Controling Complexity: Feature Selection and Regularization

Based on David Rosenberg's and He He's materials

Tal Linzen

CDS, NYU

Feb 8, 2022

Complexity of Hypothesis Spaces

What is the trade-off between approximation error and estimation error?

Complexity of Hypothesis Spaces

What is the trade-off between approximation error and estimation error?

- Bigger $\mathcal{F}$: better approximation but can overfit
- Smaller $\mathcal{F}$: less likely to overfit but can be farther from the Bayes prediction function

Complexity of Hypothesis Spaces

What is the trade-off between approximation error and estimation error?

- Bigger $\mathcal{F}$: better approximation but can overfit
- Smaller $\mathcal{F}$: less likely to overfit but can be farther from the Bayes prediction function

To control the "size" of $\mathcal{F}$, we need some measure of its complexity:

- Number of variables / features
- Degree of polynomial

General Approach to Control Complexity

1. Learn a sequence of models varying in complexity from the training data

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_n \cdots \subset \mathcal{F}$$

General Approach to Control Complexity

1. Learn a sequence of models varying in complexity from the training data

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_n \cdots \subset \mathcal{F}$$

Example: Polynomial Functions

- $\mathcal{F} = \{\text{all polynomial functions}\}$
- $\mathcal{F}_d = \{\text{all polynomials of degree } \leqslant d\}$

General Approach to Control Complexity

1. Learn a sequence of models varying in complexity from the training data

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_n \cdots \subset \mathcal{F}$$

   Example: Polynomial Functions

   - $\mathcal{F} = \{\text{all polynomial functions}\}$
   - $\mathcal{F}_d = \{\text{all polynomials of degree } \leqslant d\}$

2. Select one of these models based on a score (e.g. validation error)

Feature Selection in Linear Regression

Nested sequence of hypothesis spaces: $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_n \cdots \subset \mathcal{F}$

- $\mathcal{F} = \{$linear functions using all features$\}$
- $\mathcal{F}_d = \{$linear functions using fewer than d features$\}$

Feature Selection in Linear Regression

Nested sequence of hypothesis spaces: $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_n \cdots \subset \mathcal{F}$

- $\mathcal{F} = \{$linear functions using all features$\}$
- $\mathcal{F}_d = \{$linear functions using fewer than d features$\}$

Best subset selection:

- Choose the subset of features that is best according to the score (e.g. validation error)
  - Example with two features: Train models using $\{\}$, $\{X_1\}$, $\{X_2\}$, $\{X_1, X_2\}$, respectively
- No efficient search algorithm; iterating over all subsets becomes very expensive with a large number of features

Complexity Penalty

An objective that balances number of features and prediction performance:

$$\text{score}(S) = \text{training\_loss}(S) + \lambda |S| \tag{1}$$

Complexity Penalty

An objective that balances number of features and prediction performance:

$$\text{score}(S) = \text{training\_loss}(S) + \lambda |S| \tag{1}$$

λ balances the training loss and the number of features used:

- Adding an extra feature must be justified by at least λ improvement in training loss
- Larger $\lambda \to$ complex models are penalized more heavily
- Criteria such as AIC and BIC use different values of λ

Greedy Selection Methods

Forward selection:

1. Start with an empty set of features S

Greedy Selection Methods

Forward selection:

1. Start with an empty set of features S
2. For each feature i not in S
   - Learn a model using features $S \cup i$
   - Compute score of the model: α_i

Greedy Selection Methods

Forward selection:

1. Start with an empty set of features S
2. For each feature i not in S
   - Learn a model using features $S \cup i$
   - Compute score of the model: α_i
3. Find the candidate feature with the highest score: $j = \arg\max_i \alpha_i$

Greedy Selection Methods

Forward selection:

1. Start with an empty set of features S
2. For each feature i not in S
   - Learn a model using features $S \cup i$
   - Compute score of the model: α_i
3. Find the candidate feature with the highest score: $j = \arg\max_i \alpha_i$
4. If α_j improves the current best score, add feature j: $S \leftarrow S \cup j$ and go to step 2; return S otherwise.

Greedy Selection Methods

Forward selection:

1. Start with an empty set of features S
2. For each feature i not in S
   - Learn a model using features $S \cup i$
   - Compute score of the model: α_i
3. Find the candidate feature with the highest score: $j = \arg\max_i \alpha_i$
4. If α_j improves the current best score, add feature j: $S \leftarrow S \cup j$ and go to step 2; return S otherwise.

