Lecture 2

Von-Wun Soo

National Tsing Hua University

Outlines

- Naïve Bayesian learning
- Dimension reduction, principle component analysis (1hr)
- Feature selection, feature scaling, feature transformation,
- Batch normalization (1 hr)
- Support vector machine classification (1hr)
- Active learning (1hr)

Bayesian learning

- Bayes theorem: $P(h|D)=P(D|h)P(h)/P(D)$
- Maximum a posteriori (MAP) hypothesis:

$$h_{MAP} \equiv \text{argmax}_h\ p(h|D)$$
$$= \text{argmax}_h\ P(D|h)P(h)/P(D)$$
$$= \text{argmax}_h\ P(D|h)P(h)$$

- Maximum likelihood (ML) hypothesis:

$$h_{ML} \equiv \text{argmax}_h\ P(D|h)$$

Bayesian learning

- Each training instance x is described by a tuple of attribute values <a1,a2,…,an>
- The training instances are classified into V classes
- The Bayesian learning is to classify the training instance into the most probable class v, v_{MAP}, given the attribute values <a1,a2,…,an>
- v_{MAP}= $\text{argmax}_{vj}P(v_j|a1,a2,…,an)$

Naïve Bayesian learning

- According to Bayes theorem
- v_{MAP}= $\text{argmax}_{vj}P(v_j|a1,a2,\ldots,an)$

  = $\text{argmax}_{vj}P(a1,a2,\ldots,an|v_j)P(v_j)/P(a1,a2,\ldots an)$

  = $\text{argmax}_{vj}P(a1,a2,\ldots,an|v_j)P(v_j)$
- Naïve Bayesian assume attribute values are conditional independence:

  v_{MAP}= $\text{argmax}_{vj}P(v_j)\prod_i P(a_i|v_j)$

Example

day	outlook	temp	humidity	wind	Play tennis
d1	sunny	hot	high	weak	no
d2	sunny	hot	high	strong	no
d3	overcast	hot	high	weak	yes
d4	rain	mild	high	weak	yes
d5	rain	cool	normal	weak	yes
d6	rain	cool	normal	strong	no
d7	overcast	cool	normal	strong	yes

Example

d8	sunny	mild	high	weak	no
d9	sunny	cool	normal	weak	yes
d10	rain	mild	normal	weak	yes
d11	sunny	mild	normal	strong	yes
d12	overcast	mild	high	strong	yes
d13	overcast	hot	normal	weak	yes
d14	rain	mild	high	strong	no

Example --instance

- Query instance:

<outlook=sunny,temp=cool,humidity=high,
wind=strong>

- v_{NB}= argmax$_{vj=\{yes,no\}}$ p(v_j)Π_i(P(a|v_j)

  =argmax$_{vj=\{yes,no\}}$ p(v_j) P(outlook=sunny|yes)
  P(temp=cool|yes)P(humidity=high|yes)
  P(wind=strong|yes)

- P(yes)=9/14, P(no)=5/14

Example --instance

- P(wind=strong|yes)=3/9
  P(wind=strong|no)=3/5
- P(yes)p(sunny|yes)p(cool|yes)P(high|yes)p(strong|yes)=0.0053
- P(no)p(sunny|no)p(cool|no)P(high|no)p(strong|no)=0.0206
- So the answer is no.

Naïve Bayesian learning

- Advantages:
  - no need to search hypothesis space, just combine counting
  - Has been shown to be good for learning text classification
- Weakness: can be inaccurate in estimating probabilities when sample size is small

Remedy of estimating probability

- Remedy: m-estimate of probability

  $(n_c + m\ p)/(n + m)$

  where m is equivalent sample size and

  p is prior estimate of the probability

- Namely, adding m virtual instances

Curse of dimensionality

- In machine learning, it usually encounters domains with high dimension of features to express the training examples.
- Most of them might be irrelevant to problem solving goals.
- But it causes computational intractable.

- For example, $10^2=100$ evenly spaced sample points suffice to sample a unit interval (a "1-dimensional cube") with no more than $10^{-2}=0.01$ distance between points; an equivalent sampling of a 10-dimensional unit hypercube with a lattice that has a spacing of $10^{-2}=0.01$ between adjacent points would require $10^{20}[=(10^2)^{10}]$ sample points.

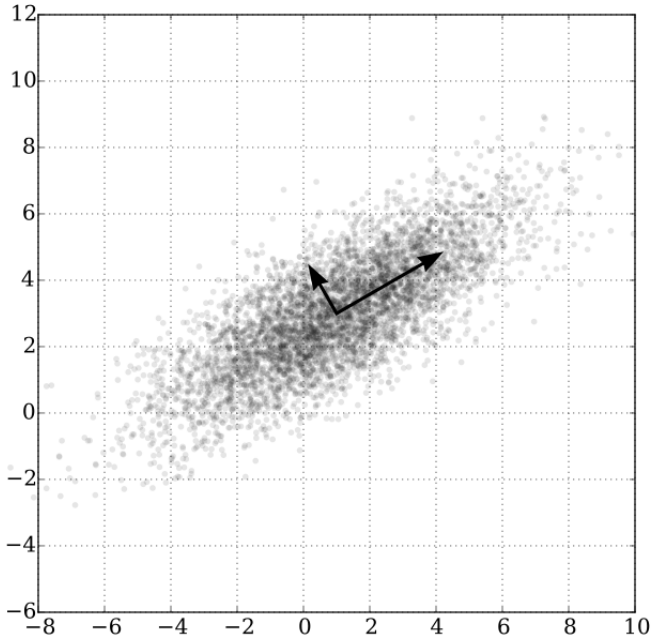
Principle component analysis as Dimension reduction

- **PCA** is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called **principal components.**

- **PCA** is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

PCA -- an illustration

- PCA of a multivariate Gaussian distribution centered at (1,3) with a standard deviation of 3 in roughly the (0.866, 0.5) direction and of 1 in the orthogonal direction.
- The vectors shown are the eigenvectors of the covariance matrix (XX^T) scaled by the square root of the corresponding eigenvalue, and shifted so their tails are at the mean.



Mapping X to unit vectors

- Data matrix $\mathbf{X_{nxp}}$, **with column-wise zero** empirical mean while each of the *n* rows represents a record of experiment, each of the *p* columns represents a specific feature.

- Mathematically, the transformation by a set of *p*-dimensional vectors of weights or *loadings* $w_{(k)} = (w_1,\ldots,w_p)_{(k)}$ that
- Map each row vector $x_{(i)}$ of **X** to a new vector of principal component *scores* $t_{(i)} = (t_1,\ldots,t_m)_{(i)}$,

$$t_{k(i)} = x_{(i)} \cdot w_{(k)} \quad \text{for} \quad i=1,\ldots,n \;\; ; \;\; k=1,\ldots,m$$

- Individual variables of **t** considered over the data set successively inherit the maximum possible variance from **x**, with each loading vector **w** constrained to be a unit vector.

