Review for Midterm

DS-GA 1003 Machine Learning

NYU CDS

March 17, 2021

Contents

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 $a \in \mathcal{A}$:

$$\begin{array}{rccc} f: & \mathcal{X} & \rightarrow & \mathcal{A} \\ & x & \mapsto & f(x) \end{array}$$

Loss Function

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

$$\begin{array}{rccc} \ell: & \mathcal{A} \times \mathcal{Y} & \rightarrow & \mathrm{R} \\ & (a, y) & \mapsto & \ell(a, y) \end{array}$$

Risk and the Bayes Prediction Function

Definition

The **risk** of a prediction function $f : \mathcal{X} \to \mathcal{A}$ 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** $f^* : \mathcal{X} \to \mathcal{A}$ 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**.

Bayes Prediction Function

- If loss function is $L2$, then $f^*(x) = E[Y|X = x]$
- if loss function is $L1$, then $f^*(x)$ is the median of the distribution of Y conditioned on $X = x$.
- If $\mathcal{Y}$ is discrete and loss function is $0-1$ loss, then $f^*(x) = \underset{c \in \mathcal{Y}}{\textit{argmax}}\ p(y = c|x)$

Question: Let x be sampled uniformly from $\{-100, -99, \ldots, 99, 100\}$. For every sample x_i, y_i is generated as $y_i = x_i + \eta$, $\eta \sim \mathcal{N}(0, \sigma)$, $\sigma > 0$. What is the Bayes prediction function under L_2 and L_1 loss?

The Empirical Risk

- Let $\mathcal{D}_n = ((x_1, y_1), \ldots, (x_n, y_n))$ be drawn i.i.d. from $\mathcal{P}_{\mathcal{X} \times \mathcal{Y}}$.
- The **empirical risk** of $f : \mathcal{X} \to \mathcal{A}$ with respect to $\mathcal{D}_n$ 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

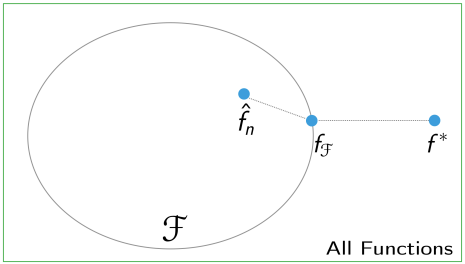
- Hypothesis space $\mathcal{F}$, a set of [prediction] functions mapping $\mathcal{X} \to \mathcal{A}$
- **Empirical risk minimizer** (ERM) in $\mathcal{F}$ is

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

- **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



$$f^* = \arg\min_f \mathbb{E}\ell(f(X), Y)$$

$$f_{\mathcal{F}} = \arg\min_{f \in \mathcal{F}} \mathbb{E}\ell(f(X), Y))$$

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

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

Excess Risk Decomposition for ERM

- The excess risk of the ERM $\hat{f}_n$ can be decomposed:

$$\begin{aligned} \textbf{Excess Risk}(\hat{f}_n) &= R(\hat{f}_n) - R(f^*) \\ &= \underbrace{R(\hat{f}_n) - R(f_{\mathcal{F}})}_{\text{estimation error}} + \underbrace{R(f_{\mathcal{F}}) - R(f^*)}_{\text{approximation error}} . \end{aligned}$$

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:

$$\begin{aligned}\textbf{Excess Risk}(\tilde{f}_n) &= R(\tilde{f}_n) - R(f^*) \\ &= \underbrace{R(\tilde{f}_n) - R(\hat{f}_n)}_{\text{optimization error}} + \underbrace{R(\hat{f}_n) - R(f_{\mathcal{F}})}_{\text{estimation error}} + \underbrace{R(f_{\mathcal{F}}) - R(f^*)}_{\text{approximation error}}\end{aligned}$$

Question

Select true of false for each of the following statements:

1. Approximation Error is a Random Variable
2. Estimation Error is a Random Variable
3. Optimization Error is a Random Variable.
4. If the hypothesis space consists of all possible functions, then approximation error is non-zero.
5. Estimation Error can be negative.
6. Optimization Error can be negative.
7. The empirical risk of the ERM, $\hat{R}(\hat{f})$, is an unbiased estimator of the risk of the ERM $R(\hat{f})$. Does your answer change if it's a $\hat{R}(f)$ where f is independent of training data?

