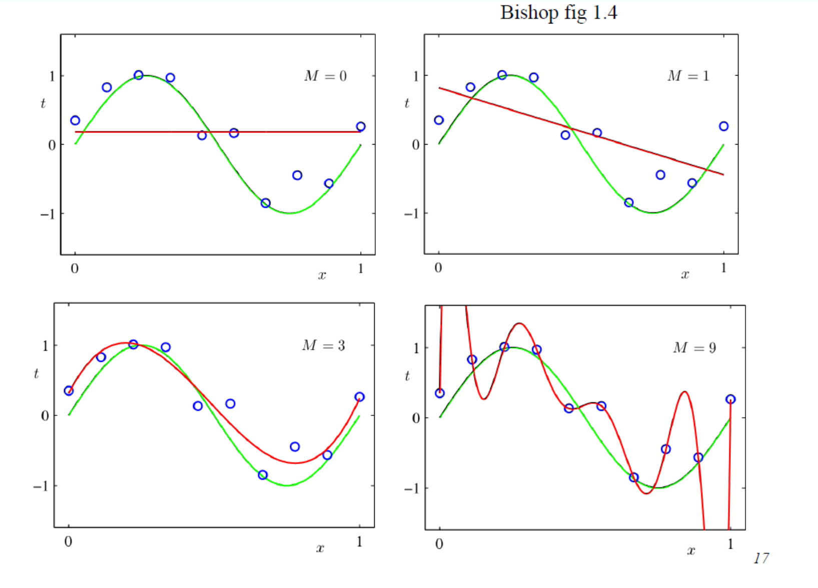
L_2 Regularization, Representer theorem, Kernelization

Thanks to Professor Dale Schuurmans
University of Alberta and Google Brain

Instructor: Farzaneh Mirzazadeh
Department of Computer Science, UCSC, Winter 2017

Overfitting- underfitting



Strategies for preventing overfitting

- Too many features $\implies$ overfitting.
  Hypothesis is too complex for the truth.
- Too few features $\implies$ underfitting.
  Hypothesis is too simple for the truth.

Strategies to avoid

- Model selection
  Choose the right number of features!
  (later)
- **Regularization:** Smooth functions by limiting the size of weights

Regularization

Regularization concept

- Regularization = **Smoothing** learned functions by limiting size of weights (limiting slope of hypothesis function)
- How? By adding a penalty term, **regularizer**, with value proportional to the size of $\mathbf{w}$ to error term:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{t} err(X_{i:}\mathbf{w}, \mathbf{y}_i) + \beta\|\mathbf{w}\|$$

- $\beta \geq 0$ **regularization parameter** (How to set it?)
- Trade-off between minimizing **error** vs **size of $\mathbf{w}$**
- Error function (aka loss function) tries to fit the model to data. Regularizer tries to shrink the weights.
- (Prediction phase is as before. For a test example $x_\circ$ will predict $\hat{y} = x_\circ^T\mathbf{w}$)

Regularization

Important types of regularizers

- How to measure size of $\mathbf{w}$? i.e which norm to use?
  1. L_2 norm squared regularization $\implies$ Representers theorem $\implies$ kernels
     Meaning: Add $\|\mathbf{w}\|_2^2$ term
     This lecture.
  2. L_1 norm regularization $\implies$ Sparsity
     Meaning: Add $\|\mathbf{w}\|_1$ term
     Later in the course.
     (A sparse vector $\mathbf{w}$ will have many zero elements and a few nonzero element. Ignores many features. Only takes into account a small number of good features, i.e. performs feature selection.)

Math background. Recall: Definition of L_P norm of a vector

$$\|\mathbf{a}\|_p = \|[a_1, \ldots, a_n]^\top\|_p = ([a_1|^p + |a_2|^p + \cdots + |a_n|^p)^{1/p}, p \geq 1$$

Example 1: L_2 regularized L_2 error minimization

Fancy name: Ridge Regression

Training Problem

$$\min_{\mathbf{w}} \sum_{i=1}^{t} \|X_{i:}\mathbf{w} - \mathbf{y}_i\|^2 + \beta\|\mathbf{w}\|_2^2$$
$$= \min_{\mathbf{w}} (X\mathbf{w} - \mathbf{y})^\top (X\mathbf{w} - \mathbf{y}) + \beta\, \mathbf{w}^\top \mathbf{w}$$

How to solve?

