

Controlling Complexity: Regularization

Mengye Ren

(Slides credit to David Rosenberg, He He, et al.)

NYU

September 17, 2024

Logistic Regression

- If the label is 0 or 1:
- $\hat{y} = \sigma(z)$, where σ is the sigmoid function.

$$\sigma(z) = \frac{1}{1 + \exp(-z)}.$$

Logistic Regression

- If the label is 0 or 1:
- $\hat{y} = \sigma(z)$, where σ is the sigmoid function.

$$\sigma(z) = \frac{1}{1 + \exp(-z)}.$$

- The loss is binary cross entropy:

$$\ell_{\text{Logistic}} = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}).$$

Logistic Regression

- If the label is 0 or 1:
- $\hat{y} = \sigma(z)$, where σ is the sigmoid function.

$$\sigma(z) = \frac{1}{1 + \exp(-z)}.$$

- The loss is binary cross entropy:

$$\ell_{\text{Logistic}} = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}).$$

- Remember the negative sign!

Logistic Regression

- If the label is -1 or 1:
- Note: $1 - \sigma(z) = \sigma(-z)$

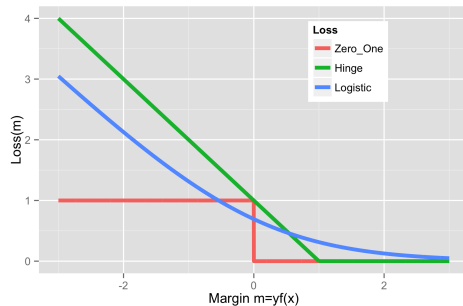
Logistic Regression

- If the label is -1 or 1:
- Note: $1 - \sigma(z) = \sigma(-z)$
- Now we can derive an equivalent loss form:

$$\begin{aligned}\ell_{\text{Logistic}} &= \begin{cases} -\log(\sigma(z)) & \text{if } y = 1 \\ -\log(\sigma(-z)) & \text{if } y = -1 \end{cases} \\ &= -\log(\sigma(yz)) \\ &= -\log\left(\frac{1}{1 + e^{-yz}}\right) \\ &= \log(1 + e^{-m}).\end{aligned}$$

Logistic Loss

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



Logistic loss is differentiable. Logistic loss always rewards a larger margin (the loss is never 0).

What About Square Loss for Classification?

- Loss $\ell(f(x), y) = (f(x) - y)^2$.

What About Square Loss for Classification?

- Loss $\ell(f(x), y) = (f(x) - y)^2$.
- Turns out, can write this in terms of margin $m = f(x)y$:
- Using fact that $y^2 = 1$, since $y \in \{-1, 1\}$.

$$\ell(f(x), y) = (f(x) - y)^2$$

What About Square Loss for Classification?

- Loss $\ell(f(x), y) = (f(x) - y)^2$.
- Turns out, can write this in terms of margin $m = f(x)y$:
- Using fact that $y^2 = 1$, since $y \in \{-1, 1\}$.

$$\begin{aligned}\ell(f(x), y) &= (f(x) - y)^2 \\ &= f^2(x) - 2f(x)y + y^2\end{aligned}$$

What About Square Loss for Classification?

- Loss $\ell(f(x), y) = (f(x) - y)^2$.
- Turns out, can write this in terms of margin $m = f(x)y$:
- Using fact that $y^2 = 1$, since $y \in \{-1, 1\}$.

$$\begin{aligned}\ell(f(x), y) &= (f(x) - y)^2 \\ &= f^2(x) - 2f(x)y + y^2 \\ &= f^2(x)y^2 - 2f(x)y + 1\end{aligned}$$

What About Square Loss for Classification?

- Loss $\ell(f(x), y) = (f(x) - y)^2$.
- Turns out, can write this in terms of margin $m = f(x)y$:
- Using fact that $y^2 = 1$, since $y \in \{-1, 1\}$.

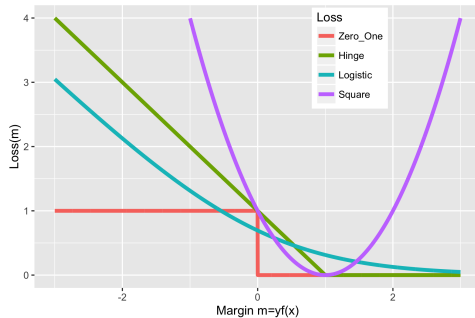
$$\begin{aligned}\ell(f(x), y) &= (f(x) - y)^2 \\ &= f^2(x) - 2f(x)y + y^2 \\ &= f^2(x)y^2 - 2f(x)y + 1 \\ &= (1 - f(x)y)^2\end{aligned}$$

What About Square Loss for Classification?

- Loss $\ell(f(x), y) = (f(x) - y)^2$.
- Turns out, can write this in terms of margin $m = f(x)y$:
- Using fact that $y^2 = 1$, since $y \in \{-1, 1\}$.

$$\begin{aligned}\ell(f(x), y) &= (f(x) - y)^2 \\ &= f^2(x) - 2f(x)y + y^2 \\ &= f^2(x)y^2 - 2f(x)y + 1 \\ &= (1 - f(x)y)^2 \\ &= (1 - m)^2\end{aligned}$$

What About Square Loss for Classification?



Heavily penalizes outliers (e.g. mislabeled examples).