Question

For each, use $\leqslant$, $\geqslant$, or $=$ to determine the relationship between the two quantities, or if the relationship cannot be determined. Throughout assume $\mathcal{F}_1, \mathcal{F}_2$ are hypothesis spaces with $\mathcal{F}_1 \subset \mathcal{F}_2$, and assume we are working with a fixed loss function ℓ.

1. The estimation errors of two decision functions f_1, f_2 that minimize the empirical risk over the same hypothesis space, where f_2 uses 5 extra data points.
2. The approximation errors of the two decision functions f_1, f_2 that minimize risk with respect to $\mathcal{F}_1, \mathcal{F}_2$, respectively (i.e., $f_1 = f_{\mathcal{F}_1}$ and $f_2 = f_{\mathcal{F}_2}$).
3. The empirical risks of two decision functions f_1, f_2 that minimize the empirical risk over $\mathcal{F}_1, \mathcal{F}_2$, respectively. Both use the same fixed training data.
4. The estimation errors (for $\mathcal{F}_1, \mathcal{F}_2$, respectively) of two decision functions f_1, f_2 that minimize the empirical risk over $\mathcal{F}_1, \mathcal{F}_2$, respectively.
5. The risk of two decision functions f_1, f_2 that minimize the empirical risk over $\mathcal{F}_1, \mathcal{F}_2$, respectively.

Regularization

Constrained Empirical Risk Minimization

Constrained ERM (Ivanov regularization)

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

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$
$$\text{s.t. } \Omega(f) \leqslant r$$

- Choose r using validation data or cross-validation.
- Each r corresponds to a different hypothesis spaces. 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 $\Omega : \mathcal{F} \to [0, \infty)$ and fixed $\lambda \geqslant 0$,

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

- Choose λ using validation data or cross-validation.
- (Ridge regression in homework is of this form.)

Ridge Regression: Workhorse of Modern Data Science

Ridge Regression (Tikhonov Form)

The ridge regression solution for regularization parameter $\lambda \geqslant 0$ is

$$\hat{w} = \underset{w \in \mathbb{R}^d}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2 + \lambda \|w\|_2^2,$$

where $\|w\|_2^2 = w_1^2 + \cdots + w_d^2$ is the square of the ℓ_2-norm.

Ridge Regression (Ivanov Form)

The ridge regression solution for complexity parameter $r \geqslant 0$ is

$$\hat{w} = \underset{\|w\|_2^2 \leqslant r^2}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2.$$

Lasso Regression: Workhorse (2) of Modern Data Science

Lasso Regression (Tikhonov Form)

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

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

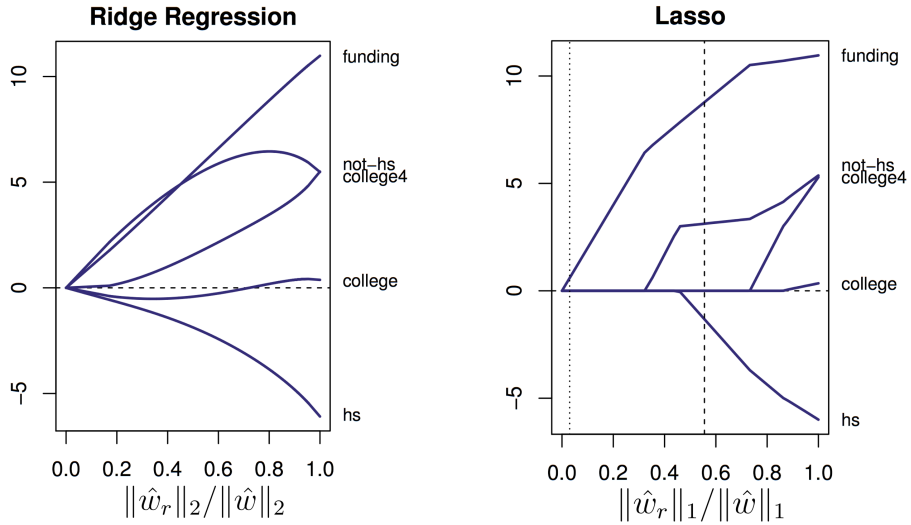
where $\|w\|_1 = |w_1| + \cdots + |w_d|$ is the ℓ_1-norm.