It has closed-form solution

- Approach similar to least squares. Solve a system of equations.
- Again convex quadratic $\implies$ Any local min is also a global min.
- How to find a local min? Since the objective function (function that you want to optimize) is differentiable. Set the gradient to zero. Find $\mathbf{w}^*$. (Exercise: Derive $\mathbf{w}^*$.) (Derivation next slide)

Example 1: L_2 regularized L_2 error minimization

Fancy name: Ridge Regression

Training Problem

$$\begin{aligned} &\min_{\mathbf{w}} \sum_{i=1}^{t} \|X_{i:}\mathbf{w} - \mathbf{y}_i\|^2 + \beta\|\mathbf{w}\|_2^2 \\ &= \min_{\mathbf{w}} (X\mathbf{w} - \mathbf{y})^\top (X\mathbf{w} - \mathbf{y}) + \beta\,\mathbf{w}^\top\mathbf{w} \end{aligned}$$

How to solve?

It has closed-form solution

- Approach similar to least squares. Solve a system of equations.
- Again convex quadratic $\implies$ Any local min is also a global min.
- How to find a local min? Since the objective function (function that you want to optimize) is differentiable. Set the gradient to zero. Find $\mathbf{w}^*$. (Exercise: Derive $\mathbf{w}^*$.) (Derivation next slide)

Example 1 (con't)

How to solve?

- The function to minimize:

$$J(\mathbf{w}) = (X\mathbf{w} - \mathbf{y})^\top (X\mathbf{w} - \mathbf{y}) + \beta\, \mathbf{w}^\top \mathbf{w}$$

- Computing the gradient and setting it to zero.

$$\nabla_{\mathbf{w}}(J) = 2X^\top (X\mathbf{w} - \mathbf{y}) + 2\beta\mathbf{w} = 0$$
$$(X^\top X + \beta I)\mathbf{w} = X^\top \mathbf{y}$$

How to cancel $(X^\top X + \beta I)$ from the left side?

$$\mathbf{w}^* = (X^\top X + \beta I)^{-1} X^\top \mathbf{y}$$

Matlab code: `w = (X'*X+ beta* I ) \(X'*y)`

- $(X^\top X + \beta I)$ guaranteed to be invertible. Why?

Example 1 (con't)

How to solve?

- The function to minimize:

$$J(\mathbf{w}) = (X\mathbf{w} - \mathbf{y})^\top (X\mathbf{w} - \mathbf{y}) + \beta\, \mathbf{w}^\top \mathbf{w}$$

- Computing the gradient and setting it to zero.

$$\nabla_{\mathbf{w}}(J) = 2X^\top (X\mathbf{w} - \mathbf{y}) + 2\beta\mathbf{w} = 0$$

$$(X^\top X + \beta I)\mathbf{w} = X^\top \mathbf{y}$$

How to cancel $(X^\top X + \beta I)$ from the left side?

$$\mathbf{w}^* = (X^\top X + \beta I)^{-1} X^\top \mathbf{y}$$

Matlab code: `w = (X'*X+ beta* I ) \(X'*y)`

- $(X^\top X + \beta I)$ guaranteed to be invertible. Why?

Example 1 (con't)

How to solve?

- The function to minimize:

$$J(\mathbf{w}) = (X\mathbf{w} - \mathbf{y})^\top (X\mathbf{w} - \mathbf{y}) + \beta\, \mathbf{w}^\top \mathbf{w}$$

- Computing the gradient and setting it to zero.

$$\nabla_{\mathbf{w}}(J) = 2X^\top (X\mathbf{w} - \mathbf{y}) + 2\beta \mathbf{w} = 0$$
$$(X^\top X + \beta I)\mathbf{w} = X^\top \mathbf{y}$$

How to cancel $(X^\top X + \beta I)$ from the left side?

$$\mathbf{w}^* = (X^\top X + \beta I)^{-1} X^\top \mathbf{y}$$

Matlab code: `w = (X'*X+ beta* I ) \(X'*y)`

- $(X^\top X + \beta I)$ guaranteed to be invertible. Why?

Example 1 (con't)

How to solve?

