Recap for Midterm

David S. Rosenberg

New York University

February 28, 2018
1 Learning Theory Framework
2 Regularization
3 Optimization
4 Classification
5 The Representer Theorem and Kernelization
Learning Theory Framework
Some Formalization

The Spaces

- $\mathcal{X}$: input space
- $\mathcal{Y}$: outcome space
- $\mathcal{A}$: action space

Prediction Function (or “decision function”)

A prediction function (or decision function) gets input $x \in \mathcal{X}$ and produces an action $a \in \mathcal{A}$:

$$f : \mathcal{X} \rightarrow \mathcal{A}$$

$x \mapsto f(x)$

Loss Function

A loss function evaluates an action in the context of the outcome $y$.

$$\ell : \mathcal{A} \times \mathcal{Y} \rightarrow \mathbb{R}$$

$(a, y) \mapsto \ell(a, y)$
Risk and the Bayes Prediction Function

Definition

The **risk** of a prediction function \( f : \mathcal{X} \rightarrow \mathcal{A} \) is

\[
R(f) = \mathbb{E} \ell(f(x), y).
\]

In words, it’s the **expected loss** of \( f \) on a new example \((x, y)\) drawn randomly from \( P_{\mathcal{X} \times \mathcal{Y}} \).

Definition

A **Bayes prediction function** \( f^* : \mathcal{X} \rightarrow \mathcal{A} \) is a function that achieves the **minimal risk** among all possible functions:

\[
f^* \in \arg \min_f R(f),
\]

where the minimum is taken over all functions from \( \mathcal{X} \) to \( \mathcal{A} \).

- The risk of a Bayes prediction function is called the **Bayes risk**.
The Empirical Risk

- Let $D_n = ((x_1, y_1), \ldots, (x_n, y_n))$ be drawn i.i.d. from $\mathcal{P}_{X \times Y}$.
- The **empirical risk** of $f : X \to A$ with respect to $D_n$ is
  \[
  \hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i).
  \]
- A function $\hat{f}$ is an **empirical risk minimizer** if
  \[
  \hat{f} \in \arg \min_f \hat{R}_n(f),
  \]
  where the minimum is taken over all functions.
- But unconstrained ERM can **overfit**.
Constrained Empirical Risk Minimization

- Hypothesis space $\mathcal{F}$, a set of [prediction] functions mapping $\mathcal{X} \rightarrow \mathcal{A}$
- **Empirical risk minimizer** (ERM) in $\mathcal{F}$ is

$$\hat{f}_n \in \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i).$$

- **Risk minimizer** in $\mathcal{F}$ is $f^*_\mathcal{F} \in \mathcal{F}$, where

$$f^*_\mathcal{F} \in \arg\min_{f \in \mathcal{F}} \mathbb{E}\ell(f(x), y).$$
Error Decomposition

- **Approximation Error** (of $\mathcal{F}$) = $R(f_{\mathcal{F}}) - R(f^*)$

- **Estimation error** (of $\hat{f}_n$ in $\mathcal{F}$) = $R(\hat{f}_n) - R(f_{\mathcal{F}})$

\[ f^* = \arg\min_{f} \mathbb{E}\ell(f(X), Y) \]
\[ f_{\mathcal{F}} = \arg\min_{f \in \mathcal{F}} \mathbb{E}\ell(f(X), Y) \]
\[ \hat{f}_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \]
The excess risk of the ERM $\hat{f}_n$ can be decomposed:

$$\text{Excess Risk}(\hat{f}_n) = R(\hat{f}_n) - R(f^*)$$

$$= R(\hat{f}_n) - R(f_{\mathcal{F}}) + R(f_{\mathcal{F}}) - R(f^*)$$

- estimation error
- approximation error
Optimization Error

- In practice, we don’t find the ERM $\hat{f}_n \in \mathcal{F}$.
- Optimization algorithm returns $\tilde{f}_n \in \mathcal{F}$, which we hope is good enough.
- **Optimization error**: If $\tilde{f}_n$ is the function our optimization method returns, and $\hat{f}_n$ is the empirical risk minimizer, then