Backward Selection:

- Start with all features; in each iteration, remove the worst feature

Feature Selection: Discussion

- Number of features as a measure of the complexity of a linear prediction function

Feature Selection: Discussion

- Number of features as a measure of the complexity of a linear prediction function
- General approach to feature selection:

Feature Selection: Discussion

- Number of features as a measure of the complexity of a linear prediction function
- General approach to feature selection:
  - Define a score that balances training error and complexity

Feature Selection: Discussion

- Number of features as a measure of the complexity of a linear prediction function
- General approach to feature selection:
  - Define a score that balances training error and complexity
  - Find the subset of features that maximizes the score

Feature Selection: Discussion

- Number of features as a measure of the complexity of a linear prediction function
- General approach to feature selection:
  - Define a score that balances training error and complexity
  - Find the subset of features that maximizes the score
- The search problem over 2^k feature subsets is hard

Feature Selection: Discussion

- Number of features as a measure of the complexity of a linear prediction function
- General approach to feature selection:
  - Define a score that balances training error and complexity
  - Find the subset of features that maximizes the score
- The search problem over 2^k feature subsets is hard
- Forward selection is often used in practice, but isn't guaranteed to find the best solution

Feature Selection: Discussion

- Number of features as a measure of the complexity of a linear prediction function
- General approach to feature selection:
  - Define a score that balances training error and complexity
  - Find the subset of features that maximizes the score
- The search problem over 2^k feature subsets is hard
- Forward selection is often used in practice, but isn't guaranteed to find the best solution
- Forward and backward selection do not in general result in the same subset

ℓ_2 and ℓ_1 Regularization

Complexity Penalty

Goal: Balance the complexity of the hypothesis space $\mathcal{F}$ and the training loss

Complexity measure: $\Omega : \mathcal{F} \to [0, \infty)$, e.g. number of features

Complexity Penalty

Goal: Balance the complexity of the hypothesis space $\mathcal{F}$ and the training loss

Complexity measure: $\Omega : \mathcal{F} \to [0, \infty)$, e.g. number of features

Penalized ERM (Tikhonov regularization)

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

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

As usual, we find λ using the validation data.

Complexity Penalty

Goal: Balance the complexity of the hypothesis space $\mathcal{F}$ and the training loss

Complexity measure: $\Omega : \mathcal{F} \to [0, \infty)$, e.g. number of features

Penalized ERM (Tikhonov regularization)

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

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

As usual, we find λ using the validation data.

Number of features as complexity measure is hard to optimize—other measures?

Weight Shrinkage: Intuition

- (Draw on board)

Weight Shrinkage: Intuition

- (Draw on board)
- Why would we prefer a regression line with smaller slope (unless the data strongly supports a larger slope)?

Weight Shrinkage: Intuition

- (Draw on board)
- Why would we prefer a regression line with smaller slope (unless the data strongly supports a larger slope)?
- More conservative: small change in the input does not cause large change in the output

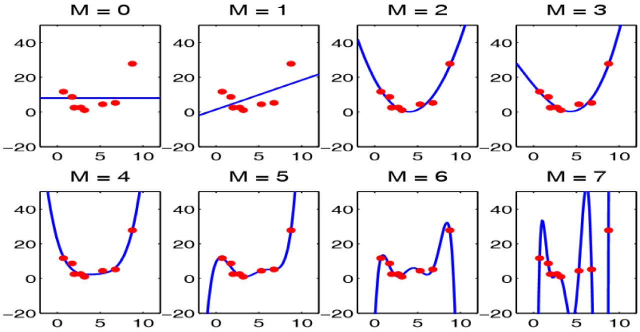
Weight Shrinkage: Intuition

- (Draw on board)
- Why would we prefer a regression line with smaller slope (unless the data strongly supports a larger slope)?
- More conservative: small change in the input does not cause large change in the output
- If we push the estimated weights to be small, re-estimating them on a new dataset wouldn't cause the prediction function to change dramatically (less sensitive to noise in data)

Weight Shrinkage: Intuition

- (Draw on board)
- Why would we prefer a regression line with smaller slope (unless the data strongly supports a larger slope)?
- More conservative: small change in the input does not cause large change in the output
- If we push the estimated weights to be small, re-estimating them on a new dataset wouldn't cause the prediction function to change dramatically (less sensitive to noise in data)
- Bayesian intuition: pull the regression weights towards a prior centered at 0

Weight Shrinkage: Polynomial Regression



- Large weights are needed to make the curve wiggle sufficiently to overfit the data
- $\hat{y} = 0.001x^7 + 0.003x^3 + 1$ less likely to overfit than $\hat{y} = 1000x^7 + 500x^3 + 1$

(Adapated from Mark Schmidt's slide)

Linear Regression with ℓ_2 Regularization

- We have a linear model

$$\mathcal{F} = \left\{ f : \mathbb{R}^d \to \mathbb{R} \mid f(x) = w^T x \text{ for } w \in \mathbb{R}^d \right\}$$

- Square loss: $\ell(\hat{y}, y) = (y - \hat{y})^2$
- Training data $\mathcal{D}_n = ((x_1, y_1), \ldots, (x_n, y_n))$

Linear Regression with ℓ_2 Regularization

- We have a linear model

$$\mathcal{F} = \left\{ f : \mathbf{R}^d \to \mathbf{R} \mid f(x) = w^T x \text{ for } w \in \mathbf{R}^d \right\}$$

- Square loss: $\ell(\hat{y}, y) = (y - \hat{y})^2$
- Training data $\mathcal{D}_n = ((x_1, y_1), \ldots, (x_n, y_n))$
- Linear least squares regression is ERM for square loss over $\mathcal{F}$:

$$\hat{w} = \underset{w \in \mathbf{R}^d}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

- This often overfits, especially when d is large compared to n (e.g. in NLP one can have 1M features for 10K documents).

