SESSION 17: STATISTICAL MACHINE LEARNING (VII)

Gang Li

Deakin University, Australia

2021-10-16

Table of Content

Regularized Loss Minimization (RLM)

Regularized Loss Minimization (RLM) is a learning paradigm in which we jointly minimize the empirical risk and a regularization function, which is a mapping $R : \mathscr{R}^d \mapsto \mathscr{R}$, the regularized loss minimization rule outputs a hypothesis which

$$\underset{\omega}{\operatorname{argmin}}(L_S(\omega) + R(\omega))$$

Tikhonov regularization is one popular regularization function: $R(\omega) = \lambda\|\omega\|^2$, where $\lambda > 0$ is a scalar, and the norm is the l_2 norm.

Notes.

- It is similar to SRM and MDL paradigm:

  RLM and MDL The "prior belief" of biasing to "short" vector in the $\mathscr{H}$.
  RLM and SRM We can define a sequence of hypothesis classes, $\mathscr{H}_1 \subset \mathscr{H}_2 \subset \mathscr{H}_3 \ldots$, where $\mathscr{H}_i = \{\omega : \|\omega\| \le i\}$. If the sample complexity of each $\mathscr{H}_i$ depends on i, then the RLM is similar to SRM for this sequence of nested classes.
- **Stabilizer**: Tikhonov regularization makes the learner stable w.r.t. small perturbation of the training set, which in turn leads to better generalization.

□

Stability

Given a training set $S = (z_1, \cdots, z_{i-1}, z_i, z_{i+1}, \cdots, z_m)$ and an additional example z', let $S^{(i)}$ be the training set obtained by replacing $z_i \in S$ by z', namely $S^{(i)} = (z_1, \cdots, z_{i-1}, z', z_{i+1}, \cdots, z_m)$ and let $U(m)$ be the uniform distribution over $[m]$. Let $\epsilon : \mathcal{N} \mapsto \mathcal{R}$ be a monotonically decreasing function. We say that a learning algorithm A is On-Average-Replace-One-Stable with rate $\epsilon(m)$ if every distribution $\mathcal{D}$:

$$\underset{(S,z') \sim \mathcal{D}^{m+1}, i \sim U(m)}{\mathrm{E}} [l(A(S^{(i)}, z_i)) - l(A(S), z_i)] \le \epsilon(m)$$

Notes.

- Informally: an algorithm A is stable if a small change of its input S will lead to a small change of its output hypothesis.
- Need to specify what is "small change of input" and what is "small change of output".

□

Stable Rules Do Not Overfit

if A is *on-average-replace-one-stable* with rate $\epsilon(m)$ then

$$\underset{S \sim \mathcal{D}^m}{\mathrm{E}} [L_{\mathcal{D}}(A(S)) - L_S(A(S))] \le \epsilon(m)$$

Proof.

- Since S and z' are both drawn i.i.d. from $\mathcal{D}$, we have that for every i

$$\underset{S}{\mathrm{E}}[L_{\mathcal{D}}(A(S))] = \underset{(S,z')}{\mathrm{E}} [l(A(S), z')] = \underset{(S,z')}{\mathrm{E}} [l(A(S^{(i)}), z_i)]$$

- On the other hand, we can write

$$\underset{S}{\mathrm{E}}[L_S(A(S))] = \underset{(S),i}{\mathrm{E}} [l(A(S), z_i)]$$

- The proof follows from the definition of *stability*.

□

Tikhonov Regularization as Stabilizer

Assume that the loss function is convex and ρ-Lipschitz. Then, the RLM rule with the regularizer $\lambda\|\omega\|^2$ is on-average-replace-one-stable with rate $\frac{2\rho^2}{\lambda m}$. It follows that:

$$\mathop{\mathrm{E}}_{S\sim\mathscr{D}^m}[L_{\mathscr{D}}(A(S)) - L_S(A(S))] \leq \frac{2\rho^2}{\lambda m}$$

Similarly, for convex, β-smooth, and non-negative, the loss rate is $\frac{48\beta C}{\lambda m}$, with C is the upper bound on $\max_z l(\vec{0}, z)$, where $\omega = \vec{0}$.

Notes.

- The proof relies on the notion of strong convexity and is omitted here.

□

The Fitting-Stability Trade-off

The expected risk of a learning algorithm A can be rewritten as

$$\mathop{\mathrm{E}}_{S}[L_{\mathscr{D}}(A(S))] = \mathop{\mathrm{E}}_{S}[L_S(A(S))] + \mathop{\mathrm{E}}_{S}[L_{\mathscr{D}}(A(S)) - L_S(A(S))]$$

Notes.

- The first term is how good A fits the training set.
- The second term is the overfitting, and is bounded by the stability of A.
- λ controls the trade-off between above two terms.