  $$
  \text{Optimization Error} = R(\tilde{f}_n) - R(\hat{f}_n).
  $$

- Extended decomposition:

  $$
  \text{Excess Risk}(\tilde{f}_n) = R(\tilde{f}_n) - R(f^*)
  = \underbrace{R(\tilde{f}_n) - R(\hat{f}_n)}_{\text{optimization error}} + \underbrace{R(\hat{f}_n) - R(f_\mathcal{F})}_{\text{estimation error}} + \underbrace{R(f_\mathcal{F}) - R(f^*)}_{\text{approximation error}}
  $$
Regularization
Constrained Empirical Risk Minimization

Constrained ERM (Ivanov regularization)

For complexity measure $\Omega : \mathcal{F} \rightarrow [0, \infty)$ and fixed $r \geq 0$,

$$
\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)
$$

s.t. $\Omega(f) \leq r$

- Choose $r$ using validation data or cross-validation.
- Each $r$ corresponds to a different hypothesis space. Could also write:

$$
\min_{f \in \mathcal{F}_r} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)
$$
Penalized Empirical Risk Minimization

Penalized ERM (Tikhonov regularization)

For complexity measure $\Omega : \mathcal{F} \rightarrow [0, \infty)$ and fixed $\lambda \geq 0$,

$$
\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) + \lambda \Omega(f)
$$

- Choose $\lambda$ using validation data or cross-validation.
- (Ridge regression in homework is of this form.)
Ridge Regression (Tikhonov Form)

The ridge regression solution for regularization parameter \( \lambda \geq 0 \) is

\[
\hat{w} = \arg \min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left( w^T x_i - y_i \right)^2 + \lambda \| w \|_2^2,
\]

where \( \| w \|_2^2 = w_1^2 + \cdots + w_d^2 \) is the square of the \( \ell_2 \)-norm.

Ridge Regression (Ivanov Form)

The ridge regression solution for complexity parameter \( r \geq 0 \) is

\[
\hat{w} = \arg \min_{\| w \|_2^2 \leq r^2} \frac{1}{n} \sum_{i=1}^{n} \left( w^T x_i - y_i \right)^2.
\]
Lasso Regression (Tikhonov Form)

The lasso regression solution for regularization parameter $\lambda \geq 0$ is

$$\hat{w} = \arg \min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2 + \lambda \| w \|_1,$$

where $\| w \|_1 = |w_1| + \cdots + |w_d|$ is the $\ell_1$-norm.

Lasso Regression (Ivanov Form)

The lasso regression solution for complexity parameter $r \geq 0$ is

$$\hat{w} = \arg \min_{\| w \|_1 \leq r} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2.$$
Ridge vs. Lasso: Regularization Paths

Modified from Hastie, Tibshirani, and Wainwright’s *Statistical Learning with Sparsity*, Fig 2.1. About predicting crime in 50 US cities.
For identical features
- $\ell_1$ regularization spreads weight arbitrarily (all weights same sign)
- $\ell_2$ regularization spreads weight evenly

Linearly related features
- $\ell_1$ regularization chooses variable with larger scale, 0 weight to others
- $\ell_2$ prefers variables with larger scale – spreads weight proportional to scale
Intersection could be anywhere on the top right edge.

Minor perturbations (in data) can drastically change intersection point – very unstable solution.

Makes division of weight among highly correlated features (of same scale) seem arbitrary.
  - If $x_1 \approx 2x_2$, ellipse changes orientation and we hit a corner. (Which one?)
Elastic Net

- The **elastic net** combines lasso and ridge penalties:

\[
\hat{w} = \arg \min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2 + \lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2
\]

- We expect correlated random variables to have similar coefficients.
Elastic net solution is closer to $w_2 = w_1$ line, despite high correlation.
Elastic Net Results on Model

- Lasso on left; Elastic net on right.
- Ratio of $\ell_2$ to $\ell_1$ regularization roughly 2:1.
Suppose design matrix $X$ is orthogonal, so $X^T X = I$, and contours are circles (and features uncorrelated).

