Kernels

- Kernel: $K(x, x') = \langle \phi(x), \phi(x') \rangle$
- Sometimes calculating (and even specifying) inner product is easier then writing out and evaluating $\phi(\cdot)$
Polynomial Kernels

- $K(x, x') = (\langle x, x' \rangle + 1)^r$
  - corresponds to feature space with all degree-$r$ monomials
  - Predictors: all degree-$r$ polynomials
  - $D = O(d^r)$ features, still $O(d)$ to calculate

- Variant: $K(x, x') = (\langle x, x' \rangle + a)^r$
  - E.g. for $r=2$:
    \[
    \tilde{\phi}(x) = [a^2, \sqrt{2}a^2x[1], \sqrt{2}a^2x[2], \ldots, \sqrt{2}a^2x[d],
    a\sqrt{2}x[1]x[2], a\sqrt{2}x[1]x[3], \ldots, a\sqrt{2}x[d-1]x[d], x[1]^2, x[2]^2, \ldots, x[d]^2]
    \]
  - Larger $a \Rightarrow$ higher weight on lower order features
    \Rightarrow can use these features with lower (hence cheaper) weights
  - $a$ specifies prior bias / knowledge
RBF Kernels

• Gaussian RBF (Radial Basis Function):
  \[ K(x, x') = e^{-\beta \|x - x'\|^2} \]
Other Kernels

- Many other kernels for specific domain (e.g. images, text, etc)

- E.g., $\phi(\sigma) = \text{all length } r \text{ subsequences (with gaps), weighted by } \beta^{\#gaps}$
  - Measures similarity in terms of appearance of common subsequences, even if with a few indels ($\beta = 0$ doesn’t allow indels)
  - $r = 2, \beta = 0.5$:
    - $\phi('lior') = 1_{li} + 1_{io} + 1_{or} + 0.5 \cdot 1_{io} + 0.5 \cdot 1_{ir} + 0.25 \cdot 1_{ir}$
    - $\phi('igor') = 1_{ig} + 1_{go} + 1_{or} + 0.5 \cdot 1_{io} + 0.5 \cdot 1_{gr} + 0.25 \cdot 1_{ir}$
    - $K('lior', 'igor') = 0.5 \cdot \langle 1_{io}, 1_{io} \rangle + \langle 1_{or}, 1_{or} \rangle + 0.5 \cdot \langle 1_{ir}, 1_{ir} \rangle = 0.5 + 1 + 0.5 = 2$
    - Actually, normalize $\tilde{\phi}(\sigma) = \phi(\sigma)/\|\phi(\sigma)\|$, so $K('lior', 'igor') = 0.56$

- $D = |\Sigma|^r$ features
- $O(|\sigma|^r)$ to calculate explicitly
- Can calculate $K(\sigma, \sigma')$ in time $O(r|\sigma||\sigma'|)$
Can any Function be a Kernel?

• No! Must be an inner product of some feature space
• Technical condition: for any finite set of points, Gram matrix (matrix of kernel evaluations) must be positive semi definite:

Claim: $K$ is a valid kernel if and only if $\forall m \forall x_1, x_2, \ldots, x_m,$

$$G = \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_m) \\ \vdots & \ddots & \vdots \\ K(x_m, x_1) & \cdots & K(x_m, x_m) \end{bmatrix} \succeq 0$$
Radius of a Kernel

• Recall the radius \( R = \sup_x \|x\| \) was important in order to understand the margin.

• Effective dimension = \( \left( \frac{R}{\gamma} \right)^2 \)

• What’s the radius for points represented by a kernel?

\[
R = \sup_x \|\phi(x)\| = \sup_x \sqrt{\langle \phi(x), \phi(x) \rangle} = \sup_x \sqrt{K(x,x)}
\]