Linear Regression with L2 Regularization

Penalizes large weights:

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

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

- Also known as **ridge regression**.

Linear Regression with L2 Regularization

Penalizes large weights:

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

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

- Also known as **ridge regression**.
- Equivalent to linear least square regression when $\lambda = 0$.

Linear Regression with L2 Regularization

Penalizes large weights:

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

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

- Also known as **ridge regression**.
- Equivalent to linear least square regression when $\lambda = 0$.
- ℓ_2 regularization can be used for other models too (e.g. neural networks).

ℓ_2 regularization reduces sensitivity to changes in input

- $\hat{f}(x) = \hat{w}^T x$ is **Lipschitz continuous** with Lipschitz constant $L = \|\hat{w}\|_2$: when moving from x to $x + h$, $\hat{f}$ changes no more than $L\|h\|$.

ℓ_2 regularization reduces sensitivity to changes in input

- $\hat{f}(x) = \hat{w}^T x$ is **Lipschitz continuous** with Lipschitz constant $L = \|\hat{w}\|_2$: when moving from x to $x+h$, $\hat{f}$ changes no more than $L\|h\|$.
- ℓ_2 regularization controls the maximum rate of change of $\hat{f}$.

ℓ_2 regularization reduces sensitivity to changes in input

- $\hat{f}(x) = \hat{w}^T x$ is **Lipschitz continuous** with Lipschitz constant $L = \|\hat{w}\|_2$: when moving from x to $x + h$, $\hat{f}$ changes no more than $L\|h\|$.
- ℓ_2 regularization controls the maximum rate of change of $\hat{f}$.
- Proof:

$$\begin{aligned}\left|\hat{f}(x+h) - \hat{f}(x)\right| &= |\hat{w}^T(x+h) - \hat{w}^T x| = \left|\hat{w}^T h\right| \\ &\leqslant \|\hat{w}\|_2 \|h\|_2 \quad \text{(Cauchy-Schwarz inequality)}\end{aligned}$$

ℓ_2 regularization reduces sensitivity to changes in input

- $\hat{f}(x) = \hat{w}^T x$ is **Lipschitz continuous** with Lipschitz constant $L = \|\hat{w}\|_2$: when moving from x to $x+h$, $\hat{f}$ changes no more than $L\|h\|$.
- ℓ_2 regularization controls the maximum rate of change of $\hat{f}$.
- Proof:

$$\begin{aligned} \left|\hat{f}(x+h) - \hat{f}(x)\right| &= |\hat{w}^T(x+h) - \hat{w}^T x| = \left|\hat{w}^T h\right| \\ &\leqslant \|\hat{w}\|_2 \|h\|_2 \quad \text{(Cauchy-Schwarz inequality)} \end{aligned}$$

- Other norms also provide a bound on L due to the equivalence of norms: $\exists C > 0$ s.t. $\|\hat{w}\|_2 \leqslant C\|\hat{w}\|_p$

Linear Regression vs. Ridge Regression

Objective:

- Linear: $L(w) = \frac{1}{2}\|Xw - y\|_2^2$
- Ridge: $L(w) = \frac{1}{2}\|Xw - y\|_2^2 + \frac{\lambda}{2}\|w\|_2^2$

Linear Regression vs. Ridge Regression

Objective:

- Linear: $L(w) = \frac{1}{2}\|Xw - y\|_2^2$
- Ridge: $L(w) = \frac{1}{2}\|Xw - y\|_2^2 + \frac{\lambda}{2}\|w\|_2^2$

Gradient:

- Linear: $\nabla L(w) = X^T(Xw - y)$
- Ridge: $\nabla L(w) = X^T(Xw - y) + \lambda w$
  - Also known as **weight decay** in neural networks

Linear Regression vs. Ridge Regression

Objective:

- Linear: $L(w) = \frac{1}{2}\|Xw - y\|_2^2$
- Ridge: $L(w) = \frac{1}{2}\|Xw - y\|_2^2 + \frac{\lambda}{2}\|w\|_2^2$