- The function to minimize:

$$J(\mathbf{w}) = (X\mathbf{w} - \mathbf{y})^\top (X\mathbf{w} - \mathbf{y}) + \beta\, \mathbf{w}^\top \mathbf{w}$$

- Computing the gradient and setting it to zero.

$$\nabla_{\mathbf{w}}(J) = 2X^\top (X\mathbf{w} - \mathbf{y}) + 2\beta\mathbf{w} = 0$$
$$(X^\top X + \beta I)\mathbf{w} = X^\top \mathbf{y}$$

How to cancel $(X^\top X + \beta I)$ from the left side?

$$\mathbf{w}^* = (X^\top X + \beta I)^{-1} X^\top \mathbf{y}$$

Matlab code: `w = (X'*X+ beta* I ) \(X'*y)`

- $(X^\top X + \beta I)$ guaranteed to be invertible. Why?

Example 1 (con't)

How to solve?

- The function to minimize:

$$J(\mathbf{w}) = (X\mathbf{w} - \mathbf{y})^\top (X\mathbf{w} - \mathbf{y}) + \beta\, \mathbf{w}^\top \mathbf{w}$$

- Computing the gradient and setting it to zero.

$$\nabla_{\mathbf{w}}(J) = 2X^\top (X\mathbf{w} - \mathbf{y}) + 2\beta \mathbf{w} = 0$$
$$(X^\top X + \beta I)\mathbf{w} = X^\top \mathbf{y}$$

  How to cancel $(X^\top X + \beta I)$ from the left side?

$$\mathbf{w}^* = (X^\top X + \beta I)^{-1} X^\top \mathbf{y}$$

  Matlab code: w = (X'*X+ beta* I) \(X'*y)

- $(X^\top X + \beta I)$ guaranteed to be invertible. Why?

Example 1 (con't)

How to solve?

- The function to minimize:

$$J(\mathbf{w}) = (X\mathbf{w} - \mathbf{y})^\top (X\mathbf{w} - \mathbf{y}) + \beta\, \mathbf{w}^\top \mathbf{w}$$

- Computing the gradient and setting it to zero.

$$\nabla_{\mathbf{w}}(J) = 2X^\top (X\mathbf{w} - \mathbf{y}) + 2\beta\mathbf{w} = 0$$
$$(X^\top X + \beta I)\mathbf{w} = X^\top \mathbf{y}$$

How to cancel $(X^\top X + \beta I)$ from the left side?

$$\mathbf{w}^* = (X^\top X + \beta I)^{-1} X^\top \mathbf{y}$$

Matlab code: `w = (X'*X+ beta* I ) \(X'*y)`

- $(X^\top X + \beta I)$ guaranteed to be invertible. Why?

Example 1 (con't)

How to solve?

- The function to minimize:

$$J(\mathbf{w}) = (X\mathbf{w} - \mathbf{y})^\top (X\mathbf{w} - \mathbf{y}) + \beta\, \mathbf{w}^\top \mathbf{w}$$

- Computing the gradient and setting it to zero.

$$\nabla_{\mathbf{w}}(J) = 2X^\top (X\mathbf{w} - \mathbf{y}) + 2\beta \mathbf{w} = 0$$
$$(X^\top X + \beta I)\mathbf{w} = X^\top \mathbf{y}$$

  How to cancel $(X^\top X + \beta I)$ from the left side?

$$\mathbf{w}^* = (X^\top X + \beta I)^{-1} X^\top \mathbf{y}$$

  Matlab code: `w = (X'*X+ beta* I ) \(X'*y)`

- $(X^\top X + \beta I)$ guaranteed to be invertible. Why?

Example 2: L_2 regularized L_∞ error minimization

$$\min_{\mathbf{w}} \max_{i=1}^{t} \|X_{i:}\mathbf{w} - \mathbf{y}_i\| + \beta\|\mathbf{w}\|_2^2$$

How to solve?

$$\min_{\mathbf{w},\delta} \delta + \beta\mathbf{w}^\top\mathbf{w}$$

subject to

$$X\mathbf{w} \leq \mathbf{y} + \delta\mathbf{1}$$
$$X\mathbf{w} \geq \mathbf{y} - \delta\mathbf{1}$$

Convex quadratic program. Still reasonably efficient. quadprog() in Matlab

Optimization background:

What is a quadratic programming problem?
An optimization problem, with quadratic objective function and linear constraint.