□

Notes.

- Let A be the RLM rule.
- We saw (for convex-Lipschitz losses) $\mathrm{E}_{S\sim\mathscr{D}^m}[L_{\mathscr{D}}(A(S)) - L_S(A(S))] \leq \frac{2\rho^2}{\lambda m}$
- Fix some arbitrary vector ω^*, then $L_S(A(S)) \leq L_S(A(S)) + \lambda\|A(S)\|^2 \leq L_S(\omega^*) + \lambda\|\omega^*\|^2$.
- Taking expectation of both sides with respect to S and noting that $E_S[L_S(\omega^*)] = L_{\mathscr{D}}(\omega^*)$, we obtain that $E[L_S(A(S))] \leq L_{\mathscr{D}}(\omega^*) + \lambda\|\omega^*\|^2$.
- Therefore,

$$E[L_{\mathscr{D}}(A(S))] \leq L_{\mathscr{D}}(\omega^*) + \lambda\|\omega^*\|^2 + \frac{2\rho^2}{\lambda m}$$

- The stability term decreases as λ increases, and the empirical risk increases with λ. So a trade-off is needed.

□

The Regularization Path

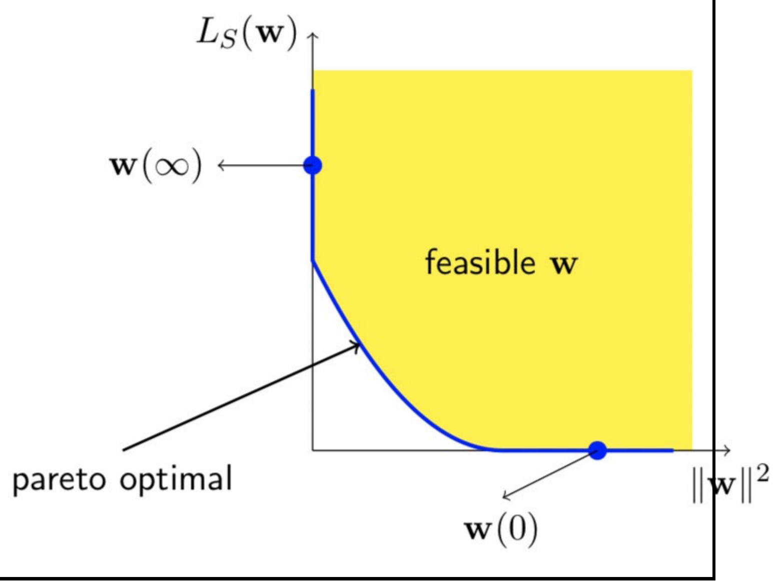
The RLM rule as a function of λ is $\omega(\lambda) = \text{argmin}_\omega L_S(\omega) + \lambda\|\omega\|^2$. It can be seen as a *Pareto* objective: minimize both $L_S(\omega)$ and $\|\omega\|^2$.

How to choose λ.

Bound minimization choose λ according to the bound on $L_{\mathscr{D}}(\omega)$ usually far from optimal as the bound is the worst case.

Validation calculate several *Pareto* optimal points on the regularization path (by varying λ) and use validation set to choose the best one.

□



Dimension vs. Norm Bounds

The expected risk of a learning algorithm A can be rewritten as

$$E[L_{\mathscr{D}}(A(S))] \le L_{\mathscr{D}}(\omega^*) + \lambda\|\omega^*\|^2 + \frac{2\rho^2}{\lambda m}$$

Notes.

- Previously in the course, when we learned d parameters, the *sample complexity* grew with d.
- Here, we learn d parameters but the *sample complexity* depends on the norm of $\|\omega\|$ and on the Lipschitzness/smoothness, rather than on d.
- Which approach is better depends on the properties of the distribution.

□

Binary Classification

Consider a binary classification problem:

Hypothesis $\mathscr{H}$ the function set as:

$$h(x) = \begin{cases} 1 & \text{when } f(x) > 0 \\ -1 & \text{when } f(x) < 0 \end{cases}$$

Loss Function The number of times h get incorrect results on the sample.

$$L(h(x), y) = \sum_{i=1}^{n} l^{0-1}(h(x_i) \neq y_i) \approx \sum_{i=1}^{n} l(f(x_i), y_i)$$