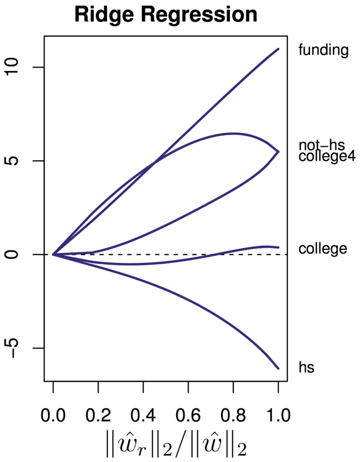
Gradient:

- Linear: $\nabla L(w) = X^T(Xw - y)$
- Ridge: $\nabla L(w) = X^T(Xw - y) + \lambda w$
  - Also known as **weight decay** in neural networks

Closed-form solution:

- Linear: $X^T Xw = X^T y$
- Ridge: $(X^T X + \lambda I)w = X^T y$
  - $(X^T X + \lambda I)$ is always invertible

Ridge Regression: Regularization Path



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

$$\hat{w} = \hat{w}_\infty = \text{Unconstrained ERM}$$

- For $r = 0$, $\|\hat{w}_r\|_2/\|\hat{w}\|_2 = 0$.
- For $r = \infty$, $\|\hat{w}_r\|_2/\|\hat{w}\|_2 = 1$

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

Lasso Regression

Penalize the ℓ_1 norm of the weights:

Lasso Regression (Tikhonov Form)

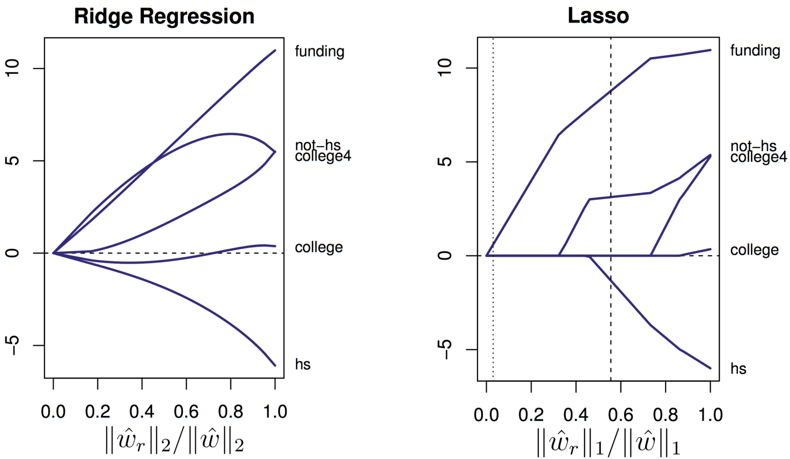
$$\hat{w} = \arg\min_{w \in \mathbf{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2 + \lambda \|w\|_1,$$

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

("Least Absolute Shrinkage and Selection Operator")

Ridge vs. Lasso: Regularization Paths

Lasso yields sparse weights:



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

The Benefits of Sparsity

The coefficient for a feature is 0 $\implies$ the feature is not needed for prediction. Why is that useful?

The Benefits of Sparsity

The coefficient for a feature is 0 $\implies$ the feature is not needed for prediction. Why is that useful?

- Faster to compute the features; cheaper to measure or annotate them

The Benefits of Sparsity

The coefficient for a feature is 0 $\implies$ the feature is not needed for prediction. Why is that useful?

- Faster to compute the features; cheaper to measure or annotate them
- Less memory to store features (deployment on a mobile device)

The Benefits of Sparsity

The coefficient for a feature is 0 $\implies$ the feature is not needed for prediction. Why is that useful?

- Faster to compute the features; cheaper to measure or annotate them
- Less memory to store features (deployment on a mobile device)
- Interpretability: identifies the important features

The Benefits of Sparsity

The coefficient for a feature is 0 $\implies$ the feature is not needed for prediction. Why is that useful?

- Faster to compute the features; cheaper to measure or annotate them
- Less memory to store features (deployment on a mobile device)
- Interpretability: identifies the important features
- Prediction function may generalize better

The Benefits of Sparsity

The coefficient for a feature is 0 $\implies$ the feature is not needed for prediction. Why is that useful?

- Faster to compute the features; cheaper to measure or annotate them
- Less memory to store features (deployment on a mobile device)
- Interpretability: identifies the important features
- Prediction function may generalize better
- Feature-selection step for training a slower non-linear model

Why does ℓ_1 Regularization Lead to Sparsity?

Regularization as Constrained Empirical Risk Minimization

Constrained ERM (Ivanov regularization)

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

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

$$\text{s.t. } \Omega(f) \leqslant r$$

Lasso Regression (Ivanov Form)

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

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

r has the same role as λ in penalized ERM (Tikhonov).

Ivanov vs. Tikhonov Regularization

- Let $L : \mathcal{F} \to \mathbf{R}$ be any performance measure of f
  - e.g. $L(f)$ could be the empirical risk of f

Ivanov vs. Tikhonov Regularization

- Let $L : \mathcal{F} \to \mathrm{R}$ be any performance measure of f
  - e.g. $L(f)$ could be the empirical risk of f
- For many L and Ω, Ivanov and Tikhonov are equivalent:
  - Any solution f^* we can get from Ivanov, we can also get from Tikhonov.
  - Any solution f^* we can get from Tikhonov, we can also get from Ivanov.