Example 2: L_2 regularized L_∞ error minimization

$$\min_{\mathbf{w}} \max_{i=1}^{t} \|X_{i:}\mathbf{w} - \mathbf{y}_i\| + \beta\|\mathbf{w}\|_2^2$$

How to solve?

$$\min_{\mathbf{w},\delta} \delta + \beta\mathbf{w}^\top\mathbf{w}$$

subject to

$$X\mathbf{w} \leq \mathbf{y} + \delta\mathbf{1}$$
$$X\mathbf{w} \geq \mathbf{y} - \delta\mathbf{1}$$

Convex quadratic program. Still reasonably efficient. quadprog() in Matlab

Optimization background:

What is a quadratic programming problem?
An optimization problem, with quadratic objective function and linear constraint.

Example 2: L_2 regularized L_∞ error minimization

$$\min_{\mathbf{w}} \max_{i=1}^{t} \|X_{i:}\mathbf{w} - \mathbf{y}_i\| + \beta\|\mathbf{w}\|_2^2$$

How to solve?

$$\min_{\mathbf{w},\delta} \delta + \beta \mathbf{w}^\top \mathbf{w}$$

subject to

$$X\mathbf{w} \le \mathbf{y} + \delta\mathbf{1}$$
$$X\mathbf{w} \ge \mathbf{y} - \delta\mathbf{1}$$

Convex quadratic program. Still reasonably efficient. quadprog() in Matlab

Optimization background:

What is a quadratic programming problem?
An optimization problem, with quadratic objective function and linear constraint.

Example 3: L_2 regularized L_1 error minimization

$$\min_{\mathbf{w}} \sum_{i=1}^{t} |X_{i:}\mathbf{w} - \mathbf{y}_i| + \beta \|\mathbf{w}\|_2^2$$

How to solve?

$$\min_{\mathbf{w},\boldsymbol{\delta}} \mathbf{1}^\top \boldsymbol{\delta} + \beta \mathbf{w}^\top \mathbf{w}$$

subject to

$$X\mathbf{w} \leq \mathbf{y} + \boldsymbol{\delta}$$
$$X\mathbf{w} \geq \mathbf{y} - \boldsymbol{\delta}$$

Convex quadratic program. Still reasonably efficient.

Optimization note:

The same trick for re-expressing max and absolute value functions to linear forms repeats frequently. Get comfortable with it and practice it!

Example 3: L_2 regularized L_1 error minimization

$$\min_{\mathbf{w}} \sum_{i=1}^{t} |X_{i:}\mathbf{w} - \mathbf{y}_i| + \beta\|\mathbf{w}\|_2^2$$

How to solve?

$$\min_{\mathbf{w},\boldsymbol{\delta}} \mathbf{1}^\top \boldsymbol{\delta} + \beta \mathbf{w}^\top \mathbf{w}$$

subject to

$$X\mathbf{w} \leq \mathbf{y} + \boldsymbol{\delta}$$
$$X\mathbf{w} \geq \mathbf{y} - \boldsymbol{\delta}$$

Convex quadratic program. Still reasonably efficient.

Optimization note:

The same trick for re-expressing max and absolute value functions to linear forms repeats frequently. Get comfortable with it and practice it!

Representer Theorem

A consequence of L_2 regularization that leads to kernels

Representer Theorem

For any loss function $l(\hat{y}, y) = err(\hat{y}, y)$ and for the optimal solution of training problem:

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \sum_{i=1}^{\top} l(X_{i:}\mathbf{w}, \mathbf{y}) + \beta \|\mathbf{w}\|_2^2$$

$\mathbf{w}^*$ satisfies $\mathbf{w}^* = X^\top \boldsymbol{\alpha}^*, \qquad$ for some $\boldsymbol{\alpha}^*$.

- In other words, the optimal $\mathbf{w}^*$ is actually a weighted average of training examples. Each element of α^* explains how much the corresponding training example contributes.
- Leads to *instance-based learning*: Learn weights for each data example instead of each feature

Proof of Representer Theorem ($\mathbf{w}^* \in rowSpan(X)$)

Representer Theorem

For any loss function $l(\hat{y}, y) = err(\hat{y}, y)$ and for the optimal solution of training problem, $\mathbf{w}^* = \text{argmin}_{\mathbf{w}} \sum_{i=1}^{\top} l(X_{i:}\mathbf{w}, \mathbf{y}) + \beta\|\mathbf{w}\|_2^2$, $\mathbf{w}^*$ satisfies $\mathbf{w}^* = X^\top \boldsymbol{\alpha}^*$, for some $\boldsymbol{\alpha}^*$.