Controlling the Complexity through Regularization

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 (need more samples)

Complexity of Hypothesis Spaces

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

- Bigger \mathcal{F} : better approximation but can overfit (need more samples)
- Smaller \mathcal{F} : less likely to overfit but can be farther from the true function

Complexity of Hypothesis Spaces

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

- Bigger \mathcal{F} : better approximation but can overfit (need more samples)
- Smaller \mathcal{F} : less likely to overfit but can be farther from the true function

To control the “size” of \mathcal{F} , we need some measure of its **complexity**:

- Number of variables / features

Complexity of Hypothesis Spaces

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

- Bigger \mathcal{F} : better approximation but can overfit (need more samples)
- Smaller \mathcal{F} : less likely to overfit but can be farther from the true 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

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

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

- $\mathcal{F} = \{\text{all polynomial functions}\}$
 - $\mathcal{F}_d = \{\text{all polynomials of degree } \leq d\}$
2. Select one of these models based on a score (e.g. validation error)

Feature Selection in Linear Regression

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

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

Feature Selection in Linear Regression

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

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

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

Feature Selection in Linear Regression

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

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

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
- **Not an efficient search algorithm**; iterating over all subsets becomes very expensive with a large number of features

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
- Forward & backward selection do not guarantee 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
- Forward & backward selection do not guarantee to find the best solution.
- Forward & backward selection do not in general result in the same subset.

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
- Forward & backward selection do not guarantee to find the best solution.
- Forward & backward selection do not in general result in the same subset.
- Could there be a more consistent way of formulating feature selection as an optimization problem?

ℓ_2 and ℓ_1 Regularization

Complexity Penalty

An objective that balances number of features and prediction performance:

$$\text{score}(S) = \text{training_loss}(S) + \lambda|S| \quad (1)$$

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

Complexity Penalty

An objective that balances number of features and prediction performance:

$$\text{score}(S) = \text{training_loss}(S) + \lambda|S| \quad (1)$$

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 $\lambda \rightarrow$ complex models are penalized more heavily

Complexity Penalty

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

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

Complexity Penalty

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

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

Penalized ERM (Tikhonov regularization)

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

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

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

Penalized ERM (Tikhonov regularization)

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

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

As usual, we find λ using the validation data.

Number of features as complexity measure is not differentiable and hard to optimize—other measures?

- We can imagine having a weight for each feature dimension.

- We can imagine having a weight for each feature dimension.
- In linear regression, the model weights multiply each feature dimension:

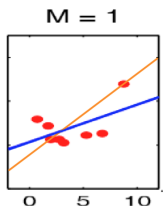
$$f(x) = w^T x$$

- We can imagine having a weight for each feature dimension.
- In linear regression, the model weights multiply each feature dimension:

$$f(x) = w^T x$$

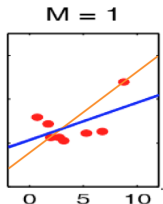
- If w_i is zero or close to zero, then it means that we are not using the i -th feature.

Weight Shrinkage: Intuition



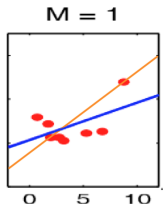
- Why would we prefer a regression line with **smaller slope** (unless the data strongly supports a larger slope)?

Weight Shrinkage: Intuition



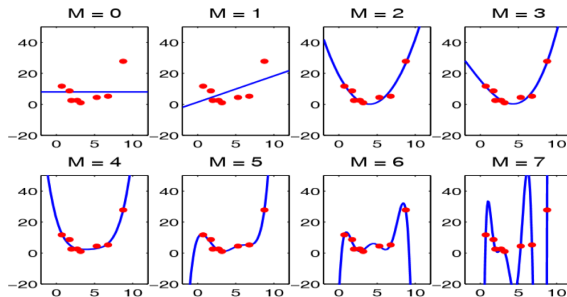
- Why would we prefer a regression line with **smaller slope** (unless the data strongly supports a larger slope)?
- More stable: small change in the input does not cause large change in the output

Weight Shrinkage: Intuition



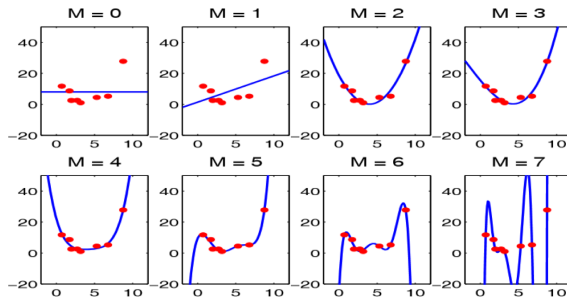
- Why would we prefer a regression line with **smaller slope** (unless the data strongly supports a larger slope)?
- More stable: small change in the input does not cause large change in the output
- If we push the estimated weights to be small, re-estimating them on a new dataset wouldn't cause the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

Weight Shrinkage: Polynomial Regression



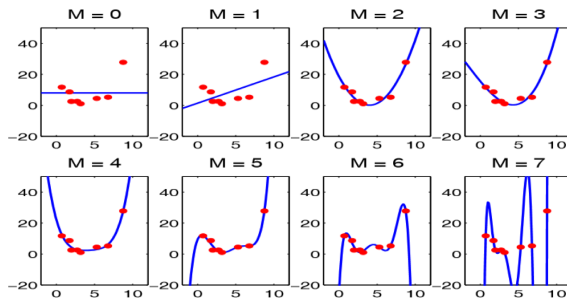
- n -th feature dimension is the n -th power of x : $1, x, x^2, \dots$