First principle component

In order to maximize variance, the first loading vector $\mathbf{w_{(1)}}$ thus has to satisfy

$$w(1) = \text{argmax}_{||w||=1}\{\sum_i (t_1)_{(i)}^2\}$$
$$= \text{argmax}_{||w||=1}\{\sum_i (x_{(i)} \cdot w)^2\}$$

Equivalently, in matrix form it gives:

$$w(1) = \text{argmax}_{||w||=1}\{||Xw||^2\}$$
$$= \text{argmax}_{||w||=1}\{w^T X^T X w\}$$

It equivalently satisfies

$$w(1) = \text{argmax}\{\frac{w^T X^T X w}{w^T w}\}$$

K-th component (orthonormal basis)

The *k*-th component can be found by subtracting the first $k - 1$ principal components from **X:**

$$\hat{X}_k = X - \sum_{s=1}^{k-1} X\, w_{(s)} w_{(s)}^T$$

and find the loading vector that extracts the maximum variance from this new data matrix

$$w_{(k)} = \operatorname{argmax}_{||w||=1}\{||\hat{X}_k w||^2\}$$

$$= \operatorname{argmax}\left\{\frac{w^T X_k^T X_k w}{w^T w}\right\}$$

Eigen values and Eigen vectors of $\mathbf{X}^T\mathbf{X}$

- The quantity is recognized as a Rayleigh quotient.
- A standard result for a symmetric matrix such as $\mathbf{X}^T\mathbf{X}$ is that the quotient's maximum possible value is the largest eigenvalue of the matrix $\mathbf{X}^T\mathbf{X}$, which occurs when $\boldsymbol{w}$ is the corresponding eigenvector.

Feature selection

- **Not All Attributes Are Equal**
  The selection of attributes is critically important in performance.

- **Misleading**
  Redundant attributes can be misleading to modeling algorithms.
- **Overfitting**
  Keeping irrelevant attributes in your dataset can result in overfitting.
  This overfitting of the training data can negatively affect the predictive accuracy.

- It is important to remove redundant and irrelevant attributes from your dataset before evaluating algorithms.

Advantage of Feature Selection

- Feature Selection or attribute selection is a process by which you automatically search for the best subset of attributes in your dataset.
- The notion of “best” is relative to the problem to solve, but typically means highest accuracy.

Three key benefits of performing feature selection are:

- **Reduce Overfitting**: Less redundant data means less opportunity to make decisions based on noise.
- **Improve Accuracy**: Less misleading data means modeling accuracy improves.
- **Reduce Training Time**: Less data means that algorithms train faster.

Feature selection methods

1. Filter method: ranking features with some measure score like mutual information or Pearson correlation coefficient
2. Wrapper method : search a subset in the feature space

   greedy search

   Forward selection: start with empty, gradually add one of “strongest” features

   Backward selection: start with full set gradually eliminate “weakest” features

   Random selection.

   Exhaustive search is usually impossible in a large feature set
3. Embedded method: part of model construction (regression)

   such as Decision tree

Attribute Evaluator

The Attribute Evaluator finds a subset of attributes and evaluating its impact to the accuracy of the model:

- **CfsSubsetEval**: Values subsets that correlate highly with the class value and low correlation with each other.
- **ClassifierSubsetEval**: Assesses subsets using a predictive algorithm and another dataset that you specify.
- **Wrapper Subset Eval**: Assess subsets using a classifier that you specify and n-fold cross validation.

Search Method

- The Search Method is the is the structured way in which the search space of possible attribute subsets is navigated based on the subset evaluation.
- Baseline methods include Random Search and Exhaustive Search, although graph search algorithms are popular such as Best First Search.

Attribute evaluation search methods can be carried out as:

- **Exhaustive search**: Tests all combinations of attributes.
- **Best First search**: Uses a best-first search strategy to navigate attribute subsets.
- **Greedy StepWise search**: Uses a forward (additive) or backward (subtractive) step-wise strategy to navigate attribute subsets.

Feature transformation

- Feature transformation refers to some immediate function applied to a collection of data points.
- For instance, we want to change “positive” and “negative” values into “1” and “0”, respectively.
- The common functions f(x) used can be: sqaure root, square, reciprocal, logarithm, power, etc.

- A common example is where you transform categorical / nominal data types into binary or one-hot encoding.

One-hot encoding as feature transformation

- “from Europe”, “from US”, “from Asia”
  encode
- 00100, 01000, 01000 only one 1’s in each code

Feature transformation

- Scaling or normalizing features within a range, say between 0 to 1.
- Transforming categorical features to numerical.
- Principle Component Analysis (PCA) tries to reduce dimension while capturing most information present in the data.
- Random Projection: a way of dimension reduction.
- Linear discriminant analysis (LDA) tries to separate inter-class clusters,
- Sparse representation represents data using a sparse combination of elementary components.

Random projection

- In random projection, the original d-dimensional data is projected to a k-dimensional (k << d) subspace, using a
- k×d - dimensional random matrix R whose rows have unit lengths.
- $\tilde{x} = R x$

- The dimensions and distribution of random projection matrices are controlled so as to approximately preserve the pairwise distances between any two samples of the dataset.

Feature scaling

- **Feature scaling** is a method used to standardize the range of independent variables or features of data.
- In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

Feature rescaling

The simplest method is rescaling the range of features to scale the range in [0, 1] or [−1, 1].

Selecting the target range depends on the nature of the data.

The general formula is given as:

$$x' = (x - \min(x)) / (\max(x) - \min(x))$$

Standardization

The general method of calculation is to determine the distribution mean and standard deviation for each feature.

Next we subtract the mean from each feature.
Then we divide the values (mean is already subtracted) of each feature by its standard deviation.

$$x' = (x - x_m)/\sigma$$

Where x is the original feature vector, x_m is the mean of that feature vector, and σ is its standard deviation

Batch normalization

- Google: “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift “ (2015)

- Why batch normalization?
- batch normalization allows each layer of a network to learn by itself a little bit more independently of other layers.

- Previous experiments showed that deep learning with batch normalized performed better than without.