Proof (continued in next slide)

1. Note: Any $\mathbf{w}$ can be decomposed as $\mathbf{w} = \mathbf{w}_0 + \mathbf{w}_1$ where $\mathbf{w}_1 \in rowSpan(X)$ and $\mathbf{w}_0 \perp rowSpan(X)$.
2. Suppose the optimal solution $\mathbf{w}^*$ is not in the rowSpan(X). Then it must have a component $\mathbf{w}_1$ in the row span and a component $\mathbf{w}_0$ perpendicular to the row span of X: $\mathbf{w}^* = \mathbf{w}_0^* + \mathbf{w}_1^*$, $\mathbf{w}_0^* \neq 0$.
3. Given $\mathbf{w}^* = \mathbf{w}_0^* + \mathbf{w}_1^*$,
   1. Expand the L_2 regularization term:

      $$\mathbf{w}^{*\top}\mathbf{w}^* = (\mathbf{w}_0^* + \mathbf{w}_1^*)^\top(\mathbf{w}_0^* + \mathbf{w}_1^*) = \|\mathbf{w}_0^*\|_2^2 + \|\mathbf{w}_1^*\|_2^2 + 2\mathbf{w}_0^{*\top}\mathbf{w}_1^* = \|\mathbf{w}_0^*\|_2^2 + \|\mathbf{w}_1^*\|_2^2$$

   2. Expand prediction term: $X_{i:}\mathbf{w}^* = X_{i:}\mathbf{w}_1^* + X_{i:}\mathbf{w}_0^* = X_{i:}\mathbf{w}^*$

Note that the values in red are zero, due to orthogonality

Proof of Representer Theorem ($\mathbf{w}^* \in rowSpan(X)$)

Representer Theorem

For any loss function $l(\hat{y}, y) = err(\hat{y}, y)$ and for the optimal solution of training problem, $\mathbf{w}^* = \text{argmin}_{\mathbf{w}} \sum_{i=1}^{\top} l(X_{i:}\mathbf{w}, \mathbf{y}) + \beta\|\mathbf{w}\|_2^2$, $\mathbf{w}^*$ satisfies $\mathbf{w}^* = X^\top \boldsymbol{\alpha}^*$, for some $\boldsymbol{\alpha}^*$.

Proof (continued in next slide)

1. Note: Any $\mathbf{w}$ can be decomposed as $\mathbf{w} = \mathbf{w}_0 + \mathbf{w}_1$ where $\mathbf{w}_1 \in rowSpan(X)$ and $\mathbf{w}_0 \perp rowSpan(X)$.
2. Suppose the optimal solution $\mathbf{w}^*$ is not in the rowSpan(X). Then it must have a component $\mathbf{w}_1$ in the row span and a component $\mathbf{w}_0$ perpendicular to the row span of X: $\mathbf{w}^* = \mathbf{w}_0^* + \mathbf{w}_1^*, \mathbf{w}_0^* \neq 0$.
3. Given $\mathbf{w}^* = \mathbf{w}_0^* + \mathbf{w}_1^*$,
    1. Expand the L_2 regularization term:

    $$\mathbf{w}^{*\top}\mathbf{w}^* = (\mathbf{w}_0^* + \mathbf{w}_1^*)^\top(\mathbf{w}_0^* + \mathbf{w}_1^*) = \|\mathbf{w}_0^*\|_2^2 + \|\mathbf{w}_1^*\|_2^2 + 2\mathbf{w}_0^{*\top}\mathbf{w}_1^* = \|\mathbf{w}_0^*\|_2^2 + \|\mathbf{w}_1^*\|_2^2$$

    2. Expand prediction term: $X_{i:}\mathbf{w}^* = X_{i:}\mathbf{w}_1^* + X_{i:}\mathbf{w}_0^* = X_{i:}\mathbf{w}^*$

Note that the values in red are zero, due to orthogonality

Proof of Representer Theorem ($\mathbf{w}^* \in rowSpan(X)$)

Representer Theorem

For any loss function $l(\hat{y}, y) = err(\hat{y}, y)$ and for the optimal solution of training problem, $\mathbf{w}^* = \text{argmin}_{\mathbf{w}} \sum_{i=1}^{\top} l(X_{i:}\mathbf{w}, \mathbf{y}) + \beta\|\mathbf{w}\|_2^2$, $\mathbf{w}^*$ satisfies $\mathbf{w}^* = X^\top \boldsymbol{\alpha}^*, \qquad$ for some $\boldsymbol{\alpha}^*$.