Weight Shrinkage: Polynomial Regression



- n -th feature dimension is the n -th power of x : $1, x, x^2, \dots$
- Large weights are needed to make the curve wiggle sufficiently to overfit the data

Weight Shrinkage: Polynomial Regression



- n-th feature dimension is the n-th power of x : $1, x, x^2, \dots$
- 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 $\hat{y} = 1000x^7 + 500x^3 + 1$

(Adapted from Mark Schmidt's slide)

Linear Regression with ℓ_2 Regularization

- We have a linear model

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

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

Linear Regression with ℓ_2 Regularization

- We have a linear model

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

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

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

- 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:

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

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

- Also known as **ridge regression**.

Linear Regression with L2 Regularization

Penalizes large weights:

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

where $\|w\|_2^2 = w_1^2 + \dots + w_d^2$ 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:

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

where $\|w\|_2^2 = w_1^2 + \dots + w_d^2$ 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:

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

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:

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

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

Linear Regression vs. Ridge Regression

Objective:

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

Linear Regression vs. Ridge Regression

Objective:

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

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: $\nabla L(w) = X^T(Xw - y)$

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: $X^T X w = X^T y \rightarrow w = (X^T X)^{-1} X^T y$

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: $X^T X w = X^T y \rightarrow w = (X^T X)^{-1} X^T y$
- Ridge: $(X^T X + \lambda I) w = X^T y \rightarrow w = (X^T X + \lambda I)^{-1} X^T y$
 - $(X^T X + \lambda I)$ is always invertible

Constrained Optimization

- L2 regularizer is a term in our optimization objective.

$$w^* = \arg \min_w \frac{1}{2} \|Xw - y\|_2^2 + \frac{\lambda}{2} \|w\|_2^2$$

- This is also called the **Tikhonov** form.

Constrained Optimization

- L2 regularizer is a term in our optimization objective.

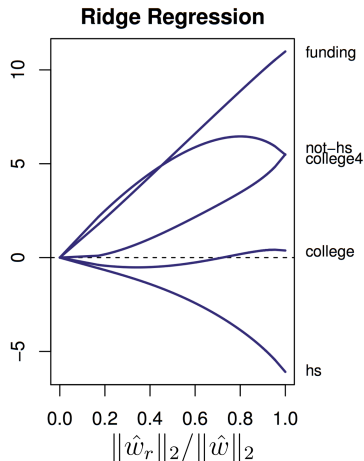
$$w^* = \arg \min_w \frac{1}{2} \|Xw - y\|_2^2 + \frac{\lambda}{2} \|w\|_2^2$$

- This is also called the **Tikhonov** form.
- The Lagrangian theory allows us to interpret the second term as a constraint.

$$w^* = \arg \min_{w: \|w\|_2^2 \leq r} \frac{1}{2} \|Xw - y\|_2^2$$

- At optimum, the gradients of the main objective and the constraint cancel out.
- This is also called the **Ivanov** form.

Ridge Regression: Regularization Path



$$\hat{w}_r = \arg \min_{\|w\|_2^2 \leq r^2} \frac{1}{n} \sum_{i=1}^n (w^T x_i - y_i)^2$$
$$\hat{w} = \hat{w}_\infty = \text{Unconstrained ERM}$$

- 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

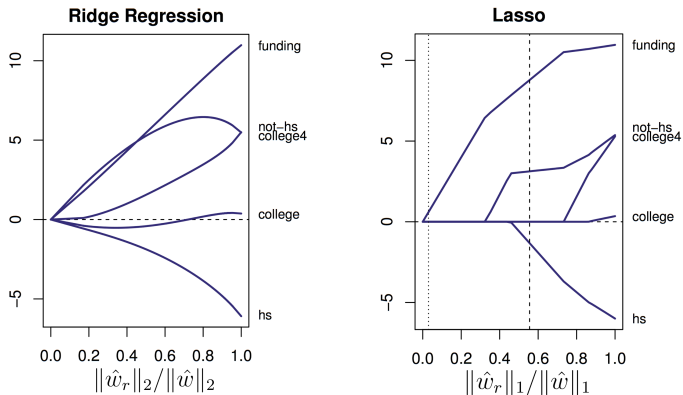
Penalize the ℓ_1 norm of the weights:

Lasso Regression (Tikhonov Form, soft penalty)

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

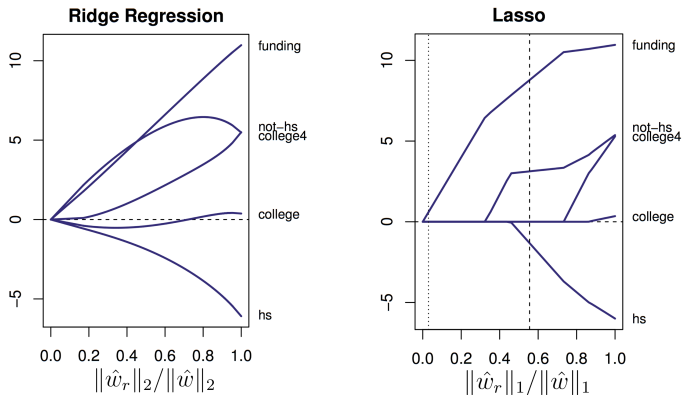
where $\|w\|_1 = |w_1| + \dots + |w_d|$ is the ℓ_1 -norm.