Ivanov vs. Tikhonov Regularization

- Let $L : \mathcal{F} \to \mathbf{R}$ be any performance measure of f
  - e.g. $L(f)$ could be the empirical risk of f
- For many L and Ω, Ivanov and Tikhonov are equivalent:
  - Any solution f^* we can get from Ivanov, we can also get from Tikhonov.
  - Any solution f^* we can get from Tikhonov, we can also get from Ivanov.
- The conditions for this equivalence can be derived from Lagrangian duality theory.

Ivanov vs. Tikhonov Regularization

- Let $L : \mathcal{F} \to \mathrm{R}$ be any performance measure of f
  - e.g. $L(f)$ could be the empirical risk of f
- For many L and Ω, Ivanov and Tikhonov are equivalent:
  - Any solution f^* we can get from Ivanov, we can also get from Tikhonov.
  - Any solution f^* we can get from Tikhonov, we can also get from Ivanov.
- The conditions for this equivalence can be derived from Lagrangian duality theory.
- In practice, both approaches are effective: we will use whichever one is more convenient for training or analysis.

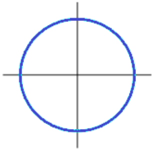
The ℓ_1 and ℓ_2 Norm Constraints

- Let's consider $\mathcal{F} = \{f(x) = w_1 x_1 + w_2 x_2\}$ space)
- We can represent each function in $\mathcal{F}$ as a point $(w_1, w_2) \in \mathrm{R}^2$.
- Where in R^2 are the functions that satisfy the Ivanov regularization constraint for ℓ_1 and ℓ_2?

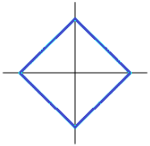
The ℓ_1 and ℓ_2 Norm Constraints

- Let's consider $\mathcal{F} = \{f(x) = w_1x_1 + w_2x_2\}$ space)
- We can represent each function in $\mathcal{F}$ as a point $(w_1, w_2) \in \mathbb{R}^2$.
- Where in $\mathbb{R}^2$ are the functions that satisfy the Ivanov regularization constraint for ℓ_1 and ℓ_2?

- ℓ_2 contour: $w_1^2 + w_2^2 = r$



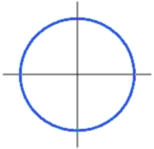
- ℓ_1 contour: $|w_1| + |w_2| = r$



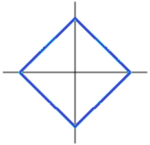
The ℓ_1 and ℓ_2 Norm Constraints

- Let's consider $\mathcal{F} = \{f(x) = w_1 x_1 + w_2 x_2\}$ space)
- We can represent each function in $\mathcal{F}$ as a point $(w_1, w_2) \in \mathbf{R}^2$.
- Where in $\mathbf{R}^2$ are the functions that satisfy the Ivanov regularization constraint for ℓ_1 and ℓ_2?

- ℓ_2 contour: $w_1^2 + w_2^2 = r$



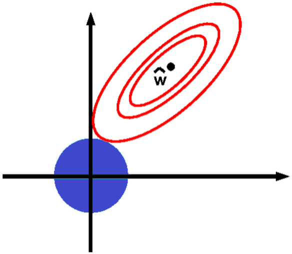
- ℓ_1 contour: $|w_1| + |w_2| = r$



- Where are the sparse solutions?

Visualizing Regularization

- $f_r^* = \arg\min_{w \in R^2} \sum_{i=1}^n \left(w^T x_i - y_i\right)^2$ subject to $w_1^2 + w_2^2 \leqslant r$

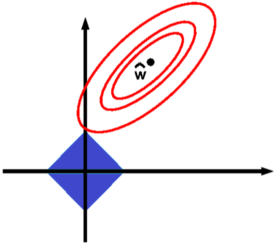


- Blue region: Area satisfying complexity constraint: $w_1^2 + w_2^2 \leqslant r$
- Red lines: contours of the empirical risk $\hat{R}_n(w) = \sum_{i=1}^n \left(w^T x_i - y_i\right)^2$.

KPM Fig. 13.3

Why Does ℓ_1 Regularization Encourage Sparse Solutions?

- $f_r^* = \arg\min_{w \in \mathbb{R}^2} \frac{1}{n} \sum_{i=1}^{n} \left(w^T x_i - y_i\right)^2$ subject to $|w_1| + |w_2| \leqslant r$



- Blue region: Area satisfying complexity constraint: $|w_1| + |w_2| \leqslant r$
- Red lines: contours of the empirical risk $\hat{R}_n(w) = \sum_{i=1}^{n} \left(w^T x_i - y_i\right)^2$.
- ℓ_1 solution tends to touch the corners.