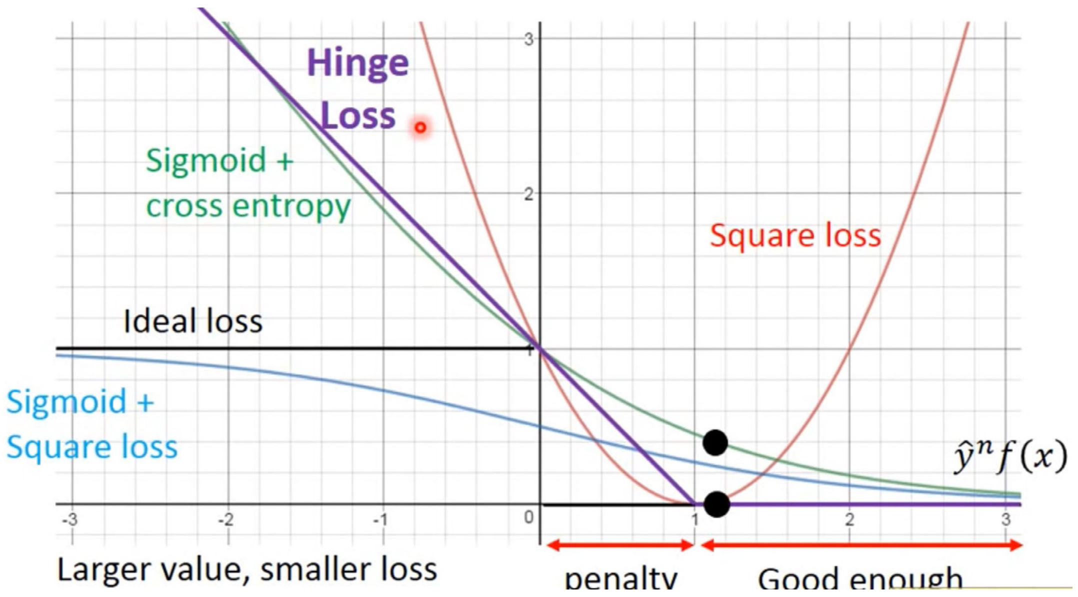
Training by Optimization Gradient descent is possible if both $h(x)$ and $f(x)$ are differentiable, otherwise difficult.

Loss Function

How to choose a differentiable function to approximate l^{0-1}?

Loss Function The number of times h get incorrect results on the sample.

$$L(h(x), y) = \sum_{i=1}^{n} l^{0-1}(h(x_i) \neq y_i) \approx \sum_{i=1}^{n} l(f(x_i), y_i)$$



Squared Loss.

$$l(f(x_i), y_i) = (y_i f(x_i) - 1)^2$$

- Intuitively, it wants to achieve:

$$f(x_i) = \begin{cases} 1 & \text{when } y_i = 1 \\ -1 & \text{when } y_i = -1 \end{cases}$$

- It penalizes the very correct examples where $y_i f(x_i) \gg 1$

□

Sigmoid + Squared Loss.

$$l(f(x_i), y_i) = \sigma(y_i f(x_i)) - 1)^2 = \begin{cases} \sigma(f(x_i)) - 1)^2 & \text{when } y_i = 1 \\ \sigma(f(x_i))^2 & \text{when } y_i = -1 \end{cases}$$

- It serves the purpose by achieving

$$\sigma(y_i f(x_i)) = \begin{cases} 1 & \text{when } y_i = 1 \\ 0 & \text{when } y_i = -1 \end{cases}$$

□

Sigmoid + Cross Entropy Loss.

$$l(f(x_i), y_i) = \ln(1 + e^{-y_i f(x_i)})$$

- It achieve the cross entropy between two Bernoulli distributions: $(y_i, 1 - y_i)$ and $\sigma(f(x_i)), 1 - \sigma(f(x_i))$. Here we divide it by $\ln 2$ so that it is a surrogate loss function for l^{0-1}.
- It serves the purpose.

□

Hinge Loss.