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.

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 (model is less complex)

Why does ℓ_1 Regularization Lead to Sparsity?

Lasso Regression

Penalize the ℓ_1 norm of the weights:

Lasso Regression (Tikhonov Form, soft penalty)

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

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

Regularization as Constrained ERM

Constrained ERM (Ivanov regularization)

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

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

Lasso Regression (Ivanov Form, hard constraint)

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

$$\hat{w} = \arg \min_{\|w\|_1 \leq r} \frac{1}{n} \sum_{i=1}^n \{w^T x_i - y_i\}^2.$$

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

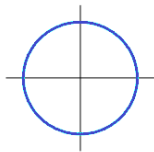
The ℓ_1 and ℓ_2 Norm Constraints

- 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 ?

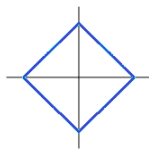
The ℓ_1 and ℓ_2 Norm Constraints

- 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:
 $w_1^2 + w_2^2 = r$



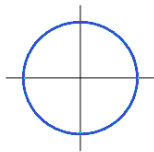
- ℓ_1 contour:
 $|w_1| + |w_2| = r$



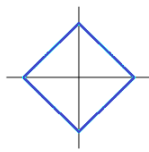
The ℓ_1 and ℓ_2 Norm Constraints

- Let's consider $\mathcal{F} = \{f(x) = w_1x_1 + w_2x_2\}$ 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:
 $w_1^2 + w_2^2 = r$



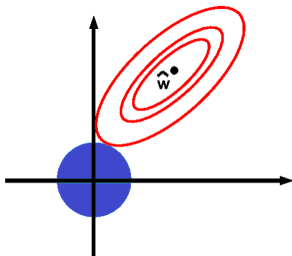
- ℓ_1 contour:
 $|w_1| + |w_2| = r$



- Where are the sparse solutions?

Visualizing Regularization

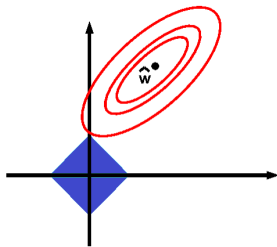
- $f_r^* = \arg \min_{w \in \mathbb{R}^2} \sum_{i=1}^n (w^T x_i - y_i)^2$ subject to $w_1^2 + w_2^2 \leq r$



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

Why Does ℓ_1 Regularization Encourage Sparse Solutions?

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



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

Why Does ℓ_1 Regularization Encourage Sparse Solutions?

Suppose the loss contour is growing like a perfect circle/sphere.

Geometric intuition: Projection onto diamond encourages solutions at corners.

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

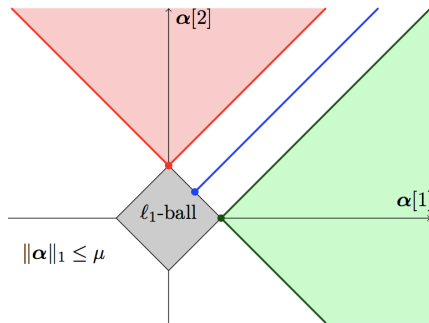


Fig from Mairal et al.'s Sparse Modeling for Image and Vision Processing Fig 1.6

Why Does ℓ_1 Regularization Encourage Sparse Solutions?

Suppose the loss contour is growing like a perfect circle/sphere.

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

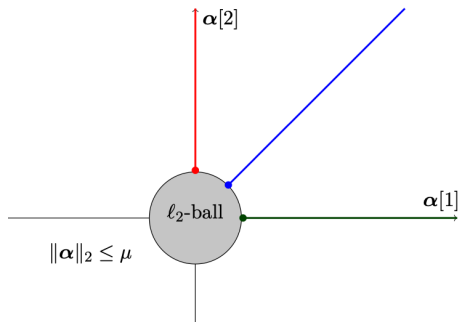


Fig from Mairal et al.'s Sparse Modeling for Image and Vision Processing Fig 1.6

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

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,

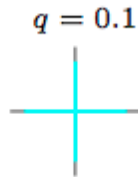
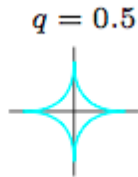
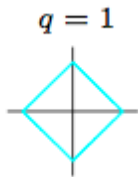
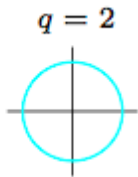
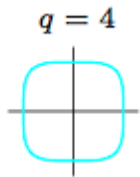
- The gradient stays the same as the weights approach zero
- This pushes the weights to be exactly zero even if they are already small

(ℓ_q) Regularization

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

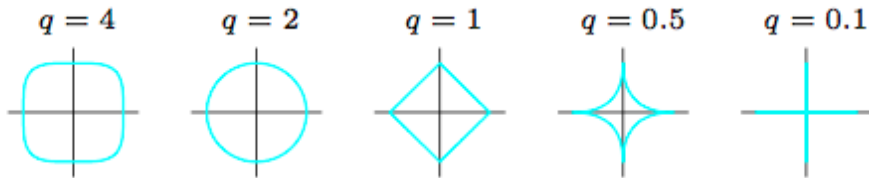
(ℓ_q) Regularization

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



(ℓ_q) Regularization

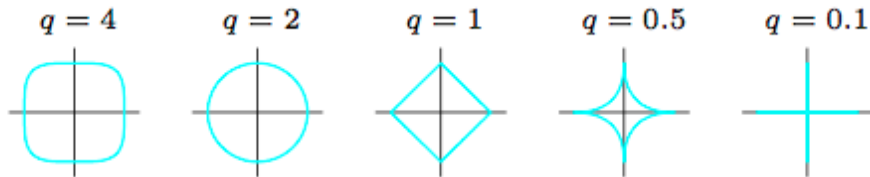
- We can generalize to ℓ_q : $(\|w\|_q)^q = |w_1|^q + |w_2|^q$.



