + 1$$

where m_0 is the dimensionality of the input space. [Haykin]

6.2 Cover's Theorem

A multidimensional space may be transformed into a new feature space where the patterns are linearly separable with high probability, provided two conditions are satisfied:

1. the trasformation is non linear
2. the dimensionality of the feature space is high enough

[Haykin]

6.3 SVM and curse of dimensionality

Numerical optimization in a high-dimensional space suffers from the curse of dimensionality. This computational problem is avoided by using the notion of an inner-product kernel (defined in accordance with Mercer's theorem) and solving the dual form of the constrained optimization problem formulated in the input space. [Haykin]

Chapter 7

Bias Variance

7.1 Variance Lemma

Expected value or mean

$$\underline{Z} = E_P[Z] = \sum_{i \to 1}^{n} z_i P(z_i)$$

Variance

$$Var[Z] = E[(Z - \underline{Z})^2] = \sum_{i \to 1}^{n} (z_i - \underline{Z})^2 P(z_i) = [...] = E[Z^2] - \underline{Z}^2$$

We will use the form

$$E[Z^2] = \underline{Z}^2 + E[(Z - \underline{Z})^2]$$

7.2 Bias-Varince Decomposition

$$E_p[(y - h(\mathbf{x}))^2] = E_P[h(\mathbf{x})^2] + E_P[y^2] - 2E_P[y]E_P[h(\mathbf{x})]$$

$E_P[XY] = E_P[X]E_P[Y] \Leftrightarrow$ X and Y are independent
In this case y and h($\mathbf{x}$) are independent.

$$\overline{h}(\mathbf{x}) = E_P[h(\mathbf{x})]$$
$$E_P[h(\mathbf{x})^2] = E_P[(h(\mathbf{x}) - \overline{h}(\mathbf{x}))^2] + \overline{h}(\mathbf{x})^2$$
$$E_P[y] = E_P[f(\mathbf{x}) + \epsilon] = f(\mathbf{x})$$
$$E_P[y^2] = E_P[(y - f(\mathbf{x}))^2] + f(\mathbf{x})^2$$
$$[...]$$
$$E_P[(h(\mathbf{x}) - \overline{h}(\mathbf{x}))^2] \quad \mathit{variance}$$
$$(\overline{h}(\mathbf{x}) - f(\mathbf{x}))^2 \quad \mathit{bias}^2$$
$$E_P[(y - f(\mathbf{x}))^2] \quad \mathit{noise}$$

Chapter 8

Unsupervised Learning

8.1 Quantization Error

$E = \sum_i \sum_j \|\mathbf{x}_i - \mathbf{w}_j\|^2 \, \delta_{winner}(i,j)$

8.2 K-means

$\Delta \mathbf{w}_{i^*} = \eta \, \delta_{winner}(i,j)(\mathbf{x}_i - \mathbf{w}_{i^*})$

8.3 SOM

Competitive stage
$i^*(\mathbf{x}) = argmin_i \|\mathbf{x} - \mathbf{w}_i\|$

Cooperative stage
$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta(t) h_{i,i*}(t)[\mathbf{x} - \mathbf{w}_i(t)]$
$h_{i,i*}(t) = exp(-\frac{\|\mathbf{r}_i - \mathbf{r}_{i^*}\|^2}{2\sigma^2_{nh}(t)})$
where $\mathbf{r}_i$ are the coord of unit i and σ is the widht