$$l(f(x_i), y_i) = \max(0, 1 - y_i f(x_i))$$

- It achieves

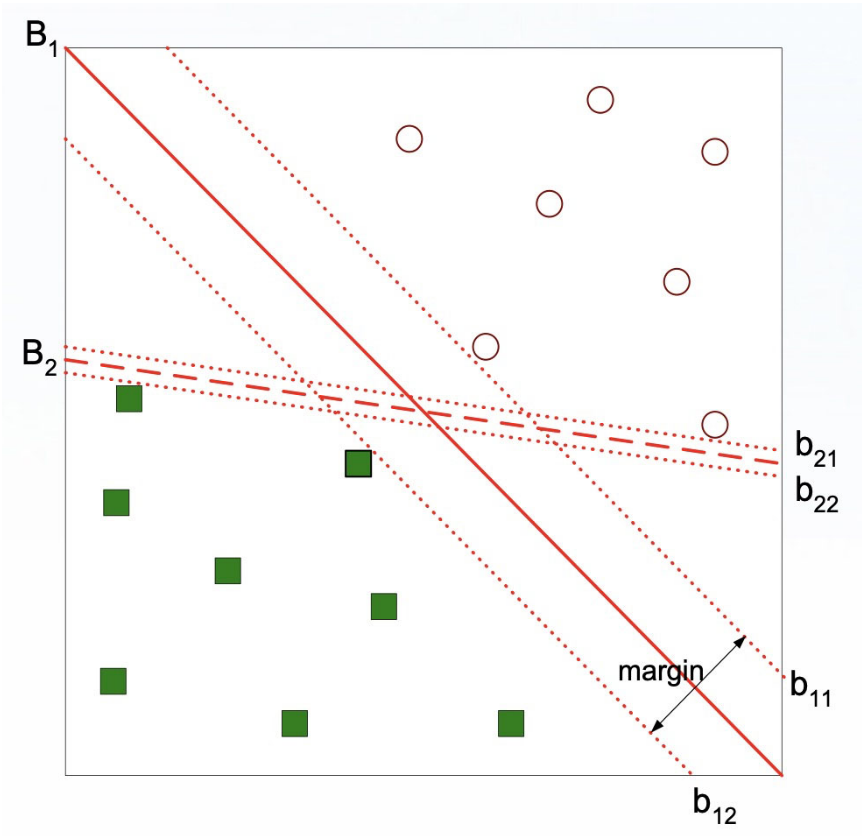
$$f(x_i) = \begin{cases} \geq 1 & \text{when } y_i = 1 \\ \leq -1 & \text{when } y_i = -1 \end{cases}$$

- It serves the purpose but different from *Sigmoid + Cross Entropy* loss.

□

Margin

Which separating hyperplane is better?



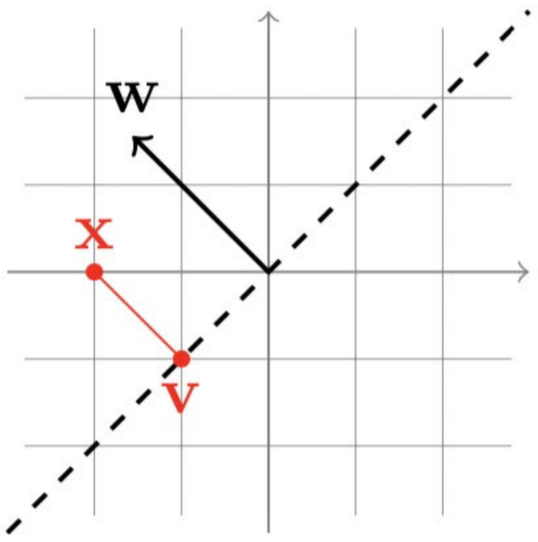
- Intuitively, solid red line is better.

Margin

Given hyperplane defined by $L = \{v : \langle \omega, v\rangle + b = 0\}$ and give a point x, the distance of x to L is

$$d(x, L) = \min \|x - v\| : v \in L$$

- If $\|\omega\| = 1$, then $d(x, L) = \|\langle \omega, x\rangle + b\|$



- Proof can be done easily.
- Some observation on the inner product: $\langle \omega, x\rangle = \|\omega\| \cdot \|x\| \cdot cos(\theta)$

Support Vector Machine (Hard-SVM)

When the sample is linearly separable, we seek for the separating hyperplane with largest margin $\mathrm{argmax}_{(\omega,b):\|\omega\|=1} \min_{i\in m} \|\langle\omega,x_i\rangle+b\|$, subject to $\forall i,\ y_i(\langle\omega,x_i\rangle+b)>0$.

- equivalent to $\mathrm{argmax}_{(\omega,b):\|\omega\|=1} \min_{i\in m} y_i(\langle\omega,x_i\rangle+b)$
- equivalent to $(\omega_0,b_0)=\mathrm{argmin}_{(\omega,b)}\|\omega\|^2$ subject to $\forall i,\ y_i(\langle\omega,x_i\rangle+b)\geq 1$.
- the margin of $(\frac{\omega_0}{\|\omega_0\|},\frac{b_0}{\|\omega_0\|})$ is $\frac{1}{\|\omega_0\|}$, and it is the maximal margin.

Notes.

Margin is Scale Sensitive The margin depends on the scale of the examples

- if (ω,b) separates $(x_1,y_1),\ldots,(x_m,y_m)$ with margin γ, then it separates $(2x_1,y_1),\ldots,(2x_m,y_m)$ with a margin of 2γ

Margin of distribution We say that $\mathscr{D}$ is separable with a (γ,ρ)-margin if exists (ω^*,b^*) s.t. $\|\omega^*\|=1$ and $\mathscr{D}(\{(x,y):\|x\|\leq\rho\bigwedge y(\langle\omega^*,x\rangle+b^*)\geq\gamma\})=1$

- then its sample complexity is $m(\epsilon,\delta)\leq\frac{8}{\epsilon^2}2(\rho/\gamma)^2+\log(2/\delta)$
- unlike the VC bounds, here the sample complexity depends on ρ/γ rather than d.