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

(ℓ_q) Regularization

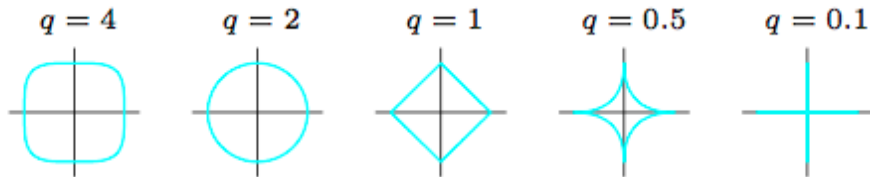
- We can generalize to ℓ_q : $(\|w\|_q)^q = |w_1|^q + |w_2|^q$.



- Note: $\|w\|_q$ is only a norm if $q \geq 1$, 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 : $(\|w\|_q)^q = |w_1|^q + |w_2|^q$.

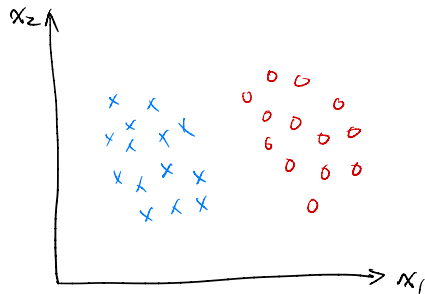


- Note: $\|w\|_q$ is only a norm if $q \geq 1$, but not for $q \in (0, 1)$
- When $q < 1$, the ℓ_q constraint is non-convex, so it is hard to optimize; lasso is good enough in practice
- ℓ_0 ($\|w\|_0$) is defined as the number of non-zero weights, i.e. subset selection

Maximum Margin Classifier

Linearly Separable Data

Consider a linearly separable dataset \mathcal{D} :

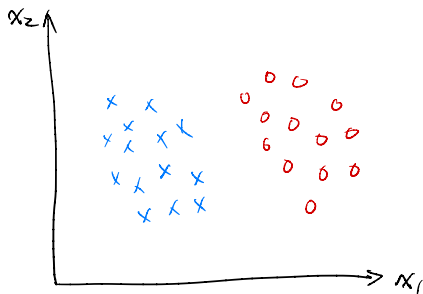


Find a separating hyperplane such that

- $w^T x_i > 0$ for all x_i where $y_i = +1$
- $w^T x_i < 0$ for all x_i where $y_i = -1$

Linearly Separable Data

Consider a linearly separable dataset \mathcal{D} :



Now let's design a learning algorithm: If there is a misclassified example, change the hyperplane according to the example.

The Perceptron Algorithm

- Initialize $w \leftarrow 0$
- While not converged (exists misclassified examples)
 - For $(x_i, y_i) \in \mathcal{D}$
 - If $y_i w^T x_i < 0$ (wrong prediction)
 - Update $w \leftarrow w + y_i x_i$

The Perceptron Algorithm

- Initialize $w \leftarrow 0$
- While not converged (exists misclassified examples)
 - For $(x_i, y_i) \in \mathcal{D}$
 - If $y_i w^T x_i < 0$ (wrong prediction)
 - Update $w \leftarrow w + y_i x_i$
- Intuition: move towards misclassified positive examples and away from negative examples

The Perceptron Algorithm

- Initialize $w \leftarrow 0$
- While not converged (exists misclassified examples)
 - For $(x_i, y_i) \in \mathcal{D}$
 - If $y_i w^T x_i < 0$ (wrong prediction)
 - Update $w \leftarrow w + y_i x_i$
- Intuition: move towards misclassified positive examples and away from negative examples
- Guarantees to find a zero-error classifier (if one exists) in finite steps

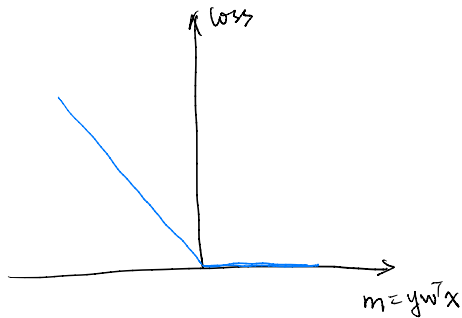
The Perceptron Algorithm

- Initialize $w \leftarrow 0$
- While not converged (exists misclassified examples)
 - For $(x_i, y_i) \in \mathcal{D}$
 - If $y_i w^T x_i < 0$ (wrong prediction)
 - Update $w \leftarrow w + y_i x_i$
- Intuition: move towards misclassified positive examples and away from negative examples
- Guarantees to find a zero-error classifier (if one exists) in finite steps
- What is the loss function if we consider this as a SGD algorithm?