KPM Fig. 13.3

Why Does ℓ_1 Regularization Encourage Sparse Solutions?

Geometric intuition: Projection onto diamond encourages solutions at corners.

- $\hat{w}$ in red/green regions are closest to corners in the ℓ_1 "ball".

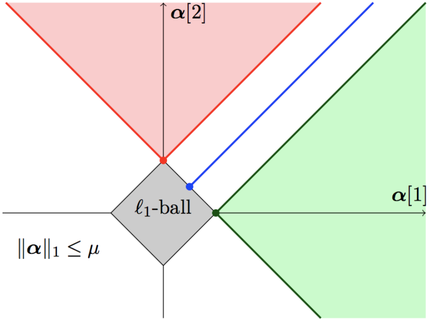


Fig from Mairal et al.'s Sparse Modeling for Image and Vision Processing Fig 1.6

Why Does ℓ_1 Regularization Encourage Sparse Solutions?

Geometric intuition: Projection onto ℓ_2 sphere favors all directions equally.

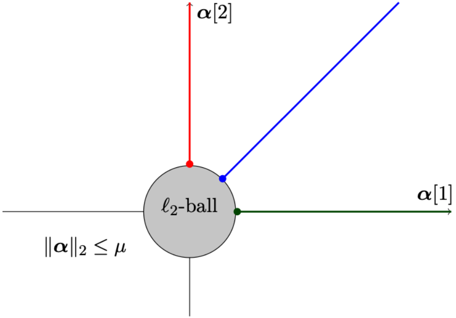


Fig from Mairal et al.'s Sparse Modeling for Image and Vision Processing Fig 1.6

Why does ℓ_2 Encourage Sparsity? Optimization Perspective

For ℓ_2 regularization,

- As w_i becomes smaller, there is less and less penalty
  - What is the ℓ_2 penalty for $w_i = 0.0001$?
- The gradient—which determines the pace of optimization—decreases as w_i approaches zero
- Less incentive to make a small weight equal to exactly zero

Why does ℓ_2 Encourage Sparsity? Optimization Perspective

For ℓ_2 regularization,

- As w_i becomes smaller, there is less and less penalty
  - What is the ℓ_2 penalty for $w_i = 0.0001$?
- The gradient—which determines the pace of optimization—decreases as w_i approaches zero
- Less incentive to make a small weight equal to exactly zero

For ℓ_1 regularization,

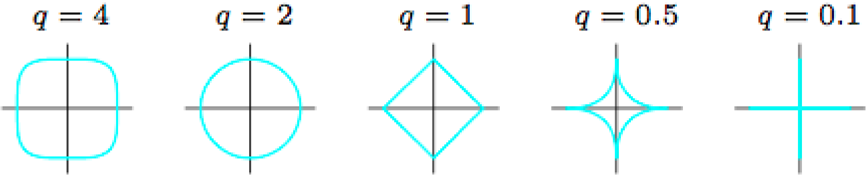
- The gradient stays the same as the weights approach zero
- This pushes the weights to be exactly zero even if they are already small

(ℓ_q) Regularization

- We can generalize to ℓ_q : $(\|w\|_q)^q = |w_1|^q + |w_2|^q$.

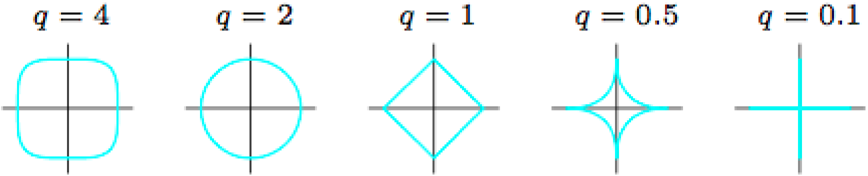
(ℓ_q) Regularization

- We can generalize to ℓ_q : $(\|w\|_q)^q = |w_1|^q + |w_2|^q$.



(ℓ_q) Regularization

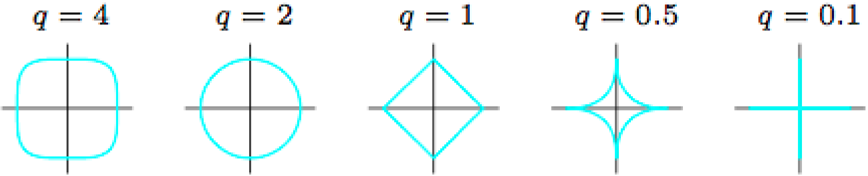
- We can generalize to ℓ_q : $(\|w\|_q)^q = |w_1|^q + |w_2|^q$.