□

Support Vector Machine (Soft-SVM)

What if the sample is not linearly separable, we seek for the separating hyperplane with slack variable ϵ_n, minimizing the loss function L with RLM:

$$\underset{(\omega,b)\,:\|\omega\|=1}{\mathrm{argmin}}\ L(\omega,S)=\sum_{i=1}^{m} l(f(x_i),y_i)+\lambda\|\omega\|^2=\sum_{i=1}^{m}\epsilon_i+\lambda\|\omega\|^2$$

where $\epsilon_i=l^{hinge}(f(x_i),y_i)=\max(0,1-y_if(x_i))$.

- the constraints are equivalent to $\epsilon_i\geq\{\begin{matrix}0\\1-y_if(x_i)\end{matrix}$
- this second one is equivalent to $y_if(x_i)\geq 1-\epsilon_i$

Notes.

- This is the popular SVM formulation, which can be solved by *quadratic programming*.
- As an optimization problem, it can also be solved by *gradient descendant*.

□

Support Vector Machine (Soft-SVM): Gradient Descendant

What if the sample is not linearly separable, we seek for the separating hyperplane with slack variable ϵ_n, minimizing the loss function L with RLM:

$$\underset{(\omega,b):\|\omega\|=1}{\operatorname{argmin}} L(\omega,S) = \sum_{i=1}^{m} l(f(x_i),y_i) + \lambda\|\omega\|^2$$

Gradient Descendant: $f(x_i) = \omega^T x_i$.

- Take partial derivatives to each component ω_j:

$$\frac{\partial L(f(x_i),y_i)}{\partial \omega_j} = \sum \frac{\partial l(f(x_i),y_i)}{\partial \omega_j} = \sum \frac{\partial l(f(x_i),y_i)}{f(x_i)} \frac{\partial f(x_i)}{\partial \omega_j}$$

  Here we ignore the regularization term for simplicity.
  - $\frac{\partial l(f(x_i),y_i)}{f(x_i)} = \begin{cases} -y_i & \text{if } y_i f(x_i) < 1 \\ 0 & \text{otherwise} \end{cases}$ while $\frac{\partial f(x_i)}{\partial \omega_j} = (x_j)_i$
  - So $\frac{\partial L(f(x_i),y_i)}{\partial \omega_j} = \sum I(y_i f(x_i) < 1)(-y_i)x_i = \sum C_i(\omega_j)(x_j)_i$
  - For ω_j, the gradient descendant updating rule is $\omega_j = \omega_j - \eta \sum C_i(\omega_j)(x_j)_i$, in the vector form: $\omega = \omega - \eta \sum C_i(\omega) x_i$.

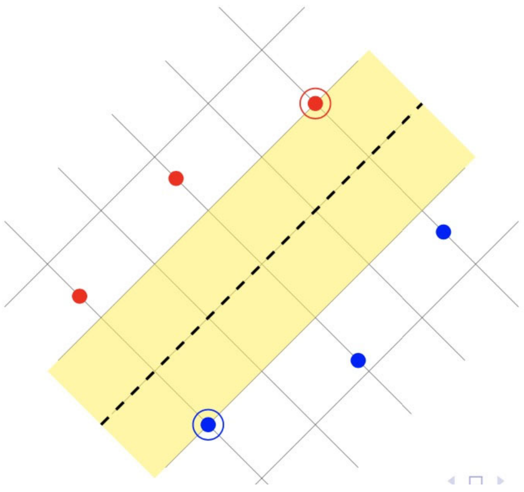
□

Support Vectors

A separating hyperplane is defined by (ω,b) subject to: $\forall i,\ y_i(\langle\omega,x_i\rangle + b) > 0$. The margin of a separating hyperplane is the distance of the closest example to it:

$$\min_i \|\langle\omega,x_i\rangle + b\|$$

Those closest examples are called support vectors.



- From the gradient descendant method, we can see that when $\omega = \vec{0}$, $\omega^* = \sum_i \alpha_i^* x_i$ is a linear combination of examples.
- α^* may be sparse, and those x_i with non-zero α_i^* are support vectors.
  - For *Hinge loss*, α^* is usually sparse.
  - For *logistic regression* or *cross entropy*, α^* is usually non-zero.