Minimize the Hinge Loss

Perceptron Loss

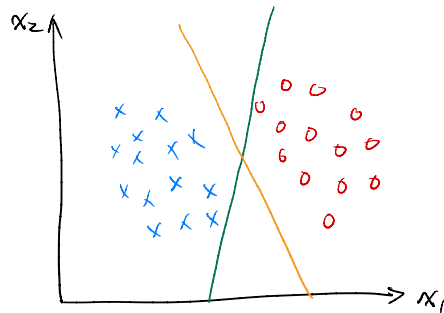
$$\ell(x, y, w) = \max(0, -yw^T x)$$



Maximum-Margin Separating Hyperplane

For separable data, there are infinitely many zero-error classifiers.

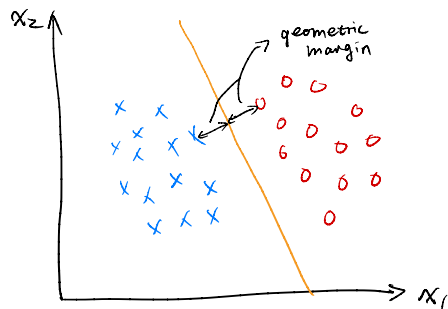
Which one do we pick?



(Perceptron does not return a unique solution.)

Maximum-Margin Separating Hyperplane

We prefer the classifier that is farthest from both classes of points



- Geometric margin: smallest distance between the hyperplane and the points
- Maximum margin: *largest* distance to the closest points

Geometric Margin

We want to maximize the distance between the **separating hyperplane** and the **closest** points.

Let's formalize the problem.

Definition (separating hyperplane)

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

Geometric Margin

We want to maximize the distance between the **separating hyperplane** and the **closest** points.

Let's formalize the problem.

Definition (separating hyperplane)

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

Definition (geometric margin)

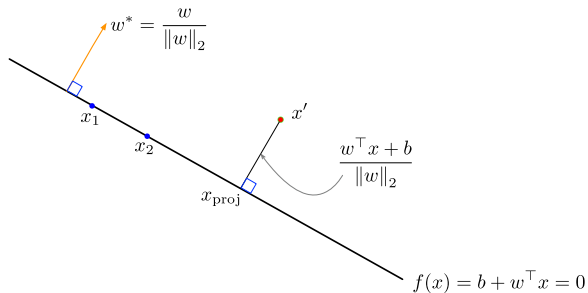
Let H be a hyperplane that separates the data (x_i, y_i) for $i = 1, \dots, n$. The **geometric margin** of this hyperplane is

$$\min_i d(x_i, H),$$

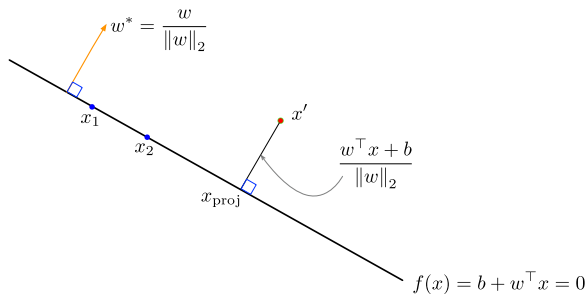
the distance from the hyperplane to the closest data point.

Distance between a Point and a Hyperplane

- Any point on the plane p , and normal vector $w/\|w\|_2$

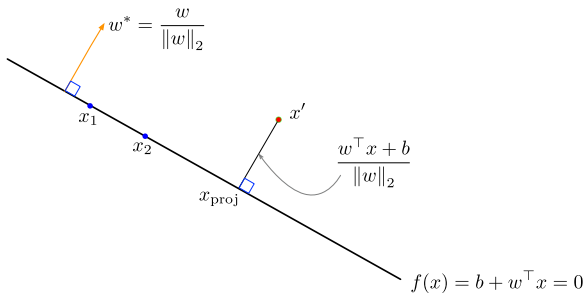


Distance between a Point and a Hyperplane



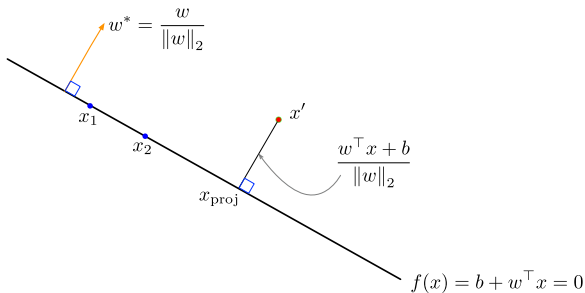
- Any point on the plane p , and normal vector $w/\|w\|_2$
- Projection of x onto the normal: $\frac{(x'-p)^T w}{\|w\|_2}$

Distance between a Point and a Hyperplane



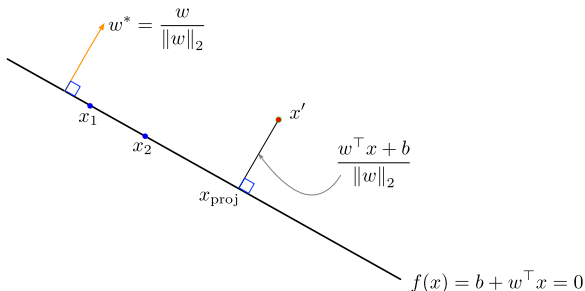
- Any point on the plane p , and normal vector $w/\|w\|_2$
- Projection of x onto the normal: $\frac{(x'-p)^T w}{\|w\|_2}$
- $(x' - p)^T w = x'^T w - p^T w = x'^T w + b$ (since $p^T w + b = 0$)