- Note: $\|w\|_q$ is only a norm if $q \geqslant 1$, but not for $q \in (0, 1)$

(ℓ_q) Regularization

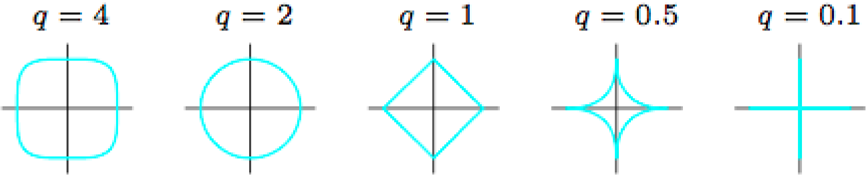
- We can generalize to ℓ_q : $(\|w\|_q)^q = |w_1|^q + |w_2|^q$.



- Note: $\|w\|_q$ is only a norm if $q \geqslant 1$, but not for $q \in (0, 1)$
- When $q < 1$, the ℓ_q constraint is non-convex, so it is hard to optimize; lasso is good enough in practice

(ℓ_q) Regularization

- We can generalize to ℓ_q : $(\|w\|_q)^q = |w_1|^q + |w_2|^q$.



- Note: $\|w\|_q$ is only a norm if $q \geqslant 1$, but not for $q \in (0, 1)$
- When $q < 1$, the ℓ_q constraint is non-convex, so it is hard to optimize; lasso is good enough in practice
- ℓ_0 $(\|w\|_0)$ is defined as the number of non-zero weights, i.e. subset selection

Minimizing the lasso objective

Minimizing the lasso objective

- The ridge regression objective is differentiable (and there is a closed form solution)
- Lasso objective function:

$$\min_{w \in \mathrm{R}^d} \sum_{i=1}^{n} \left(w^T x_i - y_i\right)^2 + \lambda \|w\|_1$$

- $\|w\|_1 = |w_1| + \ldots + |w_d|$ is not differentiable!

Minimizing the lasso objective

- The ridge regression objective is differentiable (and there is a closed form solution)
- Lasso objective function:

$$\min_{w \in \mathbf{R}^d} \sum_{i=1}^{n} \left(w^T x_i - y_i\right)^2 + \lambda \|w\|_1$$

- $\|w\|_1 = |w_1| + \ldots + |w_d|$ is not differentiable!
- We will briefly review three approaches for finding the minimum:
  - Quadratic programming
  - Projected SGD
  - Coordinate descent

Rewriting the Absolute Value

- Consider any number $a \in \mathbf{R}$.
- Let the **positive part** of a be
$$a^{+} = a\mathbb{1}(a \geqslant 0).$$
- Let the **negative part** of a be
$$a^{-} = -a\mathbb{1}(a \leqslant 0).$$

Rewriting the Absolute Value

- Consider any number $a \in R$.
- Let the **positive part** of a be

$$a^{+} = a1(a \geqslant 0).$$

- Let the **negative part** of a be

$$a^{-} = -a1(a \leqslant 0).$$

- Is it always the case that $a^{+} \geqslant 0$ and $a^{-} \geqslant 0$?

Rewriting the Absolute Value

- Consider any number $a \in R$.
- Let the **positive part** of a be
$$a^{+} = a1(a \geqslant 0).$$
- Let the **negative part** of a be
$$a^{-} = -a1(a \leqslant 0).$$
- Is it always the case that $a^{+} \geqslant 0$ and $a^{-} \geqslant 0$?
- How do you write a in terms of a^{+} and a^{-}?

Rewriting the Absolute Value

- Consider any number $a \in R$.
- Let the **positive part** of a be
$$a^{+} = a1(a \geqslant 0).$$
- Let the **negative part** of a be
$$a^{-} = -a1(a \leqslant 0).$$
- Is it always the case that $a^{+} \geqslant 0$ and $a^{-} \geqslant 0$?
- How do you write a in terms of a^{+} and a^{-}?
- How do you write $|a|$ in terms of a^{+} and a^{-}?

The Lasso as a Quadratic Program

Substituting $w = w^+ - w^-$ and $|w| = w^+ + w^-$ results in an equivalent problem:

$$\min_{w^+,w^-} \quad \sum_{i=1}^{n} \left(\left(w^+ - w^-\right)^T x_i - y_i \right)^2 + \lambda 1^T \left(w^+ + w^-\right)$$

$$\text{subject to} \quad w_i^+ \geqslant 0 \text{ for all } i \quad \text{and} \quad w_i^- \geqslant 0 \text{ for all } i,$$

- This objective is **differentiable** (in fact, **convex and quadratic**)
- How many variables does the new objective have?
- This is a **quadratic program**: a convex quadratic objective with linear constraints.
- Quadratic programming is a very well understood problem; we can plug this into a generic QP solver.

Are we missing some constraints?