Representer Theorem

> Assume that ψ is a mapping from $\mathscr{X}$ to a Hilbert space (a feature space), then the SVM optimization is an instance of the following problem:
>
> $$\underset{\omega}{\operatorname{argmin}}(f(\langle\omega,\psi(x_1)\rangle,\ldots,\langle\omega,\psi(x_m)\rangle)+R(\|\omega\|))$$
>
> where $f:\mathscr{R}^m \mapsto \mathscr{R}$ is an arbitrary function. $R:\mathscr{R}_+ \mapsto \mathscr{R}$ is a monotonically no-decreasing function, such as $\lambda\|\omega\|^2$. Then $\exists\alpha\in\mathscr{R}^m$ such that $\omega^* = \sum_{i=1}^m \alpha_i\psi(x_i)$.

Intuition.

- Because ω^* is an element of a Hilbert space, so $\omega^* = \sum_{i=1}^m \alpha_i\psi(x_i)+u$, where $u \perp \psi(x_i)\ \forall x_i$.
- Set $\omega = \omega^* - u$, observe that $\omega^* = \omega + u$, we have $\|\omega^*\|^2 = \|\omega\|^2 + u^2$ and $\forall i\ \langle\omega,\psi(x_i)\rangle = \langle\omega^*,\psi(x_i)\rangle$.
- Hence the objective at ω equals the objective at ω^* minus $\lambda\alpha$. By optimality of ω^*, u must be zero.

□

Implications.

- By representer theorem, the optimal solution can be written as $\omega^* = \sum_{i=1}^m \alpha_i\psi(x_i)$
- Denote by $\mathscr{G}$ the Gram matrix s.t. $\mathscr{G}_{i,j} = \langle\psi(x_i),\psi(x_j)\rangle$, we have:

$$\langle\omega,\psi(x_i)\rangle = \langle\sum_{i=1}^m \alpha_i\psi(x_i),\psi(x_i)\rangle = \sum_{i=1}^m \alpha_i\langle\psi(x_i),\psi(x_i)\rangle = (\mathscr{G}\alpha)_i, \quad \forall i$$

- Also $\|\omega\|^2 = \alpha^T\mathscr{G}\alpha$. Hence, the optimisation task can be written as:

$$\underset{\alpha\in\mathscr{R}^m}{\operatorname{argmin}}(f(\mathscr{G}\alpha)+\lambda\alpha^T\mathscr{G}\alpha)$$

□

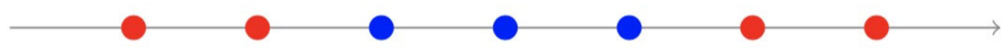
Kernel Trick

Embeddings into feature spaces

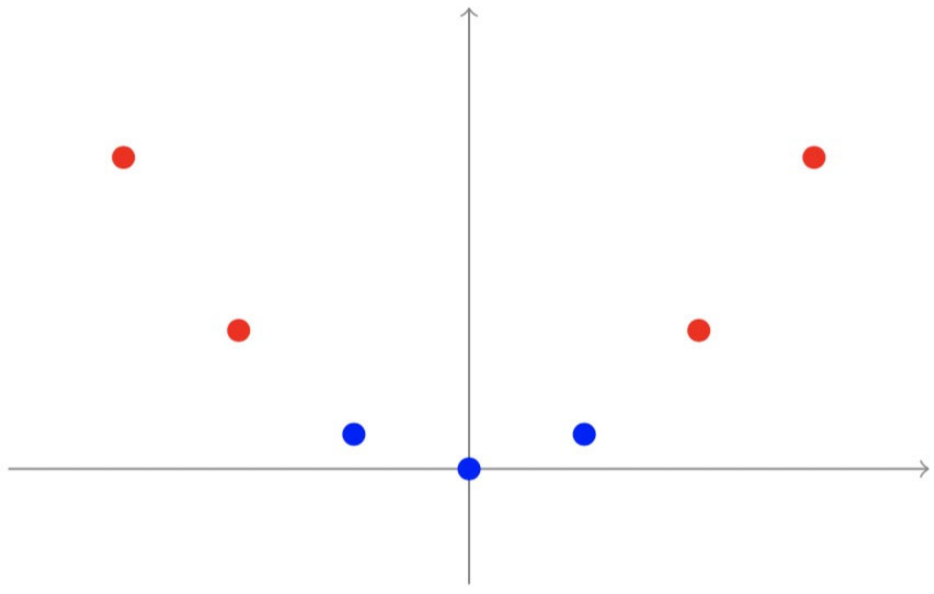
> What if the sample S is not linear separable?

Notes.

- The following sample in $\mathscr{R}^1$ is not separable by half-spaces



- It is separable in $\mathscr{R}^2$ by half-spaces if we may $x \mapsto (x, x^2)$



□

Embeddings into feature spaces

Define a mapping function $\psi : \mathcal{X} \mapsto \mathcal{F}$, where the feature space $\mathcal{F}$ is a subset of Hilbert space. Then the training of half-space is done over

$$\{(\psi(x_1), y_1), \ldots, (\psi(x_m), y_m)\}$$

Notes.