• E.g. for Gaussian Kernel:
  • \( K(x,x) = e^{-\beta \|x-x\|^2} = 1 \)
  • Implicit \( \phi(\cdot) \) maps data into unit sphere
  • Actually, since \( K(x,x') \leq 1, \phi(x) \) in single orthant of unit sphere
  • Small \( \beta \rightarrow \) all \( K(x,x') \) close to 1 \( \rightarrow \) all \( \phi(x) \) point in similar direction \( \rightarrow \) \( \phi(x) \) concentrated on small patch of sphere (effective radius even smaller)
  • Large \( \beta \rightarrow \) \( K(x,x') \) could be close to zero \( \rightarrow \) Some \( x,x' \) nearly orthogonal \( \rightarrow \) \( \phi(x) \) dispersed in orthant
One more thing: Bias Term

\[ h_{w,b}(x) = \text{sign}(\langle w, \phi(x) \rangle + b) \]

\[
SVM^\phi_\lambda(S) = \underset{h_{w,b}, w \in \mathbb{R}^d, b \in \mathbb{R}}{\arg \min} \frac{\lambda}{2} \| w \|^2 + \frac{1}{m} \sum_{i=1}^{m} [1 - y_i (\langle w, \phi(x_i) \rangle + b)]_+
\]
One more thing: Bias Term

\[ h_{w,b}(x) = \text{sign}(\langle w, \phi(x) \rangle + b) \]

\[
\text{SVM}_\lambda^\phi(S) = \arg\min_{h_{w,b}, w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{\lambda}{2} \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} [1 - y_i(\langle w, \phi(x_i) \rangle + b)]_+
\]

- Representor Theorem: \( \text{SVM}_\lambda^\phi(S) = \sum_i \alpha_i \phi(x_i) + b \)
  - \( \min_{\alpha,b} \frac{\lambda}{2} \alpha^T G \alpha + \frac{1}{m} \sum_i [1 - y_i(G_j \alpha + b)]_+ \)
  - \( h_{\alpha,S,b}(x) = \text{sign}(\sum_i \alpha_i K(x, x_i) + b) \)
- Is this equivalent to using \( \tilde{\phi}(x) = [\phi(x), 1] \in \mathbb{R}^{d+1} \) with \( w[d+1] = b \)?
  - \( \tilde{K}(x, x') = \langle \tilde{\phi}(x), \tilde{\phi}(x') \rangle = \langle [\phi(x), 1], [\phi(x'), 1] \rangle = K(x, x') + 1 \)
- Not exactly! \( b \) not included in norm
SVM as Structural Risk Minimization

\[ \arg \min_w \frac{\lambda}{2} \|w\|^2 + L_{\text{hinge}}^S(h_w) \]

“Regularizer”
- Prior bias: which predictors do we think are more likely? Which do we consider simpler?
- Sequence of hypothesis classes
  \[ \mathcal{H}_B = \{h_w(x) = \langle w, \phi(x) \rangle \mid \|w\| \leq B \} \]

Empirical Error
- Data-dependent term—ensures we actually fit the data
- Can use other loss functions
Model Selection

arg min \( \lambda R(w) + L_S(h_w) \)

• How do we choose \( \lambda \)?
• How do we choose how complicated a model to use?
  How do we select \( R(w) \)?
  How do we balance \( R(w) \) and \( L_S(h_w) \)?

Lower Approximation Error \( \inf_w L(w) \)
Higher Estimation Error \( |L_S(w) - L(w)| \)

Higher Approximation Error \( \inf_w L(w) \)
Lower Estimation Error \( |L_S(w) - L(w)| \)
The Regularization Path

\[
\text{arg min } R(w) \text{ and } L_S(h_w)
\]

Regularization Path = \{w(\lambda) = \arg \min \lambda R(w) + L_S(w) \mid \lambda \in [0, \infty]\}
Model Selection: Validation

• Use an independently drawn validation set $S_{val}$ to estimate the generalization error of each predictor on the regularization path

1. Draw $S_{tr}$
2. For each $\lambda \in \Lambda$: $w(\lambda) = \arg \min \lambda R(w) + L_{str}(w)$
3. Draw $S_{val}$
4. Return $\lambda^* = \arg \min_{\lambda \in \Lambda} L_{sval}(w(\lambda))$

• Recall that since $w(\lambda)$ depends only on $S_{tr}$ but not on $S_{val}$:

$$w.p. \geq 1 - \delta: \forall \lambda \in \Lambda: |L_{sval}(w(\lambda)) - L_{d}(w(\lambda))| \leq \sqrt{\frac{\log 2/\delta + \log |\Lambda|}{2m}}$$

• Even if we vary a single parameter, we’re still doing ERM on a fairly small class, and should be fine.