Then OLS solution in green or red regions implies elastic-net constrained solution will be at corner.
Elastic Net Summary

- With uncorrelated features, we can get sparsity.
- Among correlated features (same scale), we spread weight more evenly.
Finding Lasso Solution

- Many options.
- Convert to quadratic program using positive/negative parts

\[
\begin{align*}
\min_{w^+, w^-} & \sum_{i=1}^{n} \left( (w^+ - w^-)^T x_i - y_i \right)^2 + \lambda_1^T (w^+ + w^-) \\
\text{subject to} & \quad w_i^+ \geq 0 \text{ for all } i \\
& \quad w_i^- \geq 0 \text{ for all } i,
\end{align*}
\]

- Coordinate descent
  - Lasso has closed form solution for coordinate minimizers!
- Subgradient descent
Optimization
Suppose we have a hypothesis space of functions $\mathcal{F} = \{ f_w : \mathcal{X} \rightarrow \mathcal{A} | w \in \mathbb{R}^d \}$ parameterized by $w \in \mathbb{R}^d$. ERM is to find $w$ minimizing

$$\hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_w(x_i), y_i)$$

Suppose $\ell(f_w(x_i), y_i)$ is differentiable as a function of $w$. Then we can do gradient descent on $\hat{R}_n(w)$. 

...
Gradient Descent: How does it scale with $n$?

- At every iteration, we compute the gradient at current $w$:

$$\nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^{n} \nabla_w \ell(f_w(x_i), y_i)$$

- We have to touch all $n$ training points to take a single step. [$O(n)$]

- What if we just use an estimate of the gradient?
Minibatch Gradient

- The full gradient is

\[ \nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^{n} \nabla_w \ell(f_w(x_i), y_i) \]

- It’s an average over the full batch of data \( \mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\} \).

- Let’s take a random subsample of size \( N \) (called a minibatch):

\[ (x_{m_1}, y_{m_1}), \ldots, (x_{m_N}, y_{m_N}) \]

- The minibatch gradient is

\[ \nabla \hat{R}_N(w) = \frac{1}{N} \sum_{i=1}^{N} \nabla_w \ell(f_w(x_{m_i}), y_{m_i}) \]

- Minibatch gradient is an unbiased estimate of full-batch gradient: 

\[ \mathbb{E} \left[ \nabla \hat{R}_N(w) \right] = \nabla \hat{R}_n(w) \]
How big should minibatch be?

- Tradeoffs of minibatch size:
  - Bigger $N \implies$ Better estimate of gradient, but slower (more data to touch)
  - Smaller $N \implies$ Worse estimate of gradient, but can be quite fast

- Even $N = 1$ works, it’s traditionally called **stochastic gradient descent** (SGD).

- Quality of minibatch estimate depends on
  - size of minibatch
  - but is **independent** of full dataset size $n$

- Discussed in Concept Check question.
Descent Directions

- A step direction is a **descent direction** if, for small enough step size, the objective function value always decreases.

- Negative gradient is a descent direction.

- A negative subgradient is **not** a descent direction. But always **takes you closer to a minimizer**.

- Negative stochastic or minibatch gradient direction is **not** a descent direction. But we have convergence theorems.

- Negative stochastic subgradient step direction is **not** a descent direction. But we have convergence theorems (not discussed in class).
Classification
The Score Function

- Action space $\mathcal{A} = \mathbb{R}$  
- Output space $\mathcal{Y} = \{-1, 1\}$
- **Real-valued prediction function** $f : \mathcal{X} \to \mathbb{R}$

**Definition**

The value $f(x)$ is called the **score** for the input $x$.

- In this context, $f$ may be called a **score function**.
- Intuitively, magnitude of the score represents the **confidence of our prediction**.
The Margin

Definition

The **margin** (or **functional margin**) for predicted score $\hat{y}$ and true class $y \in \{-1, 1\}$ is $y\hat{y}$.