Proof (continued in next slide)

1. Note: Any $\mathbf{w}$ can be decomposed as $\mathbf{w} = \mathbf{w}_0 + \mathbf{w}_1$ where $\mathbf{w}_1 \in rowSpan(X)$ and $\mathbf{w}_0 \perp rowSpan(X)$.
2. Suppose the optimal solution $\mathbf{w}^*$ is not in the rowSpan(X). Then it must have a component $\mathbf{w}_1$ in the row span and a component $\mathbf{w}_0$ perpendicular to the row span of X: $\qquad \mathbf{w}^* = \mathbf{w}_0^* + \mathbf{w}_1^*, \mathbf{w}_0^* \neq 0$.
3. Given $\mathbf{w}^* = \mathbf{w}_0^* + \mathbf{w}_1^*$,
   1. Expand the L_2 regularization term:

   $$\mathbf{w}^{*\top}\mathbf{w}^* = (\mathbf{w}_0^* + \mathbf{w}_1^*)^\top(\mathbf{w}_0^* + \mathbf{w}_1^*) = \|\mathbf{w}_0^*\|_2^2 + \|\mathbf{w}_1^*\|_2^2 + 2\mathbf{w}_0^{*\top}\mathbf{w}_1^* = \|\mathbf{w}_0^*\|_2^2 + \|\mathbf{w}_1^*\|_2^2$$

   2. Expand prediction term: $X_{i:}\mathbf{w}^* = X_{i:}\mathbf{w}_1^* + X_{i:}\mathbf{w}_0^* = X_{i:}\mathbf{w}^*$

   Note that the values in red are zero, due to orthogonality

Proof of representation theorem (con't)

4. Therefor:

$$\sum_{i=1}^{t} l(X_{i:}\mathbf{w}^*, \mathbf{y}) + \beta\|\mathbf{w}^*\|_2^2 = \sum_{i=1}^{t} l(X_{i:}\mathbf{w}_1^*, \mathbf{y}) + \beta(\|\mathbf{w}_0^*\|_2^2 + \|\mathbf{w}_1^*\|_2^2)$$

$$> \sum_{i=1}^{t} l(X_{i:}\mathbf{w}_1^*, \mathbf{y}) + \beta(\|\mathbf{w}_1^*\|_2^2).$$

Contradiction!! w^* is a minimizer (attains smallest objective value). But now $\mathrm{w}_1^* \neq \mathrm{w}^*$ gets strictly better value than w unless $\mathrm{w}_0^* = 0$.
Conclusion: w_0^* must be zero. So w must be in rowSpan of X $\square$.

Proof of representation theorem (con't)

4. Therefor:

$$\sum_{i=1}^{t} l(X_{i:}\mathbf{w}^*, \mathbf{y}) + \beta\|\mathbf{w}^*\|_2^2 = \sum_{i=1}^{t} l(X_{i:}\mathbf{w}_1^*, \mathbf{y}) + \beta(\|\mathbf{w}_0^*\|_2^2 + \|\mathbf{w}_1^*\|_2^2)$$

$$> \sum_{i=1}^{t} l(X_{i:}\mathbf{w}_1^*, \mathbf{y}) + \beta(\|\mathbf{w}_1^*\|_2^2).$$

Contradiction!! $\mathbf{w}^*$ is a minimizer (attains smallest objective value). But now $\mathbf{w}_1^* \neq \mathbf{w}^*$ gets strictly better value than $\mathbf{w}$ unless $\mathbf{w}_0^* = 0$.
Conclusion: $\mathbf{w}_0^*$ must be zero. So $\mathbf{w}$ must be in rowSpan of X $\square$.

Duality

Can solve for example weights $\boldsymbol{\alpha}$ instead of feature weights $\mathbf{w}$.

Primal (one way of looking at the learning problem)

- Training $\quad \min_{\mathbf{w}} \sum_{i=1}^{t} l(X_{i:}\mathbf{w}, \mathbf{y}) + \beta\|\mathbf{w}^*\|_2^2$
- Prediction: Given $\mathbf{x}_\circ$, predict $\hat{y} = {\mathbf{w}^*}^\top \mathbf{x}_\circ$

Finds feature weights. Predicts using feature weights.