Unit vector

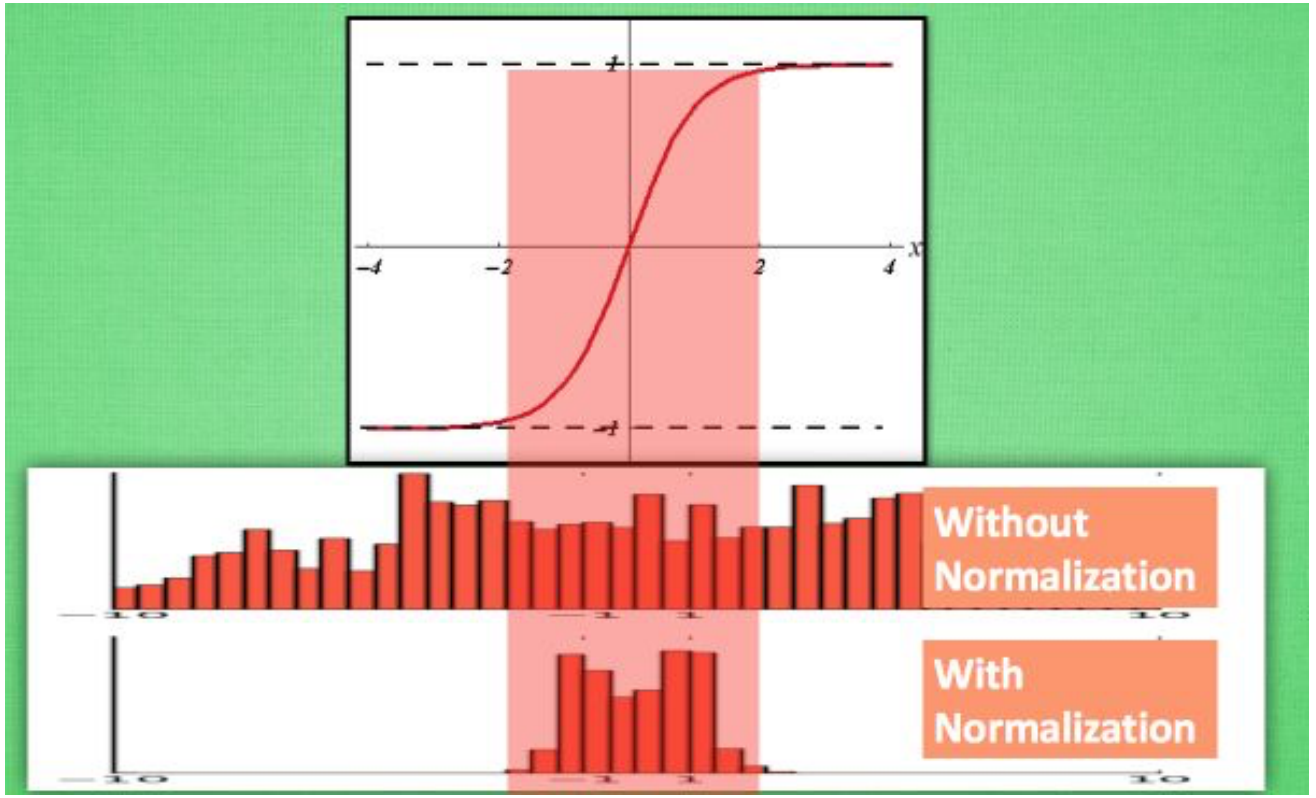
- X’= x/||x||

Batch normalization

- Batch normalization increases the stability of a neural network.
- Batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation.

- $z = g(Wu+b) \Rightarrow z = g(BN(Wu))$ [note: the bias is ignored]

With or without batch normalization



Batch normalization

- Reducing internal co-variance shift: change in the distribution of network activations due to the change in network parameters during training.
- network training converges faster
- if its inputs are whitened – i.e., linearly transformed to have zero means and unit variances, and decorrelated

- Batch Normalization allows us to use much higher learning rates and be less careful about initialization.

Batch normalization transform

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

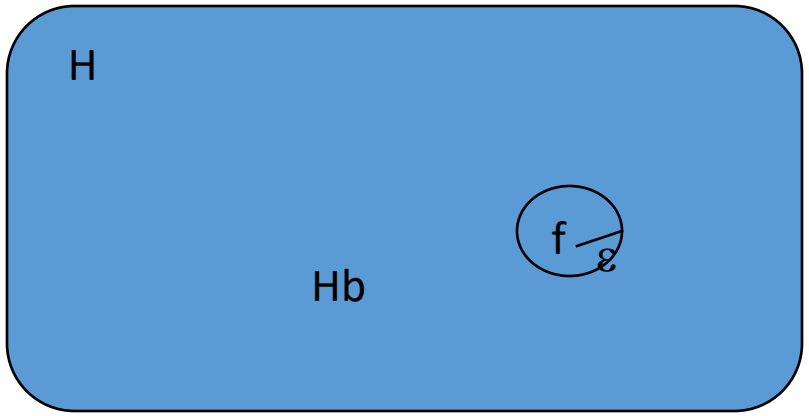
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Theory of Learnable

- Valiant [1984] proposed PAC-learnable (Probably Approximately Correct)
- Complexity of a learning problem
- # of training examples that is needed to learn a correct target concept to some accuracy 1- $\epsilon\varepsilon$ and at certain confidence (probability) 1- $\delta\delta$

- Pr(Pr(error) > $\in\varepsilon$) < $\delta\delta$ //Note: this is a math prob. statement of above statement

ε



- $H(1-\epsilon)^m \leq\leq \delta\ \delta$
  - H: total # of hypotheses
  - m: # of training examples
  - $\epsilon\ \varepsilon$: probabilistic error bound that a given hypothesis might commit to a training instance
  - $\delta\delta$: [1- δ] confidence that a hypothesis has been consistent with m training instances is correct

- m $\geq\geq$ ln(δ/H)/ln(1- ϵ) because ln (1 - $\epsilon\varepsilon$) $\leq -\varepsilon$
- $\therefore$ m $\geq\geq$ 1/ ϵ ln(H/ δ)

- Let's say if we are biased at the hypothesis space of n conjunctions of predicates. H = 3^n

- m $\geq\geq$ (n/ ϵ) ln(3/ δ δ)
- If n = 4, $\varepsilon\in$ = 0.01, $\delta\delta$=0.01

  then m $\geq\geq$ 400 ln(300) = 400*3.4 = 1360

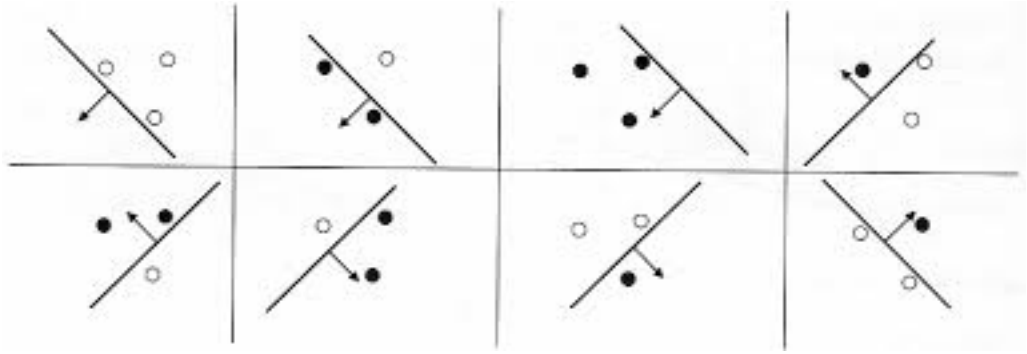
Vapnik-Chervonekis dimension (VC dimension)

- A property of a set of functions (hypothesis space H, learning machine, learner) {f(α)} (α is a generic set of parameters: a choice of α specifies a particular function)

- **Shattered**: If a given set of N points can be labeled in all possible 2^N ways, and for each labeling, a set of {f(α)} can be found which correctly assigns these labels, we say that the set of points are shattered by that set of functions.

- The functions f(α) are usually called hypotheses, and its space is called hypothesis space and denoted by H.
- The **VC dimension** for the set of functions {f(α)} i.e. the hypothesis space H is defined as the **maximum number of training points that can be shattered by H.**
- It is frequently used in determining the sample complexity in terms of mistake bound.