Lasso Regression (Ivanov Form)

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

$$\hat{w} = \underset{\|w\|_1 \leqslant r}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2.$$

Ridge vs. Lasso: Regularization Paths

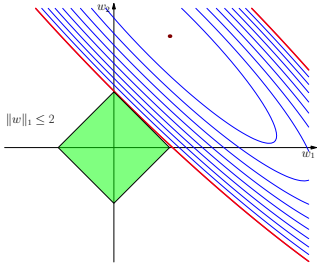


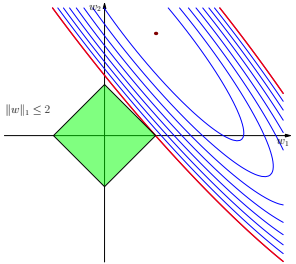
Modified from Hastie, Tibshirani, and Wainwright's *Statistical Learning with Sparsity*, Fig 2.1. About predicting crime in 50 US cities.

Linearly Dependent Features: Take Away

- For identical features
  - ℓ_1 regularization spreads weight arbitrarily (all weights same sign)
  - ℓ_2 regularization spreads weight evenly
- Linearly related features
  - ℓ_1 regularization chooses variable with larger scale, 0 weight to others
  - ℓ_2 prefers variables with larger scale – spreads weight proportional to scale

Correlated Features, ℓ_1 Regularization





- 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?)

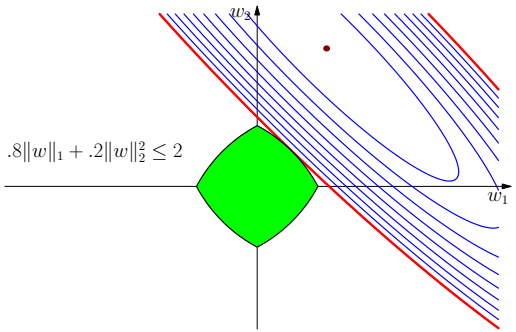
Elastic Net

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

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

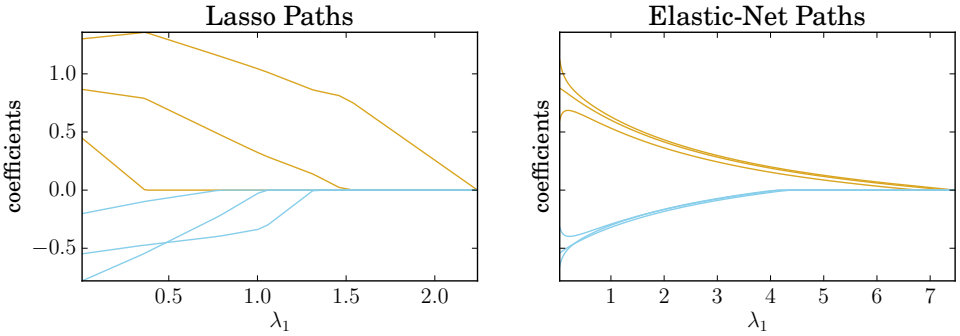
- We expect correlated random variables to have similar coefficients.

Highly Correlated Features, Elastic Net Constraint



- 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 ℓ_2 to ℓ_1 regularization roughly $2:1$.