- The margin often looks like $yf(x)$, where $f(x)$ is our score function.
- The margin is a measure of how **correct** we are.
  - If $y$ and $\hat{y}$ are the same sign, prediction is **correct** and margin is **positive**.
  - If $y$ and $\hat{y}$ have different sign, prediction is **incorrect** and margin is **negative**.
- We want to **maximize the margin**.
Classification Losses

Logistic/Log loss: $\ell_{\text{Logistic}} = \log(1 + e^{-m})$

Logistic loss is differentiable. Logistic loss always wants more margin (loss never 0).
Support Vector Machine

- Hypothesis space $\mathcal{F} = \{ f(x) = w^T x + b \mid w \in \mathbb{R}^d, b \in \mathbb{R} \}$.
- $\ell_2$ regularization (Tikhonov style)
- Loss $\ell(m) = \max \{1 - m, 0\}$
- The SVM prediction function is the solution to

$$
\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2} \|w\|^2 + \frac{c}{n} \sum_{i=1}^{n} \max (0, 1 - y_i [w^T x_i + b]) .
$$
The SVM optimization problem is equivalent to

\[
\text{minimize} \quad \frac{1}{2} \|w\|^2 + \frac{c}{n} \sum_{i=1}^{n} \xi_i \\
\text{subject to} \quad -\xi_i \leq 0 \text{ for } i = 1, \ldots, n \\
(1 - y_i [w^T x_i + b]) - \xi_i \leq 0 \text{ for } i = 1, \ldots, n
\]

- Differentiable objective function
- \( n + d + 1 \) unknowns and \( 2n \) affine constraints.
- A quadratic program that can be solved by any off-the-shelf QP solver.
The Representer Theorem and Kernelization
General Objective Function for Linear Hypothesis Space (Details)

- **Generalized objective:**

\[
\min_{w \in \mathcal{H}} R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle),
\]

where
- \(w, x_1, \ldots, x_n \in \mathcal{H}\) for some Hilbert space \(\mathcal{H}\). (We typically have \(\mathcal{H} = \mathbb{R}^d\).)
- \(\| \cdot \|\) is the norm corresponding to the inner product of \(\mathcal{H}\). (i.e. \(\|w\| = \sqrt{\langle w, w \rangle}\))
- \(R : [0, \infty) \rightarrow \mathbb{R}\) is nondecreasing (Regularization term), and
- \(L : \mathbb{R}^n \rightarrow \mathbb{R}\) is arbitrary (Loss term).

- Ridge regression and SVM are of this form.
- What if we use lasso regression? No! \(\ell_1\) norm does not correspond to an inner product.
The Representer Theorem

Let \( J(w) = R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle) \) under conditions described above.

**Theorem (Representer Theorem)**

If \( J(w) \) has a minimizer, then it has a minimizer of the form

\[
  w^* = \sum_{i=1}^{n} \alpha_i x_i.
\]

If \( R \) is strictly increasing, then all minimizers have this form.

Basic idea of proof:

- Let \( M = \text{span}(x_1, \ldots, x_n) \). [the “span of the data”]
- Let \( w = \text{Proj}_M w^* \), for some minimizer \( w^* \) of \( J(w) \).
- Then \( \langle w, x_i \rangle = \langle w^*, x_i \rangle \), so loss part doesn’t change.
- \( \|w\| \leq \|w^*\| \), since projection reduces norm. So regularization piece never increases.
Reparametrization with Representer Theorem

- Original plan:
  - Find $w^* \in \text{arg min}_{w \in \mathcal{H}} R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle)$
  - Predict with $\hat{f}(x) = \langle w^*, x \rangle$.