Example



- While it is possible to find a set of 3 points that can be shattered by the set of oriented lines, it is not possible to shatter a set of 4 points (with any labeling).
- Thus the VC dimension of the set of oriented lines in R^2 is 3.
- The VC dimension of the set of oriented hyperplane in R^n is n+1

Sample Complexity in terms of VC dimension (Blumer et. al.1989)

- A neural network with finite VC dimension h ≧1
- Any consistent learning algorithm for that neural network is a PAC learning algorithm
- There is a constant K such that a sufficient size of training set T for any such algorithm

$$N = \frac{K}{\varepsilon}(h \log(1/\varepsilon) + \log(1/\delta))$$

Expected Risk vs. Empirical Risk

- $R(\alpha) = \int \frac{1}{2} |y\text{-}f(x,\alpha)| dP(x,y)$
- $R_{emp}(\alpha) = \frac{1}{2N} \sum_{i=1}^{N} |y_i - f(x_i, \alpha)|$

- Under PAC (probably approximately correct) model, Vapnik shows that the bound for the expected risk which holds with probability 1 - η $(0 \leq \eta \leq 1)$,

- $\mathcal{R}(\alpha) \leq \mathcal{R}_{emp}(\alpha) + \sqrt{\frac{h\left(\log\left(\frac{2N}{h}\right)+1\right)-\log(\eta/4)}{N}}$

- Where h is VC dimension of $f(x, \alpha)$,

the second term on the right hand side is called VC confidence.

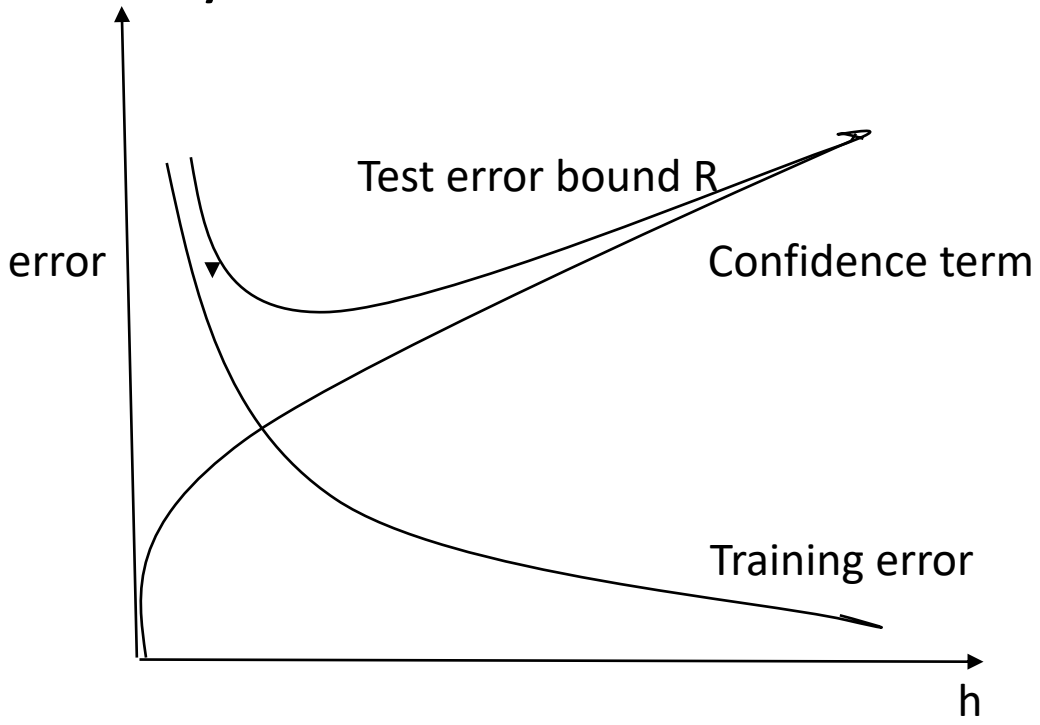
Implication of the bound for Expected Risk

- $\mathcal{R}(\alpha) \leq \mathcal{R}_{emp}(\alpha) + \sqrt{\frac{h\left(\log\left(\frac{2N}{h}\right)+1\right)-\log(\eta/4)}{N}}$

$= \mathcal{R}_{emp}(\alpha) + \sqrt{\frac{h(\eta)}{N}}$

Tradeoff between empirical error and confidence term with respect to VC-dimension

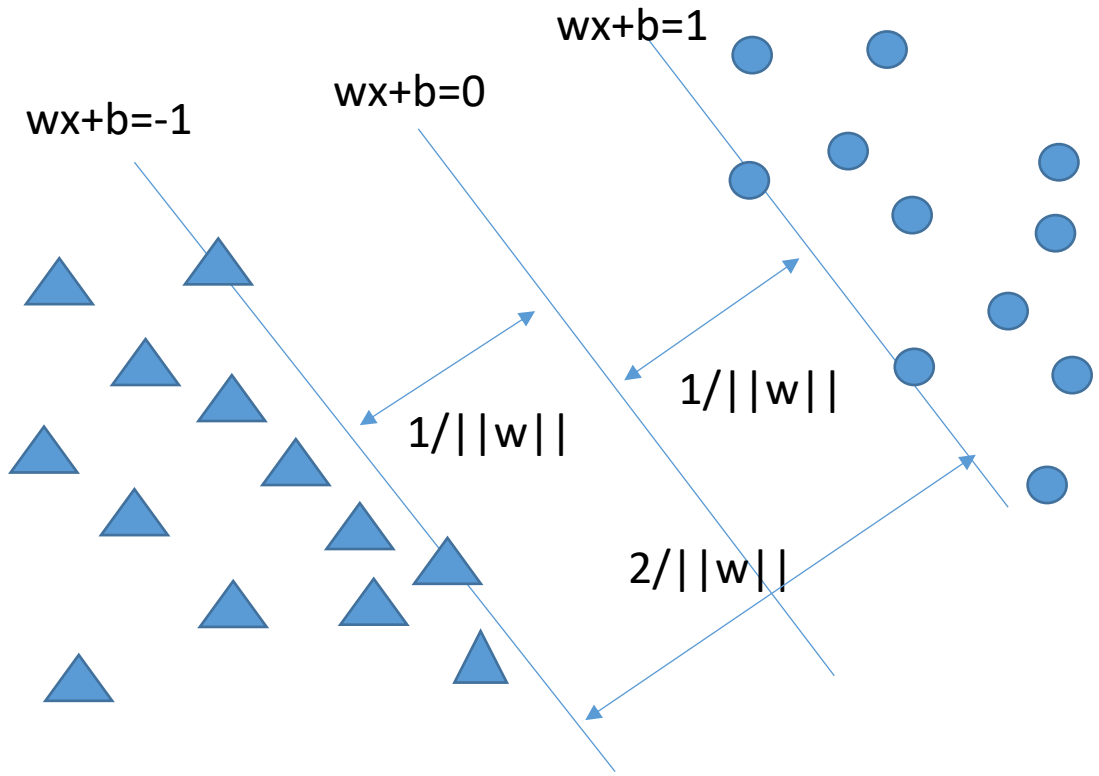
Test error bound may increase as VC-dimension is going up…