Elastic Net Summary

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

Question on correlated features

We solve lasso and ridge regression where input lives in $\mathcal{R}^4$. The first two features of all the input vector are duplicates of each other, or $x_{i1} = x_{i2}$ for all i. Consider the following weight vectors:

1. $(0, 1.2, 6.7, 2.1)^T$
2. $(0.6, 0.6, 6.7, 2.1)^T$
3. $(1.2, 0, 6.7, 2.1)^T$
4. $(-0.1, 1.3, 6.7, 2.1)^T$

Which of them are valid solution for a) Ridge Regression and b) Lasso Regression?

Finding Lasso Solution

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

$$\begin{aligned} \min_{w^+, w^-} \quad & \sum_{i=1}^{n} \left(\left(w^+ - w^-\right)^T x_i - y_i \right)^2 + \lambda 1^T \left(w^+ + w^-\right) \\ \text{subject to} \quad & w_i^+ \geqslant 0 \text{ for all } i \qquad w_i^- \geqslant 0 \text{ for all } i, \end{aligned}$$

- Coordinate descent
  - Lasso has closed form solution for coordinate minimizers!
- Subgradient descent

Optimization

Gradient Descent for Empirical Risk and Averages

- Suppose we have a hypothesis space of functions $\mathcal{F} = \left\{ f_w : \mathcal{X} \to \mathcal{A} \mid w \in \mathrm{R}^d \right\}$
  - Parameterized by $w \in \mathrm{R}^d$.
- 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

Subgradient Review

Definition (Subgradient and Subdifferential)

A vector g is a subgradient of (convex) $f : \mathcal{R}^d \to \mathcal{R}$ at x if for all z

$$f(z) \geqslant f(x) + g^T(z - x)$$

. The set of all subgradients at x is called the subdifferential of f at x $\partial f(x)$

Questions:

1. (True/False) If f is convex and differentiable everywhere in the domain, then $\partial f(x) = \{\nabla f(x)\}$
2. (True/False) The subdifferential of f at x, $\partial f(x)$ is always a convex set. (Null set is trivially complex)

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).

Question on Gradient Descent

Decide whether the following statements apply to full batch gradient descent (GD), mini- batch GD, neither, or both.

Assume we're minimizing a differentiable, convex objective function $J(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$, and we are currently at w_t, which is not a minimum. For full batch GD, take $v = \nabla_w J(w_t)$, and for minibatch GD take v to be a minibatch estimate of $\nabla_w J(w_t)$ based on a random sample of the training data.

1. For any step size $\eta > 0$, after applying the update rule $w_{t+1} \leftarrow w_{tl} - \eta v$. we must have $J(w_{t+1}) < J(w_t)$.
2. There must exist some $\eta > 0$ such that after applying the update rule $w_{t+1} + w_t - \eta v$ we have $J(w_{t+1}) < J(w_t)$.
3. v is an unbiased estimator of the full batch gradient.

Classification

The Score Function

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

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

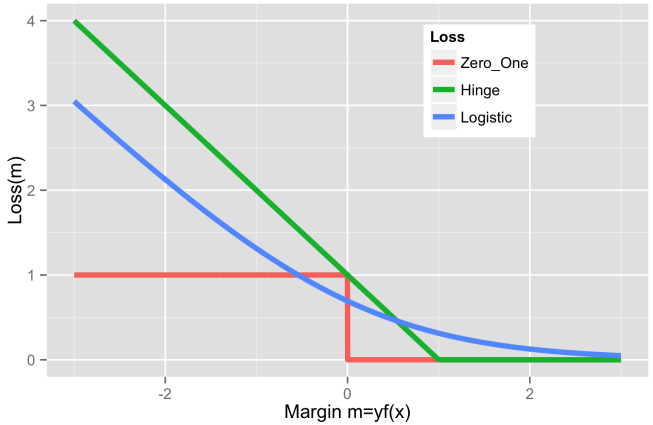
Definition

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: $\ell_{\text{Logistic}} = \log\left(1 + e^{-m}\right)$