- How to choose ψ?
  - In general, this requires prior knowledge.
  - There are some generic mappings that enrich the class of half-spaces, e.g. polynomial mappings.
- If F is high dimensional we face

  statistical challenge can be tackled using margin
  computational challenge can be tackled using kernels

□

Dual Representation of Hypothesis

We know that $\omega^* = \sum_i \alpha_i^* x_i = \vec{x}\alpha^*$ is a linear combination of x_i. Accordingly, the hypothesis $h(x)$ can be written as

$$h(x) = \omega \cdot \vec{x} = \vec{x}(\alpha\vec{x})^T = \alpha^T\vec{x}^T\vec{x} = \sum_i \alpha_i(x_i \cdot x_i) = \sum_i \alpha_i \mathcal{K}(x_i \cdot x_i)$$

Notes.

- In this representation, the training of a hypothesis is equivalent to find $\vec{\alpha}^* = \{\alpha_1, \ldots, \alpha_m\}$, minimizing the loss function L.

  $$L = \sum_{i=1}^{m} l(\sum_{j=1}^{m} \alpha_i \mathcal{K}(\vec{x}_j, \vec{x}_i), y_i)$$

- The Kernel Trick $\mathcal{K}(\vec{x}_j, \vec{x}_i)$:
  - We don't really need to know vectors $\vec{x}_j$ and $\vec{x}_i$.
  - We only need to know the inner product.
- In the mapped feature space, it is then $\mathcal{K}(x_j, x_i) = \langle\psi(\vec{x}_j), \psi(\vec{x}_i)\rangle$.

□

Kernel Trick

> A kernel function for a mapping ψ is a function that implements inner product in the feature space, namely,
>
> $\mathcal{K}(x,y) = \langle\psi(x),\psi(y)\rangle$

Polynomial Kernel. The k degree polynomial kernel is defined to be

$$\mathcal{K}(x,y) = (1 + \langle\psi(x),\psi(y)\rangle)^k$$

- Since ψ contains all the monomials up to degree k, a half space over the range of ψ corresponds to a polynomial predictor of degree k over the original space.
- Observe that calculating $\mathcal{K}(x,y)$ takes $O(n)$ time while the dimension of $\psi(x)$ is nk
- Consider mapping from two dimensional space to three dimensional space: $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and $\psi(x) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$
- $\mathcal{K}(x,y) =<\psi(x),\psi(y)>= \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix} \cdot \begin{bmatrix} y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{bmatrix} = x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 = (x\cdot y)^2$

□

Gaussian kernel or "Radial Basis Function (RBF)" kernel. It is defined to be

$$\mathcal{K}(x,y) = e^{-\frac{\|x-y\|^2}{2\delta}}$$

- Let the original instance space be R and consider the mapping ρ where for each non-negative integer $n \geq 0$ there exists an element $\psi_n(x)$ which equals to $\frac{1}{\sqrt{n!}}e^{-\frac{x^2}{2\delta}}x^n$.
- ψ can have infinite dimension:

$$\mathcal{K}(x,y) = \langle\psi(x),\psi(y)\rangle = \sum_{i=1}^{n} \frac{1}{\sqrt{n!}} e^{-\frac{x^2}{2\delta}} y^n \frac{1}{\sqrt{n!}} e^{-\frac{y^2}{2\delta}} y^n = e^{-\frac{\|x-y\|^2}{2\delta}}$$

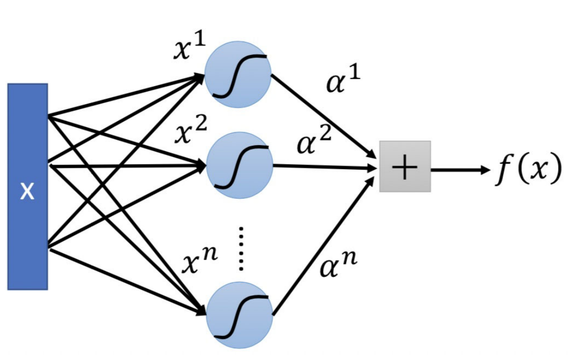
- It can learn any polynomial function.

□

Sigmoid kernel. It is defined to be

$$\mathcal{K}(x,y) = \tanh(\langle x,y\rangle)$$

- When using the sigmoid kernel, it is actually working as $h(x) = \sum_i \alpha_i \tanh(x_i, x)$.
- It can be considered as one neural network with one hidden layer.