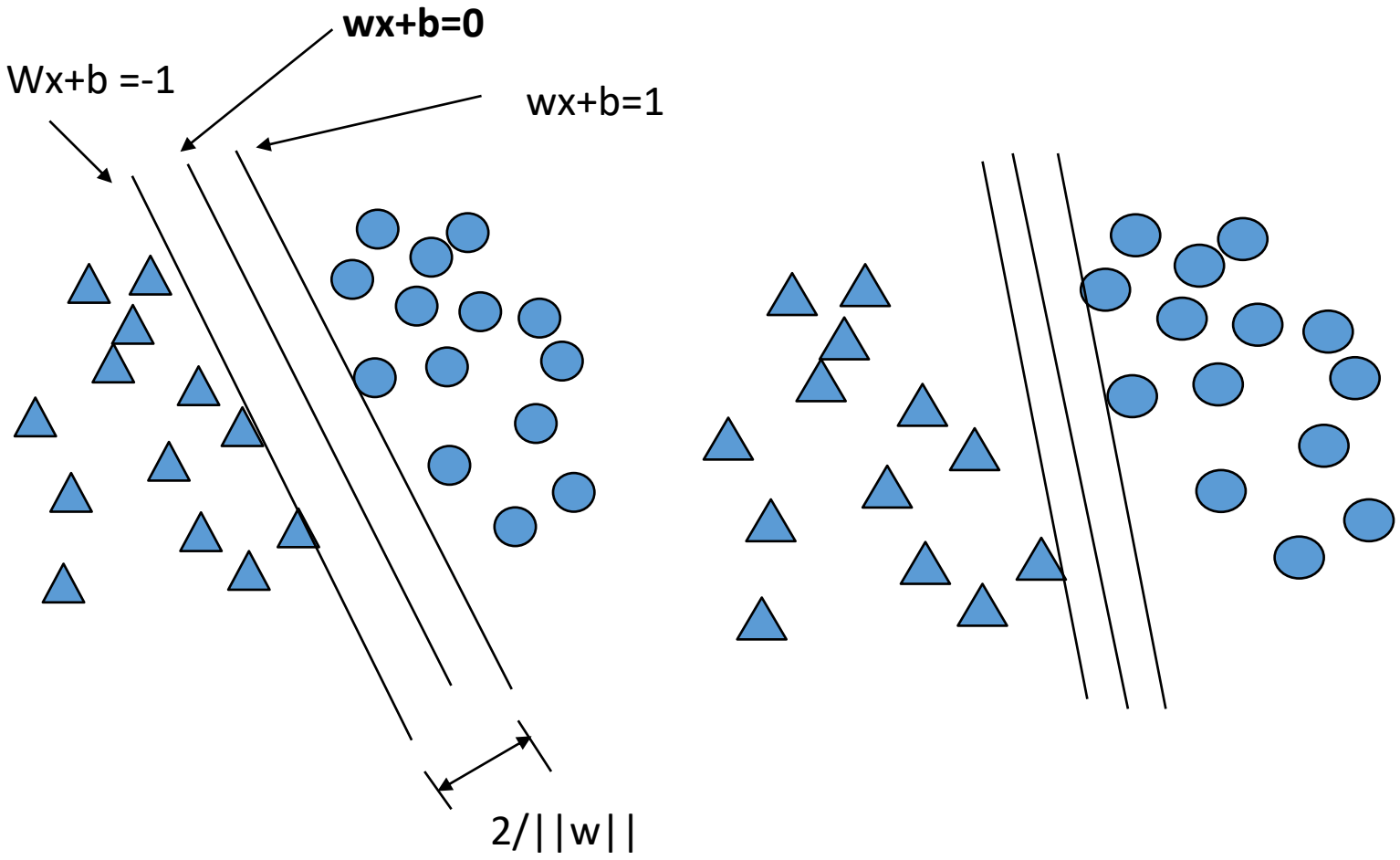


Tradeoff of choosing a hypothesis space for machine learning

- A learning machine with large hypothesis complexity (higher VC dimension) => small empirical risk (but might increase confidence bound)
- A simpler hypothesis learning machine (lower VC dimension) => low VC confidence (but can have a higher empirical error)

Hyperplane that separates two classes of data points




wx+b=0
Wx+b =-1
wx+b=1
2/||w||

Support Vector Machine

- Objective: To find an optimal hyperplane that correctly classifies data points as much as possible.
- Approach: formulate a constrained optimization problem, then solve it using constrained quadratic programming (constrained QP).
- Theorem: Structural Risk Minimization
- Issues: VC dimension, linear separability, feature space, multiple class
- svm

Support Vector Machines Classifier learning

- Find a hyperplane in terms of parameters w and b expressed as

  w x +b = 0 such that data in the two classes are separated “as much as possible” into two sides of the hyperplane.

- The two classes of data are separated by the hyperplane and centered between two boundary hyperplanes $|w x_i + b|=1$ for all training data x_i and the margin between two classes is $2/||w||$.
- So we want to maximize the margin $2/||w||$

VC-dimension and SVM

[Vapnik 95]

- Let R be the radius of the smallest ball

  $B_R(a)=\{ x \in F: ||x - a|| < R\}$ $(a \in F)$ *containing the points x1, x2, …,* x_N *and*

- *let* $f_{w,b} = \text{sgn}((w \cdot x)+b)$ be canonical hyperplane defined on these points.

Then the set $\{f_{w,b}: ||w|| \leq A$ has VC-dimension h satisfying $h < R^2A^2 + 1$

//note: this relates a hypothesis in terms of a hyperplane to its VC-dimension complexity

Minimizing ||W| = minimize VC dimension

- In other words,

Maximizing the margin 1/||w||

$\Longrightarrow$ minimizing ||W||

$\Longrightarrow$ smallest acceptable VC dimension

$\Longrightarrow$ smallest acceptable optimal hyperplane that minimize expected risk

- Question: How to find the optimal hyperplane (w,b)?

Linear Support Vector machine

- Given a set of points $x \in R^n$ with i = 1, 2,…,N.
- Each point x_i belongs to either of the two classes with the label $y_i = \{1,-1\}$
- The set S is linear separable if there exists $w \in R^n$ and

  b ∈R such that

  $$y_i (w\, x_i + b) \geq 1, \quad i = 1, \ldots, N$$

- Each point x_i to the hyperplane distance is

  $d_i = \frac{wx_i+b}{||w||}$ and

- All data points $x_i \in S$ ： $y_i\, d_i \geq 1/||w||$

- //note: we have N data points

- //note: this hyperplane is determned by w and b

Linear SVM

- The distance of any canonical separating hyperplane in terms of (w, b) pair to the closest point is 1/||w||
- The optimal separating hyperplane (OSH) is to maximize the distance 1/||w||

Using Constrained Quadratic programming to find OSH

- Minimize ½ w·w

  Subject to $y_i(w\,x_i + b) \geq 1$ i=1,…,N

- Using Lagrange multipliers α_i to convert constraints into objective:

  Lp = 1/2 w·w - $\sum_{i=1}^{N} \alpha_i(yi(wx_i + b) - 1)$

Why support vectors?