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

Support Vector Machine

- Hypothesis space $\mathcal{F} = \left\{ f(x) = w^T x + b \mid w \in \mathrm{R}^d,\, b \in \mathrm{R} \right\}$.
- ℓ_2 regularization (Tikhonov style)
- Loss $\ell(m) = \max\{1 - m, 0\}$
- The SVM prediction function is the solution to

$$\min_{w \in \mathrm{R}^d, b \in \mathrm{R}} \frac{1}{2}\|w\|^2 + \frac{c}{n} \sum_{i=1}^{n} \max\left(0, 1 - y_i \left[w^T x_i + b \right]\right).$$

SVM as a Quadratic Program

- The SVM optimization problem is equivalent to

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & -\xi_i \leqslant 0 \text{ for } i = 1, \ldots, n \\ & \left(1 - y_i\left[w^T x_i + b\right]\right) - \xi_i \leqslant 0 \text{ for } i = 1, \ldots, n \end{aligned}$$

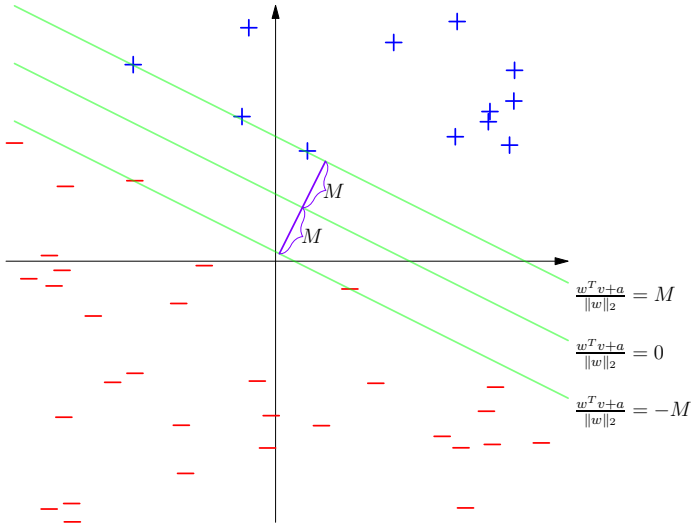
- Differentiable objective function
- $n + d + 1$ unknowns and $2n$ affine constraints.
- A quadratic program that can be solved by any off-the-shelf QP solver.
- We arrived at this optimization problem also from a geometric prospective.

Linear Separability and Hard Margin SVM

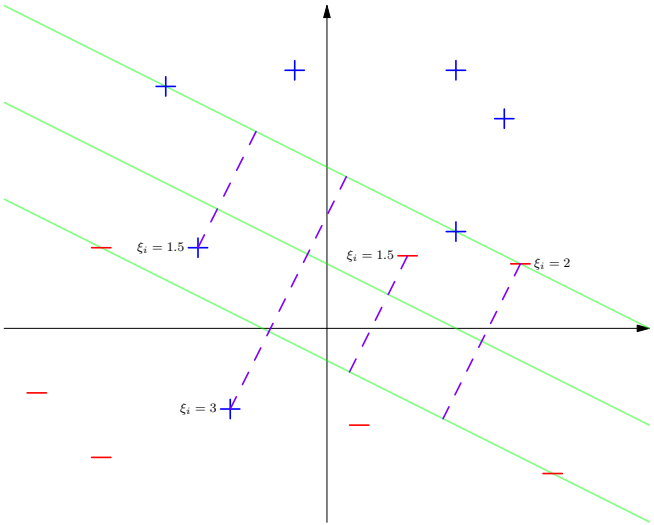
Definition (Linear Separability)

We say (x_i, y_i) for $i = 1, \ldots, n$ are *linearly separable* if there is a $w \in \mathcal{R}^d$ and $b \in \mathcal{R}$ such that $y_i(w^T x_i - b) > 0$ for all i. The set $\{v \in \mathcal{R}^d \mid w^T v - b = 0\}$ is called a *separating hyperplane*.

