Machine Learning scheme

BISHOP

* Regression (Continue output)

- Linear regression

- algorithm, based on Gaussian density p(t|x, w, B), **max likelihood = min sum of squares error** (w based on design matrix) (for small set)

- algorithm Sequential learning, based on **Stochastic descendent gradient** (weights update, for large set and real time application)

* Classification (Discrete output)

- based on Discriminant function

- (k > 2) one-versus-rest, k-1 classifier

- (k > 2) one-versus-one, k(k-1)/2 classifier

- Least square error, k classes (matrix W, X, T)

- Fisher, 2 and k classes (*direction of projection* in one dimension)

- Perceptron, 2 classes

(to learn the weigh vector, we use the following methods)

- Perceptron training rule (samples linearly separable, single weight update, wi)

- Delta rule (samples not linearly separable - not threshold), uses *gradient of the error* to update vector **w**

- Support vector, 2 and k classes (uses *Lagrange*)

- based on Conditional probability p(Ck|x) approach

- Generative model, **use p(x|Ck), p(Ck), Bayes to estimate p(Ck|x)**

(estimate of parametric model: sigmoid and softmax function)

suppose continues input and that p(x|Ck) are gaussian density where classes have the same covariance matrix. We use *max likelihood* to estimate parameters for k=2.

- Discriminative model *(Logistic regression)*, **estimate P(Ck|x) directly, from a parametric model** of probability. NO Bayes. We define a *Cross entropy error* (*to minimize* through the *Newton-Raphson* with Hessian) in order to maximize likelihood (k>=2, NO estimate parameters)

* *KNN*, based on sphere, euclidean space and Bayes theorem
* UNSUPERVISED LEARNING, Mixture Models (instances belongs to different clusters)

- *K-means*, in a space of euclidean variable we want assign a point to a cluster (group of points) by minimizing the *distortion measure J* according two iterative steps

* + - *EM*, generalization of k-means, based on two step expectation- maximization to increase the log likelihood that falls below some threshold (expensive method)
* MULTIPLE LEARNER, Combining Models (same instances are classified by different classifiers, average)

- *Bagging*, the committee prediction is based on the average of M bootstrap data set

- *Boosting*, weighting coefficient depends on the previous classifier, when all the classifiers are trained, their prediction are combined together

- Ada boost

- Ada boost explained as **Minimizing exponential error,** where for simplicity, m-1 classifier are constant

MITCHELL

* Concept learning
* Decision tree

- ID3

* Hypothesis evaluation Decision tree

- Sample error

- True error

* Bayesian learning

- Bayes theorem

- Naive Bayes classifier

DEEP LEARNING  
- All layers, use Activation function (given an input and a AF, compute output for this layer)

- ReLu

- Logistic sigmoid

- Hyperbolic tangent

- Output layer, to compute the cost function J (Algorithm 6.3) equal to the loss function

- Regression, linear output units, min mean squared error function

- Binary classification, sigmoid o.u., min binary cross-entropy

- Multiple classification, softmax o.u., min categorical cross-entropy

Backpropagation computes (not learn) gradient of cost function (Algorithm 6.4) through chain-rule (which can be interpreted as an indication of how each layer’s output should change to reduce error).

To learn the gradient are used these algorithm:

* + Stochastic gradient descent
  + SGD with momentum, to accelerate learning
  + SGD with Nesterov momentum (variance of previous case)
* Regularization, reduce the test error, possibly at the expense of increased training error (reduce overfitting)
  + - Parameter sharing
    - Early stopping, control increase of test loss
    - Dropout, remove neurons from a layer with some probability alfa