- Support vectors are those points that closest OSH, that are needed to determine OSH.
- The support vectors can be a linear combination of a relatively small percentage of points x_i; $w^* = \sum_{i=1}^{N} \alpha_i y_i x_i$
- A new data point can be classified by simply the sign of sgn($w * x + b$)

Solving constrained QP

- $\frac{\partial L}{\partial b} = \sum_{i=1}^{N} y_i \alpha_i$=0
- $\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \alpha_i y_i k_i = 0 \Longrightarrow$

$$w = \sum_{i=1}^{N} \alpha_i y_i k_i$$

where $\frac{\partial L}{\partial w} = (\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \ldots, \frac{\partial L}{\partial wn})$

Substitute into Lp, we obtain

$$L_D = \sum_{i=1}^{N} \alpha_i - 1/2 \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i \cdot xj$$

$$= \sum_{i=1}^{N} \alpha_i - 1/2 \sum_{i,j=1}^{N} \alpha_i D_{ij} \alpha_j$$

Solving constrained QP using Dual

- Maximize $\sum_{i=1}^{N} \alpha_i - 1/2 \alpha D \alpha$
- Subject to $\sum_{i=1}^{N} y_i \alpha_i = 0, \alpha \geq 0$

where D is an N x N matrix such that

$D_{ij} = y_i y_j \, x_i \cdot xj$

For the solution at the saddle point, (w*,b*),

it follows that $w^* = \sum_{i=1}^{N} \alpha^* y_i x_i$

- b can be determined from α^* which is a solution of the dual problem and from Kuhn-Tuck condition:

$$\alpha^*_i(y_i(w * x_i + b') - 1) = 0, i = 1, \dots, N$$

The original optimality problem becomes:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \Longrightarrow$$

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$y_i(w \cdot x_i + b) \geq 1, i = 1,2, \dots N$$

Soft Margin Classifier

- S is not linear separable, it can be generalized to N non-negative variables $(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N)$ such that

$$y_i(w \cdot x_i + b) \geq 1 - \varepsilon_i, \quad i = 1, \ldots, N$$

Generalized OSH

- Minimize 1/2w · w + C $\sum_{i=1}^{N} \varepsilon_i$
- subject to $y_i(w \cdot x_i + b) \geq 1 - \varepsilon_i, i = 1, \ldots, N$

  $\varepsilon_i \geq 0$
- large C => minimize the number of misclassified points
- small C => maximize the minimum distance 1/||w||

Non-linear support vector machine

- Mapping x from R^n to higher dimension R^m $\quad \phi: R^n \rightarrow R^m$

$x = (x_1, x_2) \rightarrow x' = (x_1^2, x_2^2, x_1 x_2)$

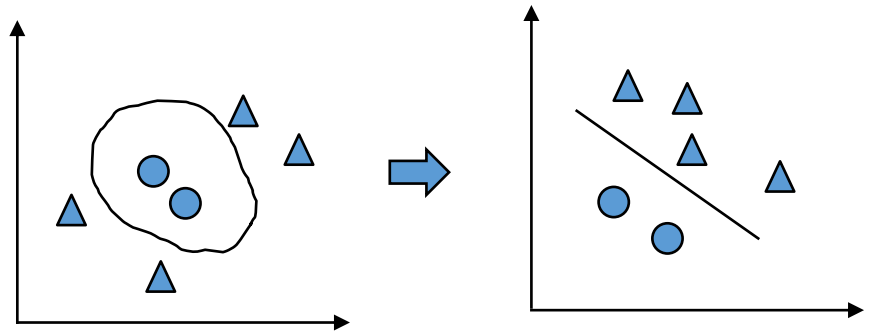
$f(x) = w \cdot x = \sum_{i=1}^{N} y_i \alpha_i x_i \cdot x + b$

$f(x) = w \cdot x = \sum_{i=1}^{N} y_i \alpha_i \phi(x_i) \cdot \phi(x) + b$

Kernel function: $K(x_i, x_j) = \phi(xi) \cdot \phi(xj)$

Transform input space

- An example of kernel function:
- $K(x_i, x_j) = e^{-||x_i - xj||^2/2\sigma^2}$



How do we choose a proper kernel function?

- Simple dot product: $K(x,y) = x \cdot y$
- Vovk's polynomial: $K(x,y) = (x \cdot y + 1)^p$
- Radial basis function (RBF): $K(x, y) = e^{-||x-y||^2/2\sigma^2}$
- Hyper tangent: $K(x,y) = \tanh(k_{xy} - \delta)$

S V M

Pro

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions to specify the decision function.

Con

- Requires positive and negative examples
- If the number of features is much greater than the number of samples, avoid over-fitting in choosing kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates.

Compared with the NN Backpropagation algorithm

- SVMs are currently slower than other neural networks because
  - There is no control over the number of data points selected by the learning algorithm for use as support vectors
  - There is no provision for incorporating prior knowledge about the task at hand into the design of the learning machine

Improved SVM

- Prior knowledge can be used to choose regularization term in the cost function.
- Use prior knowledge to generate virtual training examples to extract virtual support vectors, and then another support vector can be trained on the artificially enlarged training examples
- This can increase accuracy at a moderate time cost.

Active learning --
Can machines know what to learn?

- Assumptions:

1. Asking teachers to give correct labeling is costly.
2. We have already some training data that have labeling, while a lot are un-labeling.

How can we select limited size of unlabeled instances to ask so that it leads to better classification performance?