Maximum Margin Separating Hyperplane



Soft Margin SVM (unlabeled points have $\xi_i = 0$)



Question on Classification

Suppose $x_1, \ldots, x_n \in \mathbb{R}^d$ and $y_1, \ldots, y_n \in \{-1, 1\}$. Here we look at y_i as the label of x_i. We say the data points are linearly separable if there is a vector $v \in \mathbb{R}^d$ and $a \in \mathbb{R}$ such that $v^T x_i > a$ when $y_i = 1$ and $v^T x_i < a$ for $y_i = -1$. Give a method for determining if the given data points are linearly separable.

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 $\mathcal{H} = \mathrm{R}^d$.)
- $\|\cdot\|$ is the norm corresponding to the inner product of $\mathcal{H}$. (i.e. $\|w\| = \sqrt{\langle w, w \rangle}$)
- $R : [0, \infty) \to \mathrm{R}$ is nondecreasing (**Regularization term**), and
- $L : \mathrm{R}^n \to \mathrm{R}$ is arbitrary (**Loss term**).

- Ridge regression and SVM are of this form.
- What if we use lasso regression? No! ℓ_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.
- $\|w\| \leqslant \|w^*\|$, since projection reduces norm. So regularization piece never increases.

Reparametrization with Representer Theorem

- Original plan:
  - Find $w^* \in \arg\min_{w \in \mathcal{H}} R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle)$
  - Predict with $\hat{f}(x) = \langle w^*, x \rangle$.
- Plugging in result of representer theorem, it's equivalent to
  - Find $\alpha^* \in \arg\min_{\alpha \in \mathbb{R}^n} R\left(\sqrt{\alpha^T K \alpha}\right) + L(K\alpha)$
  - Predict with $\hat{f}(x) = k_x^T \alpha^*$, where

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

  - 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

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

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

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

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

Kernelization

- Once we have kernelized:
  - $\alpha^* \in \arg\min_{\alpha \in \mathbb{R}^n} R\left(\sqrt{\alpha^T K \alpha}\right) + L(K\alpha)$
  - $\hat{f}(x) = k_x^T \alpha^*$
- 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**: $\psi : \mathcal{X} \to \mathcal{H}$
- The **kernel function** corresponding to ψ 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:

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

- If α^* is an optimal value, then

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

- 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

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

- 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**.

Question on Kernel

Consider the objective function

$$J(w) = \|Xw - y\|_1 + \lambda \|w\|_2^2$$

Assume we have a positive semidefinite kernel k.

1. What is the kernelized version of this objective?
2. Given a new test point x, find the predicted value.

MLE and Conditional Probability Models

Maximum Likelihood Estimation

- Suppose $\mathcal{D} = (y_1, \ldots, y_n)$ is an i.i.d. sample from some distribution.

Definition

A **maximum likelihood estimator (MLE)** for θ in the **parametric model** $\{p(y;\theta) \mid \theta \in \Theta\}$ is

$$\begin{aligned} \hat{\theta} &\in \arg\max_{\theta \in \Theta} \log p(\mathcal{D}, \hat{\theta}) \\ &= \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \log p(y_i; \theta). \end{aligned}$$

Maximum Likelihood Estimation

- Finding the MLE is an **optimization problem**.
- For some model families, calculus gives a closed form for the MLE.
- Can also use numerical methods we know (e.g. SGD).