- The weight of each neuron is an example x_i.
- The number of support vectors is the number of neurons.

□

Mercer's Condition

> A symmetric function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$ implements an inner product in some Hilbert space if and only if it is positive semi-definite; namely $\forall v_i$, the Gram matrix, $\mathcal{G}(i,j) = \mathcal{K}(x_i, x_j)$, is a positive semidefinite matrix.

Notes.

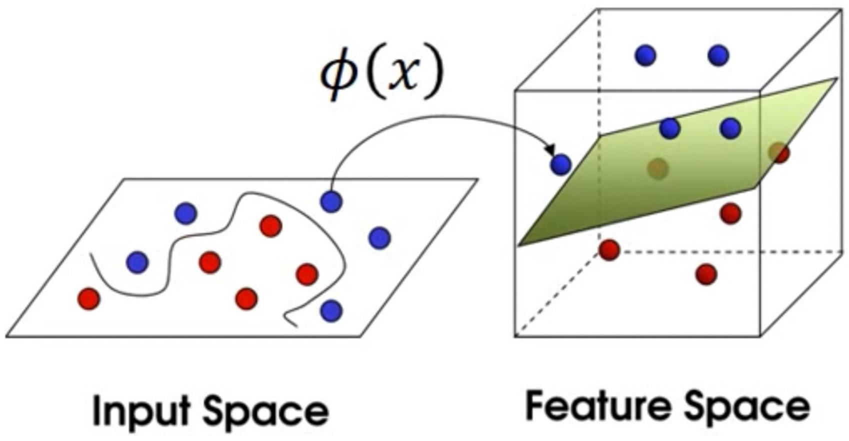
- It can be learned or designed by prior knowledge.

□

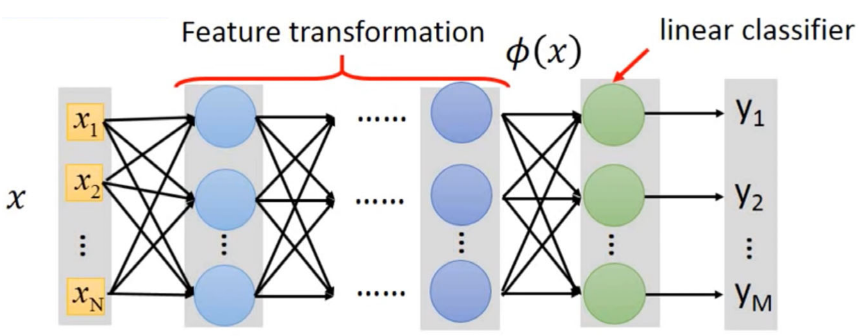
From Machine Learning to Deep Learning

Notes.

- SVM is a feature mapping followed by linear classifier (half-space): SVM kernel is learnable, but not as perfectly done as in ANN.



- Deep learning is feature transformations + linear classifier (half-space).



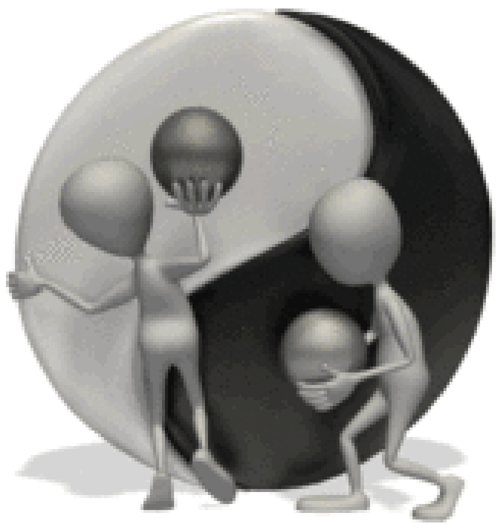
□

SGD with Projection Step

A supermarket manager would like to learn which of his customers have babies on the basis of their shopping carts. Specifically, he sampled i.i.d. customers, where for customer i, let $x_i \subset \{1, \ldots, d\}$ denote the subset of items the customer bought, and let $y_i \in \{1, -1\}$ be the label indicating whether this customer has a baby. As prior knowledge, the manager knows that there are k items such that the label is determined to be 1 iff the customer bought at least one of these k items. Of course, the identity of these k items is not known (otherwise, there was nothing to learn). In addition, according to the store regulation, each customer can buy at most s items.

- Help the manager to design a learning algorithm such that both its time complexity and its sample complexity are polynomial in s, k, and $\frac{1}{\epsilon}$.

Questions?



Contact Information

Associate Professor **Gang Li**
School of Information Technology
Deakin University
Geelong, Victoria 3216, Australia

gangli@tulip.org.au
Open Resources of TULIP-Lab
Team for Universal Learning and Intelligent Processing