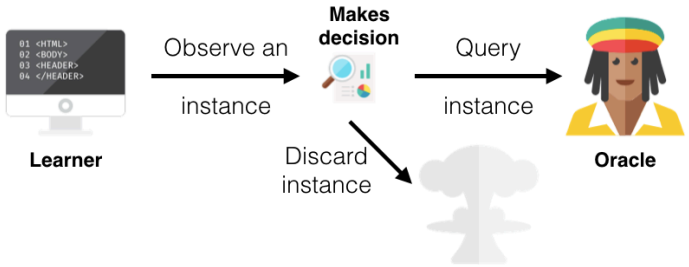
Active learning

- Semi-supervised learning
- Optimal sampling the “un-label” training examples

Three scenarios in active learning

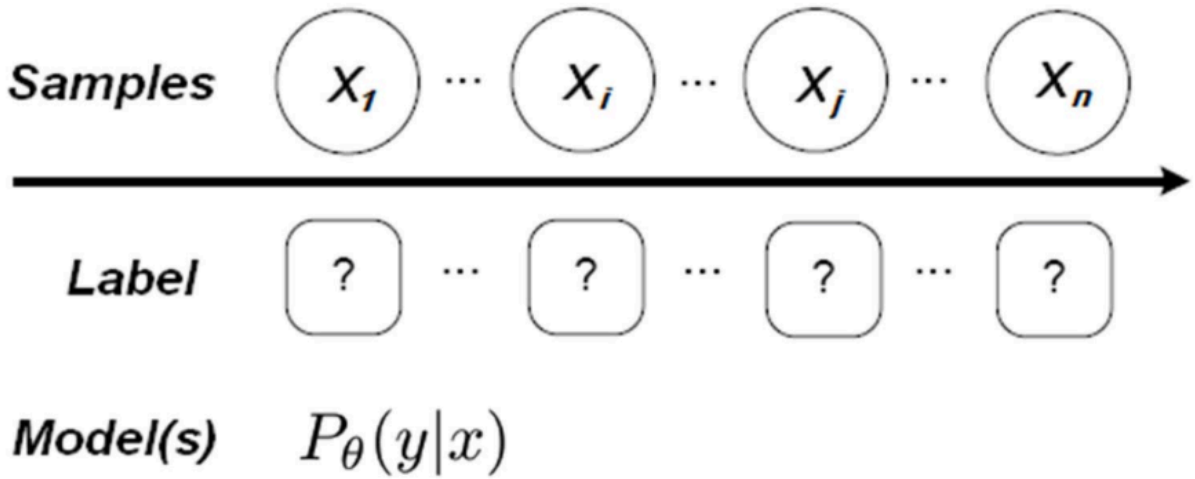
- **Membership Query Synthesis**

- **Stream-Based Selective Sampling**:



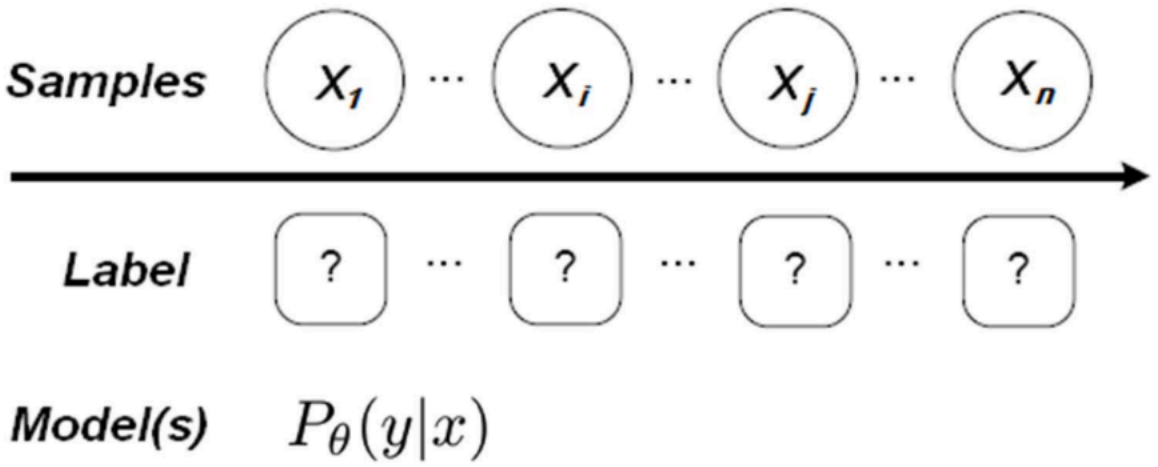
- **Pool-Based sampling:**

Pool-based active learning



- A pool of unlabeled samples
- A learned model (or a set of models)
- Choose: which sample to label next?

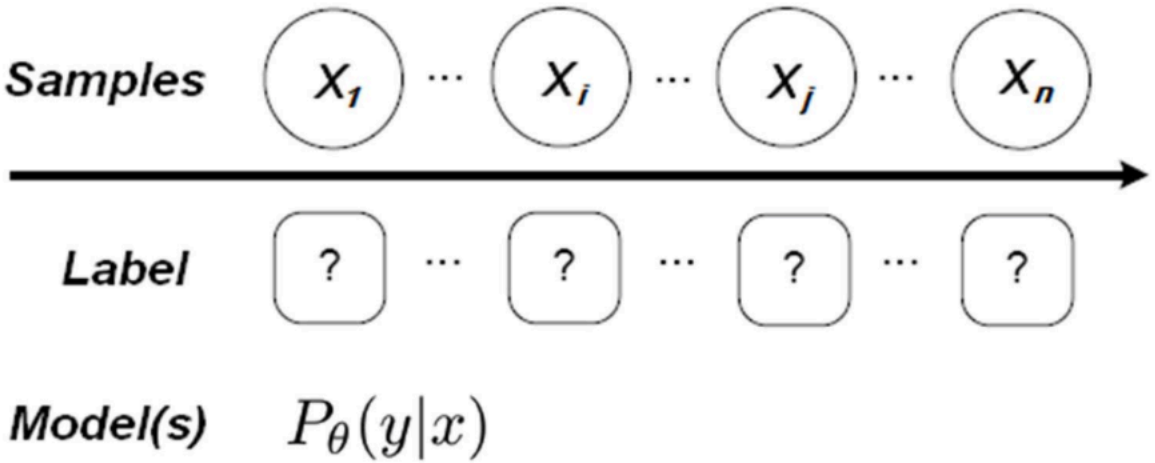
Uncertainty sampling



- Uncertainty sampling
  - Maximum entropy [Dagan & Engelson, ICML'95]

$$\phi_{ENT}(x) = -\sum_y P_\theta(y|x) \log_2 P_\theta(y|x)$$

Uncertainty sampling



- Uncertainty sampling
  - Smallest margin (between most likely and second most likely labels) [Scheffer et al., CAIDA'01]

$$\phi_M(x) = P_\theta(y_1^*|x) - P_\theta(y_2^*|x)$$

Active learning querying strategies

1. Uncertainty sampling
2. Query by committee
3. Expected error reduction EER
4. Variance reduction
5. Density weight reduction

Uncertainty sampling

The probability d_1 has label A, B and C is 0.9, 0.09 and 0.01 respectively and 0.2, 0.5 and 0.3 for d_2

Instances	Label A	Label B	Label C
d_1	0.9	0.09	0.01
d_2	0.2	0.5	0.3

Uncertainty sampling strategy

Choose x such that

- Least confidence: 1- P(y'|x)

  d1= 1- 0.9 = 0.1 (higher confidence)

  d2=1-0.5 = 0.5 (lower confidence)

  - Least confidence strategy => Select d2 (0.5) rather than d1 (0.9)
- Maximal Margin sampling: P(y'1|x) - P(y'2|x)

  d_1:the difference between its first and second most probable labels is

  0.81 i.e. (0.9 - 0.09)

  d_2: it is 0.2 i.e. (0.5 - 0.3)

  => Select d2
- Maximal Entropy: $- \Sigma_i P(y'_i|x) \log P(y'_i|x)$

  d_1:0.155

  d_2:0.447

  active learner => select d_2 .

Query by committee

- Maximal Vote entropy:

$$- \Sigma_y \frac{V(y, x)}{C} \log \frac{V(y, x)}{C}$$

- C is the size of committee , V(y, x) is member x vote for y.