We have claimed that the following objective is equivalent to the lasso problem:

$$\min_{w^+,w^-} \quad \sum_{i=1}^{n}\left(\left(w^+ - w^-\right)^T x_i - y_i\right)^2 + \lambda 1^T\left(w^+ + w^-\right)$$

$$\text{subject to} \quad w_i^+ \geqslant 0 \text{ for all } i \qquad w_i^- \geqslant 0 \text{ for all } i,$$

- When we plug this optimization problem into a QP solver,
  - it just sees $2d$ variables and $2d$ constraints.
  - Doesn't know we want w_i^+ and w_i^- to be positive and negative parts of w_i.
- Turns out that these constraints will be satisfied anyway!
- To make it clear that the solver isn't aware of the constraints of w_i^+ and w_i^-, let's denote them a_i and b_i

The Lasso as a Quadratic Program

(Trivially) reformulating the lasso problem:

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

The Lasso as a Quadratic Program

(Trivially) reformulating the lasso problem:

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

Claim: Don't need the constraint $a+b=|w|$.

Exercise: Prove by showing that the optimal solutions a^* and b^* satisfies $\min(a^*, b^*)=0$, hence $a^*+b^*=|w|$.

The Lasso as a Quadratic Program

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

Claim: Can remove $\min_w$ and the constraint $a - b = w$.

Exercise: Prove by switching the order of the minimization.

Projected SGD

- Now that we have a differentiable objective, we could also use gradient descent
- But how do we handle the constraints?

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

- Projected SGD is just like SGD, but after each step
  - We project w^+ and w^- into the constraint set.
  - In other words, if any component of w^+ or w^- becomes negative, we set it back to 0.

Coordinate Descent Method

Goal: Minimize $L(w) = L(w_1, \ldots, w_d)$ over $w = (w_1, \ldots, w_d) \in \mathbf{R}^d$.

Coordinate Descent Method

Goal: Minimize $L(w) = L(w_1, \ldots, w_d)$ over $w = (w_1, \ldots, w_d) \in \mathbf{R}^d$.

- In gradient descent or SGD, each step potentially changes all entries of w.

Coordinate Descent Method

Goal: Minimize $L(w) = L(w_1, \ldots, w_d)$ over $w = (w_1, \ldots, w_d) \in \mathrm{R}^d$.

- In gradient descent or SGD, each step potentially changes all entries of w.
- In **coordinate descent**, each step adjusts only a single coordinate w_i.

$$w_i^{\text{new}} = \arg\min_{w_i} L(w_1, \ldots, w_{i-1}, \mathrm{w_i}, w_{i+1}, \ldots, w_d)$$

- Solving this argmin may itself be an iterative process.
- Coordinate descent is an effective method when it's easy or easier to minimize w.r.t. one coordinate at a time

Coordinate Descent Method

Goal: Minimize $L(w) = L(w_1, \dots w_d)$ over $w = (w_1, \dots, w_d) \in \mathrm{R}^d$.

- **Initialize** $w^{(0)} = 0$
- **while** not converged:
  - Choose a coordinate $j \in \{1, \dots, d\}$
  - $w_j^{\text{new}} \leftarrow \arg\min_{w_j} L(w_1^{(t)}, \dots, w_{j-1}^{(t)}, \mathrm{w_j}, w_{j+1}^{(t)}, \dots, w_d^{(t)})$
  - $w_j^{(t+1)} \leftarrow w_j^{\text{new}}$ and $w^{(t+1)} \leftarrow w^{(t)}$
  - $t \leftarrow t + 1$

- Random coordinate choice $\implies$ **stochastic coordinate descent**
- Cyclic coordinate choice $\implies$ **cyclic coordinate descent**

Coordinate Descent Method for Lasso

The lasso objective coordinate minimization has a closed form! If

$$\hat{w}_j = \underset{w_j \in \mathbf{R}}{\arg\min} \sum_{i=1}^{n} \left(w^T x_i - y_i\right)^2 + \lambda |w|_1$$

Then

$$\hat{w}_j = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } c_j \in [-\lambda, \lambda] \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases}$$

$$a_j = 2\sum_{i=1}^{n} x_{i,j}^2 \qquad c_j = 2\sum_{i=1}^{n} x_{i,j}(y_i - w_{-j}^T x_{i,-j})$$

where w_{-j} is w without the j-th component, and $x_{i,-j}$ is x_i without the j-th component.

Coordinate Descent in General

- In general, coordinate descent is not competitive with gradient descent: its convergence rate is slower and the iteration cost is similar

Coordinate Descent in General

- In general, coordinate descent is not competitive with gradient descent: its convergence rate is slower and the iteration cost is similar
- But it works very well for certain problems

Coordinate Descent in General

- In general, coordinate descent is not competitive with gradient descent: its convergence rate is slower and the iteration cost is similar
- But it works very well for certain problems
- Very simple and easy to implement

Coordinate Descent in General

- In general, coordinate descent is not competitive with gradient descent: its convergence rate is slower and the iteration cost is similar
- But it works very well for certain problems
- Very simple and easy to implement
- Example applications: lasso regression, SVMs