Distance between a Point and a Hyperplane



- Any point on the plane p , and normal vector $w/\|w\|_2$
- Projection of x onto the normal: $\frac{(x'-p)^T w}{\|w\|_2}$
- $(x' - p)^T w = x'^T w - p^T w = x'^T w + b$ (since $p^T w + b = 0$)
- Signed distance between x' and Hyperplane H : $\frac{w^T x' + b}{\|w\|_2}$

Distance between a Point and a Hyperplane



- Any point on the plane p , and normal vector $w/\|w\|_2$
- Projection of x onto the normal: $\frac{(x'-p)^T w}{\|w\|_2}$
- $(x' - p)^T w = x'^T w - p^T w = x'^T w + b$ (since $p^T w + b = 0$)
- Signed distance between x' and Hyperplane H : $\frac{w^T x' + b}{\|w\|_2}$
- Taking into account of the label y :
$$d(x', H) = \frac{y(w^T x' + b)}{\|w\|_2}$$

Maximize the Margin

We want to maximize the geometric margin:

$$\text{maximize } \min_i d(x_i, H).$$

Maximize the Margin

We want to maximize the geometric margin:

$$\text{maximize } \min_i d(x_i, H).$$

Given separating hyperplane $H = \{v \mid w^T v + b = 0\}$, we have

$$\text{maximize } \min_i \frac{y_i(w^T x_i + b)}{\|w\|_2}.$$

Maximize the Margin

We want to maximize the geometric margin:

$$\text{maximize } \min_i d(x_i, H).$$

Given separating hyperplane $H = \{v \mid w^T v + b = 0\}$, we have

$$\text{maximize } \min_i \frac{y_i(w^T x_i + b)}{\|w\|_2}.$$

Let's remove the inner minimization problem by

$$\begin{array}{ll} \text{maximize} & M \\ \text{subject to} & \frac{y_i(w^T x_i + b)}{\|w\|_2} \geq M \quad \text{for all } i \end{array}$$

Maximize the Margin

We want to maximize the geometric margin:

$$\text{maximize } \min_i d(x_i, H).$$

Given separating hyperplane $H = \{v \mid w^T v + b = 0\}$, we have

$$\text{maximize } \min_i \frac{y_i(w^T x_i + b)}{\|w\|_2}.$$

Let's remove the inner minimization problem by

$$\begin{array}{ll} \text{maximize} & M \\ \text{subject to} & \frac{y_i(w^T x_i + b)}{\|w\|_2} \geq M \quad \text{for all } i \end{array}$$

Note that the solution is not unique (why?).

Maximize the Margin

Let's fix the norm $\|w\|_2$ to $1/M$ to obtain:

$$\begin{array}{ll}\text{maximize} & \frac{1}{\|w\|_2} \\ \text{subject to} & y_i(w^T x_i + b) \geq 1 \quad \text{for all } i\end{array}$$

Maximize the Margin

Let's fix the norm $\|w\|_2$ to $1/M$ to obtain:

$$\begin{array}{ll}\text{maximize} & \frac{1}{\|w\|_2} \\ \text{subject to} & y_i(w^T x_i + b) \geq 1 \quad \text{for all } i\end{array}$$

It's equivalent to solving the minimization problem

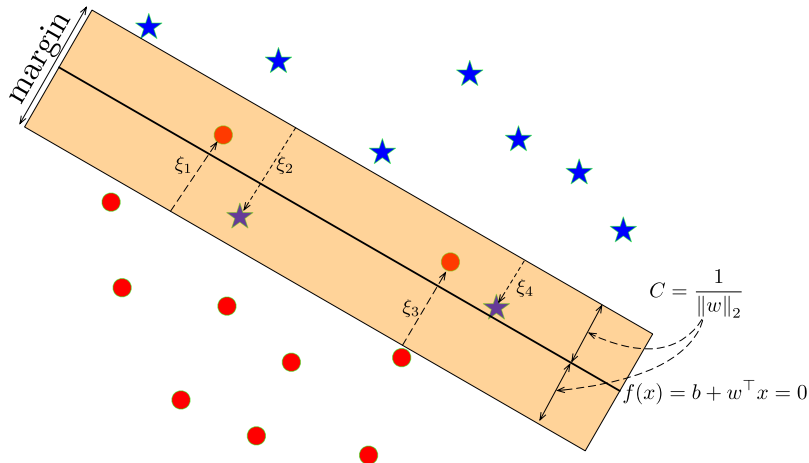
$$\begin{array}{ll}\text{minimize} & \frac{1}{2} \|w\|_2^2 \\ \text{subject to} & y_i(w^T x_i + b) \geq 1 \quad \text{for all } i\end{array}$$

Note that $y_i(w^T x_i + b)$ is the (functional) margin. The optimization finds the minimum norm solution which has a margin of at least 1 on all examples.

Not linearly separable

What if the data is *not* linearly separable?

For any w , there will be points with a negative margin.