Expected error reduction (EER)

- Roy and McCallum proposed the expected error reduction (EER), which is a popular active learning method.
- EER aimed to reduce the generalization error when labeling a new instance.
- The objective is to reduce the expected total number of incorrect predictions.
- But, a less stringent objective is to minimize the expected log-loss:

- minimize the expected 0/1-loss:

  $x^*_{0/1} = \text{argmin}_x \Sigma_i P_\theta(y_i|x) (\Sigma^U{}_{u=1} 1 - P_{\theta^+\langle x, y_i\rangle}(\hat{y}|x^{(u)}))$,
- $\theta^+\langle x,y_i\rangle$ refers to the the new model after it has been re-trained with the training tuple $\langle x, y_i\rangle$ added to L.
- $x^*_{log} = \text{argmin} \Sigma_i P_\theta(y_i|x) - \Sigma^U{}_{u=1} \Sigma_j P_{\theta+\langle x, y_i\rangle}(y_j|x^{(u)}) \log P_{\theta+\langle x,y_i\rangle}(y_j|x^{(u)})$

Expected Error reduction

L is the labeled training set, L^+ is the newer training set after the active learning selection instance is incorporated into the labeled set.

Reduce expected entropy over U.

$$\arg\min_{x\in\mathcal{U}} \sum_{y\in C} P_{\mathcal{L}}(y|x) \left(-\sum_{x_i\in\mathcal{U}} \sum_{y_i\in C} P_{\mathcal{L}+}(y_i|x_i) \log P_{\mathcal{L}+}(y_i|x_i) \right)$$

Variance Reduction

- Minimizing the expectation of a loss function directly is expensive, and in general this cannot be done in closed form.
- We can reduce generalization error *indirectly* by minimizing output variance, which sometimes does have a closed-form solution.
- learner's expected future error can be decomposed as [Geman et al. (1992)]
- $E_T[(\hat{y} - y)^2 | x] =$

$$E[(y - E[y|x])^2] + (E_L[\hat{y}] - E[y|x])^2 + E_L[(\hat{y} - E_L[\hat{y}])^2],$$

noise bias variance

- where $E_L[\cdot]$ is an expectation over the labeled set L, $E[\cdot]$ is an expectation over the conditional density $P(y|x)$, and E_T is an expectation over both.
- $\hat{y}$ is shorthand for the model's predicted output for a given instance x,
- while y indicates the true label for that instance.

Noise, bias, model variance

- *Noise is* the variance of the true label y given only x, which does not depend on the model or training data. Such noise may result from stochastic effects of the method used to obtain the labels, for example, or because the feature representation is inadequate.
- *bias*, represents the error due to the model class itself, e.g., if a linear model is used to learn a function that is only approximately linear. This component is invariant given a fixed model class.
- model's *variance* is the learner's squared-loss with respect to the target function.
- Minimizing the variance, then, is guaranteed to minimize the future generalization error of the model (since the learner itself can do nothing about the noise or bias components).

Variance reduction query selection

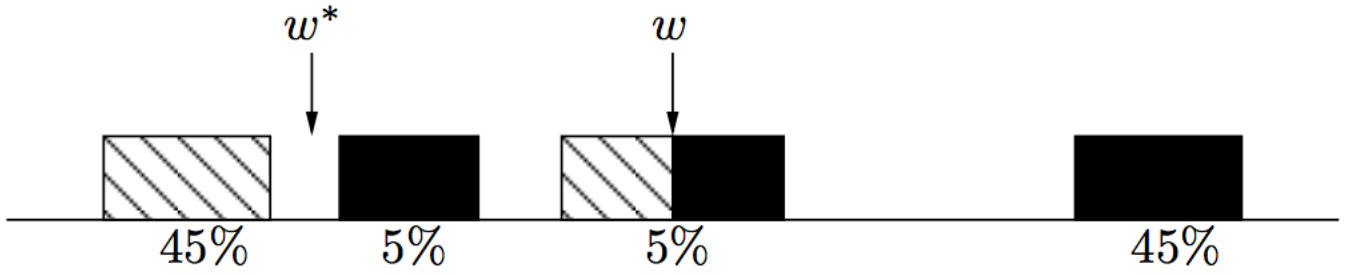
- Cohn (1994). The *variance reduction* query selection strategy then becomes:

$$x^*_{VR} = \text{argmin}_x \langle \tilde{\sigma}^2_{\hat{y}} \rangle^{+x}$$

- $\langle \tilde{\sigma}^2_{\hat{y}} \rangle^{+x}$ is the estimated mean output variance across the input distribution after the model has been re-trained on query x and its corresponding label.

Sampling bias (non linear separable case)

- The data lie in four groups on the line, and are distributed uniformly within each group. The two extremal groups contain 90% of the distribution.
- Solids have a + label, while stripes have a − label.
- Binary classifier at w has 5% error while w* has 2.5% error



Supervised vs semi-supervised

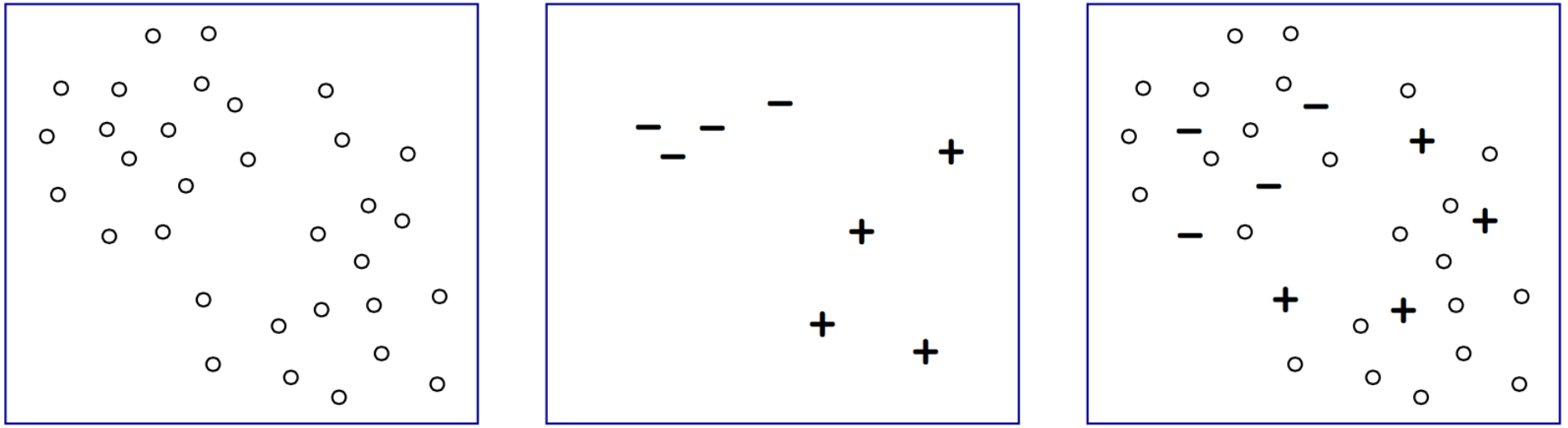


Figure 1: Each circle represents an unlabeled point, while + and − denote points of known label. *Left:* raw and cheap – a large reservoir of unlabeled data. *Middle:* supervised learning picks a few points to label and ignores the rest. *Right:* Semisupervised and active learning get more use out of the unlabeled pool, by using them to constrain the choice of classifier, or by choosing informative points to label.

Select the boundary point to query