• Fix $\lambda \in \Lambda$: Is $L_{str}(w(\lambda))$ an unbiased estimator of $L_{D}(w(\lambda))$?
  (i.e. does its expectation equals to $L_{D}(w(\lambda))$)

• Fix $\lambda \in \Lambda$: Is $L_{sval}(w(\lambda))$ an unbiased estimator of $L_{D}(w(\lambda))$?

• Is $L_{sval}(w(\lambda^*))$ an unbiased estimator of $L_{D}(w(\lambda^*))$?
  • Need to draw third independent set $S_{tst}$
Train-Validation-Test

• We usually can’t keep drawing more and more independent samples...

1. Partition available data \textbf{randomly} into \(S_{tr}, S_{val}, S_{tst}\)
2. \(\forall \lambda \in \Lambda: w(\lambda) = \arg\min_\lambda \lambda R(w) + L_{str}(w)\)
3. Choose \(\lambda^* = \arg\min_{\lambda \in \Lambda} L_{val}(w(\lambda))\)
4. Evaluate \(L_{tst}(w)\) to get a sense of its performance

• Important:
  • Don’t use \(S_{tst}\) in any way before evaluating on it (e.g. to normalize data, choose features, etc)
  • Use \(S_{tst}\) only once!
    If you use it multiple times (e.g. change something and try again), it’s no longer an unbiased estimator of the generalization error. It’s now another validation set.
Cross Validation

K-fold Cross Validation

1. Randomly partition $S$ into $S_1, S_2, ..., S_k$
2. For each $\lambda \in \Lambda$:
   For each $i = 1..k$:
   Use $\bar{S}_i = S \setminus S_i$ (everything except $S_i$)
   $w_i(\lambda) = \arg \min \lambda R(w) + L_{\bar{S}_i}(w)$
   $xverr(\lambda) = \frac{1}{k} \sum_i L_{S_i}(w_i(\lambda))$
3. $\lambda^* = \arg \min_{\lambda \in \Lambda} xverr(\lambda)$
4. Return $w = \arg \min \lambda^* R(w) + L_S(w)$

- Can also cross-validate other parameters (kernel params, etc)
- Extreme case: $k = |S|$ (Leave-One-Out Cross Validation)
- Is $xverr(\lambda^*)$ an unbiased estimator of $L_D(w)$?
  - Need a separate held-out test set
Another Use of Validation: Boosting the Confidence

• Recall how all guarantees look like (and the definition of PAC-Learning):
  For any $\epsilon, \delta$, with probability $\geq 1 - \delta$ over a sample of size $m = \text{poly}(\ldots, \log \frac{1}{\delta}, \ldots)$: $L(A(S)) \leq L^* + \epsilon$

• What if we had a method that only worked with some fixed (even low) probability, say $1 - \delta_0 = 0.1$, requiring $m_0(\epsilon)$ samples to ensure
  \[ \Pr(L(A(S)) \leq L^* + \epsilon) \geq 1 - \delta_0 \]
  (e.g. $L^* = \inf_{h \in \mathcal{H}} L(h)$)

• Can we use it to get a method that works in very high probability?
Boosting the Confidence

• For any $\delta$:

1. For $i=1..k$: 
   \[ k = \frac{\log 2/\delta}{\log 1/\delta_0} \]
   Collect $m_0(\varepsilon/2)$ independent samples $S_i$
   $h_i = A(S_i)$

2. Collect $m_{val} = \frac{8 \log(4k/\delta)}{\varepsilon^2}$ additional independent samples $S_{val}$

3. Return $\hat{h} = \arg \min_{h_1,...,h_k} L_{S_{val}}(h_i)$

• Claim: w.p. $\geq 1 - \delta$, $L(\hat{h}) \leq L^* + \varepsilon$

• Total samples used: $O \left( m_0 \left( \frac{\varepsilon}{2} \right) \cdot \log \frac{1}{\delta} + \frac{\log^4 \delta}{\varepsilon^2} \right)$
Boosting the Error?

• What if we can only find a predictor with relatively high excess error $\epsilon$?
• We can always find a predictor with error $\leq \frac{1}{2}$
• What if we have an algorithm that, for any source dist $\mathcal{D}$ s.t. $\inf_h L_{\mathcal{D}}(h) = 0$, finds $L_{\mathcal{D}}(A(S)) \leq \frac{1}{2} - \gamma$.
• Can we use $A(\cdot)$ to find a predictor with arbitrarily low error?

Answer: Next Presentation....