Dual (another equivalent way of looking at the problem

- Training $\quad \min_{\boldsymbol{\alpha}} \sum_{i=1}^{t} l(X_{i:}X_{i:}^\top \boldsymbol{\alpha}, \mathbf{y}_i) + \beta \boldsymbol{\alpha}^\top X X^\top \boldsymbol{\alpha}$
- Prediction: Given $\mathbf{x}_\circ$, predict $\hat{y} = {\boldsymbol{\alpha}^*}^\top X \mathbf{x}_\circ$

- Equivalent predictors!
- Note: both dual training and dual prediction does not need feature vectors $X_{i:}$ explicitly. Because everywhere X appears as $XX^\top$.
- Only need inner product between feature vectors.

Duality

Can solve for example weights $\boldsymbol{\alpha}$
instead of feature weights $\mathbf{w}$.

Primal (one way of looking at the learning problem)

- Training $\quad \min_{\mathbf{w}} \sum_{i=1}^{t} l(X_{i:}\mathbf{w}, \mathbf{y}) + \beta \|\mathbf{w}^*\|_2^2$
- Prediction: Given $\mathbf{x}_\circ$, predict $\hat{y} = {\mathbf{w}^*}^\top \mathbf{x}_\circ$

Finds feature weights. Predicts using feature weights.

Dual (another equivalent way of looking at the problem

- Training $\quad \min_{\boldsymbol{\alpha}} \sum_{i=1}^{t} l(X_{i:}X_{i:}^\top \boldsymbol{\alpha}, \mathbf{y}_i) + \beta \boldsymbol{\alpha}^\top X X^\top \boldsymbol{\alpha}$
- Prediction: Given $\mathbf{x}_\circ$, predict $\hat{y} = {\boldsymbol{\alpha}^*}^\top X \mathbf{x}_\circ$

- Equivalent predictors!
- Note: both dual training and dual prediction does not need feature vectors $X_{i:}$ explicitly. Because everywhere X appears as $XX^\top$.
- Only need inner product between feature vectors.

Kernelization

- Typically with a feature expansion $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_d(\mathbf{x})]^\top$ in linear prediction want to obtain best linear predictor over feature representation $h(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$
- Amazing fact about kernelization: do no need $\mathbf{w}$ no matter how many features there are!
- Instead we need only optimize for vector $\boldsymbol{\alpha}$ whose size is n (number of examples).
- So-called instance based learning!
- Computational advantage when the number of features is large.

Kernelization

Kernelized training

Kernelized Training

$$\alpha^* = \underset{\alpha}{\text{argmin}} \sum_{i=1}^{t} l(K_{i:}\boldsymbol{\alpha}, \mathbf{y}_i) + \beta\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}$$

where K_{ij} is the inner product between the feature expansions of the i th and jth examples.

Kernelized prediction

Prediction: Given $\mathbf{k}_\circ$, predict $\hat{y} = \boldsymbol{\alpha}^{*\top}\mathbf{k}_\circ$.
Note: $\mathbf{k}_\circ$ holds the similarity (in terms of inner product) of the test example to all training examples. $\mathbf{k}_i = X_{i:}^\top x_\circ$.

Only needs kernel values!!! Nice!

Example Kernelized Ridge Regression Training

Kernelized Training

Recall: Ridge regression is nothing except L_2 error + L_2 regularizer

$$\min_{\boldsymbol{\alpha}} (K\boldsymbol{\alpha} - \mathbf{y})^\top (K\boldsymbol{\alpha} - \mathbf{y}) + \beta \boldsymbol{\alpha}^\top K \boldsymbol{\alpha}$$

- Analytical solution. System of linear equations.
- Excercise: Derive the solution $\boldsymbol{\alpha}^*$

Example: polynomial kernel

Suppose we have the feature expansion below:
$\phi(x) = [\sqrt{\binom{d}{0}}, \sqrt{\binom{d}{1}}x, \sqrt{\binom{d}{2}}x^2, \sqrt{\binom{d}{3}}x^3, \ldots, \sqrt{\binom{d}{d}}x^d]$, d a large number

- With kernelized form mentioned so far, training and prediction does not depend on d. Train and test time complexity only depends on the number of examples only.
- How about forming the matrix K?
- Computational complexity challenge!
- Does it need computing inner product of two feature vectors?
- Any more efficient way of computing K? Any trick coming to mind?