- Plugging in result of representer theorem, it’s equivalent to
  - Find $\alpha^* \in \text{arg min}_{\alpha \in \mathbb{R}^n} R(\sqrt{\alpha^T K \alpha}) + L(K \alpha)$
  - Predict with $\hat{f}(x) = k_x^T \alpha^*$, where

$$K = \begin{pmatrix} \langle x_1, x_1 \rangle & \cdots & \langle x_1, x_n \rangle \\ \vdots & \ddots & \vdots \\ \langle x_n, x_1 \rangle & \cdots & \langle x_n, x_n \rangle \end{pmatrix} \quad \text{and} \quad k_x = \begin{pmatrix} \langle x_1, x \rangle \\ \vdots \\ \langle x_n, x \rangle \end{pmatrix}$$

- Every element $x \in \mathcal{H}$ occurs inside an inner products with a training input $x_i \in \mathcal{H}$.
**Kernelization**

**Definition**

A method is **kernelized** if every feature vector $\psi(x)$ only appears inside an inner product with another feature vector $\psi(x')$. This applies to both the optimization problem and the prediction function.

- Here we are using $\psi(x) = x$. Thus finding

$$\alpha^* \in \arg\min_{\alpha \in \mathbb{R}^n} R \left( \sqrt{\alpha^T K \alpha} \right) + L (K \alpha)$$

and making predictions with $\hat{f}(x) = k_x^T \alpha^*$ is a **kernelization** of finding

$$w^* \in \arg\min_{w \in \mathcal{H}} R \left( \|w\| \right) + L (\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle)$$

and making predictions with $\hat{f}(x) = \langle w^*, x \rangle$. 
Once we have kernelized:

- $\alpha^* \in \text{arg min}_{\alpha \in \mathbb{R}^n} R \left( \sqrt{\alpha^T K \alpha} \right) + L (K \alpha)$
- $\hat{f}(x) = k_x^T \alpha^*$

We can do the “kernel trick”.

Replace each $\langle x, x' \rangle$ by $k(x, x')$, for any kernel function $k$, where $k(x, x') = \langle \psi(x), \psi(x') \rangle$.

Predictions

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha^*_i k(x_i, x)$$
The Kernel Function: Why do we need this?

- **Feature map**: $\psi: \mathcal{X} \rightarrow \mathcal{H}$
- **The kernel function** corresponding to $\psi$ is
  \[ k(x, x') = \langle \psi(x), \psi(x') \rangle. \]

- Why introduce this new notation $k(x, x')$?

- We can often evaluate $k(x, x')$ without explicitly computing $\psi(x)$ and $\psi(x')$.

- For large feature spaces, can be much faster.
Kernelized SVM (From Lagrangian Duality)

- Kernelized SVM from computing the Lagrangian Dual Problem:

\[
\begin{align*}
\max_{\alpha \in \mathbb{R}^n} & \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
\text{s.t.} & \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \\
& \quad \alpha_i \in \left[0, \frac{c}{n}\right] \quad i = 1, \ldots, n.
\end{align*}
\]

- If \(\alpha^*\) is an optimal value, then

\[
\begin{align*}
\mathbf{w}^* &= \sum_{i=1}^{n} \alpha_i^* y_i x_i \\
\hat{f}(x) &= \sum_{i=1}^{n} \alpha_i^* y_i x_i^T x.
\end{align*}
\]

- Note that the prediction function is also kernelized.
Sparsity in the Data from Complementary Slackness

- Kernelized predictions given by

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i^* y_i x_i^T x.$$ 

- By a Lagrangian duality analysis (specifically from complementary slackness), we find

  $$y_i \hat{f}(x_i) < 1 \implies \alpha_i^* = \frac{c}{n}$$
  $$y_i \hat{f}(x_i) = 1 \implies \alpha_i^* \in \left[0, \frac{c}{n}\right]$$
  $$y_i \hat{f}(x_i) > 1 \implies \alpha_i^* = 0$$

- So we can leave out any $x_i$ “on the good side of the margin” ($y_i \hat{f}(x_i) > 1$).
- $x_i$'s that we must keep, because $\alpha_i^* \neq 0$, are called support vectors.