Soft Margin SVM

Introduce **slack variables** ξ 's to penalize small margin:

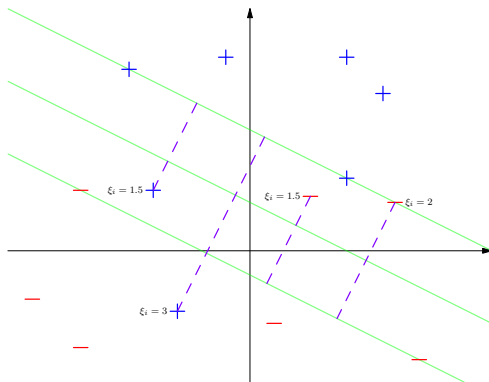
$$\begin{aligned} &\text{minimize} && \frac{1}{2} \|w\|_2^2 + \frac{C}{n} \sum_{i=1}^n \xi_i \\ &\text{subject to} && y_i(w^T x_i + b) \geq 1 - \xi_i \quad \text{for all } i \\ &&& \xi_i \geq 0 \quad \text{for all } i \end{aligned}$$

- If $\xi_i = 0 \forall i$, it's reduced to hard SVM.
- What does $\xi_i > 0$ mean?
- What does C control?

Slack Variables

$d(x_i, H) = \frac{y_i(w^T x_i + b)}{\|w\|_2} \geq \frac{1 - \xi_i}{\|w\|_2}$, thus ξ_i measures the violation by multiples of the geometric margin:

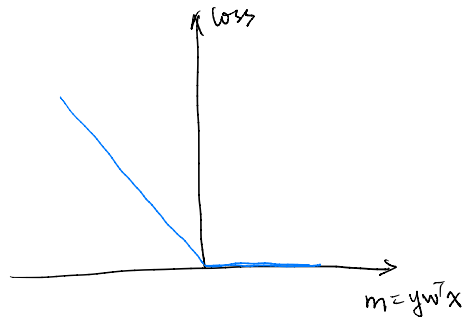
- $\xi_i = 1$: x_i lies on the hyperplane
- $\xi_i = 3$: x_i is past 2 margin width beyond the decision hyperplane



Minimize the Hinge Loss

Perceptron Loss

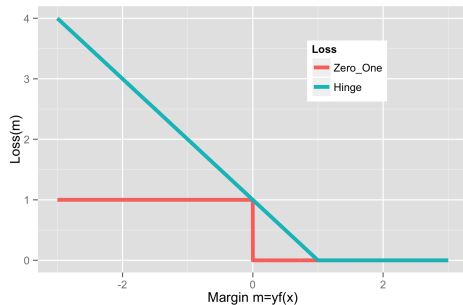
$$\ell(x, y, w) = \max(0, -yw^T x)$$



If we do ERM with this loss function, what happens?

Hinge Loss

- SVM/Hinge loss: $\ell_{\text{Hinge}} = \max\{1 - m, 0\} = (1 - m)_+$
- Margin $m = yf(x)$; “Positive part” $(x)_+ = x\mathbb{1}[x \geq 0]$.



Hinge is a **convex, upper bound** on 0–1 loss. Not differentiable at $m = 1$.
We have a “margin error” when $m < 1$.

SVM as an Optimization Problem

- The SVM optimization problem is equivalent to

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}\|w\|^2 + \frac{c}{n} \sum_{i=1}^n \xi_i \\ \text{subject to} & \xi_i \geq (1 - y_i [w^T x_i + b]) \text{ for } i = 1, \dots, n \\ & \xi_i \geq 0 \text{ for } i = 1, \dots, n\end{array}$$

SVM as an Optimization Problem

- The SVM optimization problem is equivalent to

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}\|w\|^2 + \frac{c}{n} \sum_{i=1}^n \xi_i \\ \text{subject to} & \xi_i \geq (1 - y_i [w^T x_i + b]) \text{ for } i = 1, \dots, n \\ & \xi_i \geq 0 \text{ for } i = 1, \dots, n\end{array}$$

which is equivalent to

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}\|w\|^2 + \frac{c}{n} \sum_{i=1}^n \xi_i \\ \text{subject to} & \xi_i \geq \max(0, 1 - y_i [w^T x_i + b]) \text{ for } i = 1, \dots, n.\end{array}$$

SVM as an Optimization Problem

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}\|w\|^2 + \frac{c}{n} \sum_{i=1}^n \xi_i \\ \text{subject to} & \xi_i \geq \max(0, 1 - y_i [w^T x_i + b]) \text{ for } i = 1, \dots, n.\end{array}$$

SVM as an Optimization Problem

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}\|w\|^2 + \frac{c}{n} \sum_{i=1}^n \xi_i \\ \text{subject to} & \xi_i \geq \max(0, 1 - y_i [w^T x_i + b]) \text{ for } i = 1, \dots, n.\end{array}$$

Move the constraint into the objective:

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

SVM as an Optimization Problem

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

Move the constraint into the objective:

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

- The first term is the L2 regularizer.
- The second term is the Hinge loss.

Support Vector Machine

Using ERM:

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

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

Two ways to derive the SVM optimization problem:

- Maximize the margin
- Minimize the hinge loss with ℓ_2 regularization

Both leads to the minimum norm solution satisfying certain margin constraints.

- **Hard-margin SVM:** all points must be correctly classified with the margin constraints
- **Soft-margin SVM:** allow for margin constraint violation with some penalty