- $\phi(x_1)^\top \phi(x_2) = \binom{d}{0} + \binom{d}{1}x_1x_2 + \binom{d}{1}x_1^2x_2^2 + \cdots + \binom{d}{d}x_1^dx_2^d$
- $$k(x_1, x_2) = (x_1x_2 + 1)^d = \sum_{i=1}^{d} \binom{d}{i} (x_1x_2)^i = \phi(\mathbf{x}_1)^\top \phi(\mathbf{x}_2)$$
- This is called Polynomial Kernel.

Example: polynomial kernel

Suppose we have the feature expansion below:
$\phi(x) = [\sqrt{\binom{d}{0}}, \sqrt{\binom{d}{1}}x, \sqrt{\binom{d}{2}}x^2, \sqrt{\binom{d}{3}}x^3, \dots, \sqrt{\binom{d}{d}}x^d]$, d a large number

- With kernelized form mentioned so far, training and prediction does not depend on d. Train and test time complexity only depends on the number of examples only.
- How about forming the matrix K?
- Computational complexity challenge!
- Does it need computing inner product of two feature vectors?
- Any more efficient way of computing K? Any trick coming to mind?

- $\phi(x_1)^\top \phi(x_2) = \binom{d}{0} + \binom{d}{1} x_1 x_2 + \binom{d}{1} x_1^2 x_2^2 + \cdots + \binom{d}{d} x_1^d x_2^d$
-
$$k(x_1, x_2) = (x_1 x_2 + 1)^d = \sum_{i=1}^{d} \binom{d}{i} (x_1 x_2)^i = \phi(\mathbf{x}_1)^\top \phi(\mathbf{x}_2)$$

- This is called Polynomial Kernel.

Example: polynomial kernel

Suppose we have the feature expansion below:

$\phi(x) = [\sqrt{\binom{d}{0}}, \sqrt{\binom{d}{1}}x, \sqrt{\binom{d}{2}}x^2, \sqrt{\binom{d}{3}}x^3, \ldots, \sqrt{\binom{d}{d}}x^d]$, d a large number

- With kernelized form mentioned so far, training and prediction does not depend on d. Train and test time complexity only depends on the number of examples only.
- How about forming the matrix K?
- Computational complexity challenge!
- Does it need computing inner product of two feature vectors?
- Any more efficient way of computing K? Any trick coming to mind?

- $\phi(x_1)^\top \phi(x_2) = \binom{d}{0} + \binom{d}{1}x_1x_2 + \binom{d}{1}x_1^2x_2^2 + \cdots + \binom{d}{d}x_1^dx_2^d$
- $$k(x_1, x_2) = (x_1x_2 + 1)^d = \sum_{i=1}^{d} \binom{d}{i} (x_1x_2)^i = \phi(\mathbf{x}_1)^\top \phi(\mathbf{x}_2)$$
- This is called Polynomial Kernel.

Kernel trick

Kernel trick

Use a function $k(.,.)$ to compute $k(x_i, x_j)$ directly!

- i.e. without expanding the features, multiplying and adding up
- Constant time computation for any d.
- Kernel functions directly measure similarity between feature representations

Kernel Trick

- Kernel functions directly measure similarity between data examples feature representations
- Can just choose kernel functions directly (ignoring feature representations)
- Background knowledge could be put into kenel functions
- $k(.,.)$ has to satisfy certain properties: symmetry, finitely positive semidefinite

Example: RBF Kernels

- Radial basis function kernel

$$k(\mathbf{x}_1, \mathbf{x}_2) = e^{\frac{-\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2}}$$

- Corresponds to the inner product of an infinite dimensional feature map

Where to read?

- Ridge regression: Hastie et al, Sec 3.4.1
- Regularization: Hastie et al 5.8, Shalev-Shwartz and Ben David (2014) 13.1
- Kernels: Shalev-Shwartz and Ben David (2014), Sec 16.1, 16.2 (Free online, very short)
- Bishop (2006), Sec 6.1
- Shaw-Taylor and Cristinani (2004), Sec 2.2

Simple Background Notes Useful in Computing Gradients

- Dimension of gradient of a function f with respect to a vector $\mathbf{w}$, $\nabla_{\mathbf{w}} f$ must be equal to dimension of the input vector $\mathbf{w}$.
- Two matrices can only be multiplied if their dimensions match in this way: ($d_1 \times d_2$ matrix) ($d_2 \times d_3$ matrix) = $d_1 \times d_3$ matrix.
- Use chain rule.
- Check vector calculus identities