Conditional Distribution Estimation (Generalized Regression)

- Task: Given x, predict probability distribution $p(y|x)$
- Method:
  1. Represent $p(y|x)$ with *parametric families* of distributions: $p(y; \theta(x))$ with parameters θ.
  2. Maximize likelihood of training data: $\hat{\theta} \in \arg\max_\theta \log p(\mathcal{D}, \hat{\theta})$
- Models covered:
  1. Logistic regression (Bernoulli distribution)
  2. Poisson regression (Poisson distribution)
  3. Conditional Gaussian/Linear regression (Normal distribution, fixed variance)
  4. Multinomial Logistic Regression (Multinoulli/Categorical distribution)

Linear Probabilistic Classifiers

- Setting: $\mathcal{X} = \mathrm{R}^d$, $\mathcal{Y}$ arbitrary for now
- Want prediction function to map each $x \in \mathrm{R}^d$ to $\theta \in \Theta$ for $p(y; \theta(x))$.
- For a **linear method**, we first **extract information** from $x \in \mathrm{R}^d$ and summarize in a single number with a linear function:

$$\underbrace{x}_{\in \mathrm{R}^d} \mapsto \underbrace{w^T x}_{\in \mathrm{R}}$$

  (That number is analogous to the **score** in classification.)
- As usual, $x \mapsto w^T x$ will include affine functions if we include a constant feature in x.
- $w^T x$ is called the **linear predictor.**
- Still need to map this to Θ.

The Transfer Function

- Need a function to map the linear predictor in R to Θ:

$$\underbrace{x}_{\in \mathsf{R}^d} \mapsto \underbrace{w^T x}_{\in \mathsf{R}} \mapsto \underbrace{f(w^T x)}_{\in \Theta} = \theta,$$

  where $f : \mathsf{R} \to \Theta$. We'll call f the **transfer** function.

- So prediction function is $x \mapsto f(w^T x)$.
- The prediction function gives us the parameter for $p(y; \theta(x))$ used to estimate $p(y|x)$.

Conditional Probability Modeling as Statistical Learning

- Input space $\mathcal{X}$
- Outcome space $\mathcal{Y}$
- All pairs (x, y) are independent with distribution $P_{\mathcal{X} \times \mathcal{Y}}$.
- **Action space** $\mathcal{A} = \{p(y) \mid p \text{ is a probability density or mass function on } \mathcal{Y}\}$.
- Hypothesis space $\mathcal{F}$ contains decision functions $f : \mathcal{X} \to \mathcal{A}$.
- Maximum likelihood estimation for dataset $\mathcal{D} = ((x_1, y_1), \ldots, (x_n, y_n)$ is

$$\hat{f}_{\text{MLE}} \in \arg\max_{f \in \mathcal{F}} \sum_{i=1}^{n} \log\left[f(x_i)(y_i)\right]$$

Conditional Probability Modeling as Statistical Learning

- Take loss $\ell : \mathcal{A} \times \mathcal{Y} \to \mathrm{R}$ for a predicted PDF or PMF $p(y)$ and outcome y to be

$$\ell(p, y) = -\log p(y)$$

- The risk of decision function $f : \mathcal{X} \to \mathcal{A}$ is

$$R(f) = -\mathbb{E}_{x,y} \log [f(x)(y)],$$

where $f(x)$ is a PDF or PMF on $\mathcal{Y}$, and we're evaluating it on y.

Conditional Probability Modeling as Statistical Learning

- The empirical risk of f for a sample $\mathcal{D} = \{y_1, \ldots, y_n\} \in \mathcal{Y}$ is

$$\hat{R}(f) = -\frac{1}{n} \sum_{i=1}^{n} \log [f(x_i)(y_i)] .$$

  This is called the negative **conditional log-likelihood**.

- Thus for the negative log-likelihood loss, ERM and MLE are equivalent

Question on Maximum Likelihood Estimation

1. Suppose we have samples $x_1, \ldots, x_n$ i.i.d. drawn from uniform distribution $\mathcal{U}(-a, a)$. Find the maximum likelihood estimator of a.
2. Which of the following models can be learned by MLE?
   - Perceptron
   - Logistic regression
   - SVM

References

- DS-GA 1003 Machine Learning Spring 2019
- DS-GA 1003 Machine Learning Spring 2020