Regularization

CMPUT 296: Basics of Machine Learning

Textbook §9.1

Logistics

- **Assignment #2** is due **TODAY** (at 11:59pm Mountain time)
- **Midterm exam** is **next Thursday (Oct 29)**
  - Review class on Tuesday

Recap: Comparing Models' Generalization Error

- Our goal is to minimize **generalization error**: expected cost with respect to the **underlying distribution**
  - We will often want to **compare** the generalization errors of two models
  - The **test set** gives us m **samples** of generalization error
- If the $(1-\delta)$ **confidence intervals** for the two models do not overlap, then we say that one model has **statistically significantly** better generalization error than the other, with **confidence level** δ
- More powerful: paired **hypothesis test**, e.g.:
  - Binomial counting test
  - Paired t-test
- p**-value:** Probability of seeing our dataset given that null hypothesis is true
  - **Null hypothesis:** Both models have **equal errors**

Recap:
Nonlinear Regression as Linear Regression

- **Linear regression** is useful for more than just linear models
- Can obtain nonlinear functions by **transforming** the observation vector with **arbitrary** functions of $\mathbf{x}$
- We write this as $\phi(\mathbf{x}) = (\phi_0(\mathbf{x}), \ldots, \phi_p(\mathbf{x}))$
  - E.g., for 1D polynomial regression: $\phi(x) = (1, x, x^2, \ldots, x^p)$
- We can then perform linear regression on $\phi(\mathbf{x})$ instead of $\mathbf{x}$:

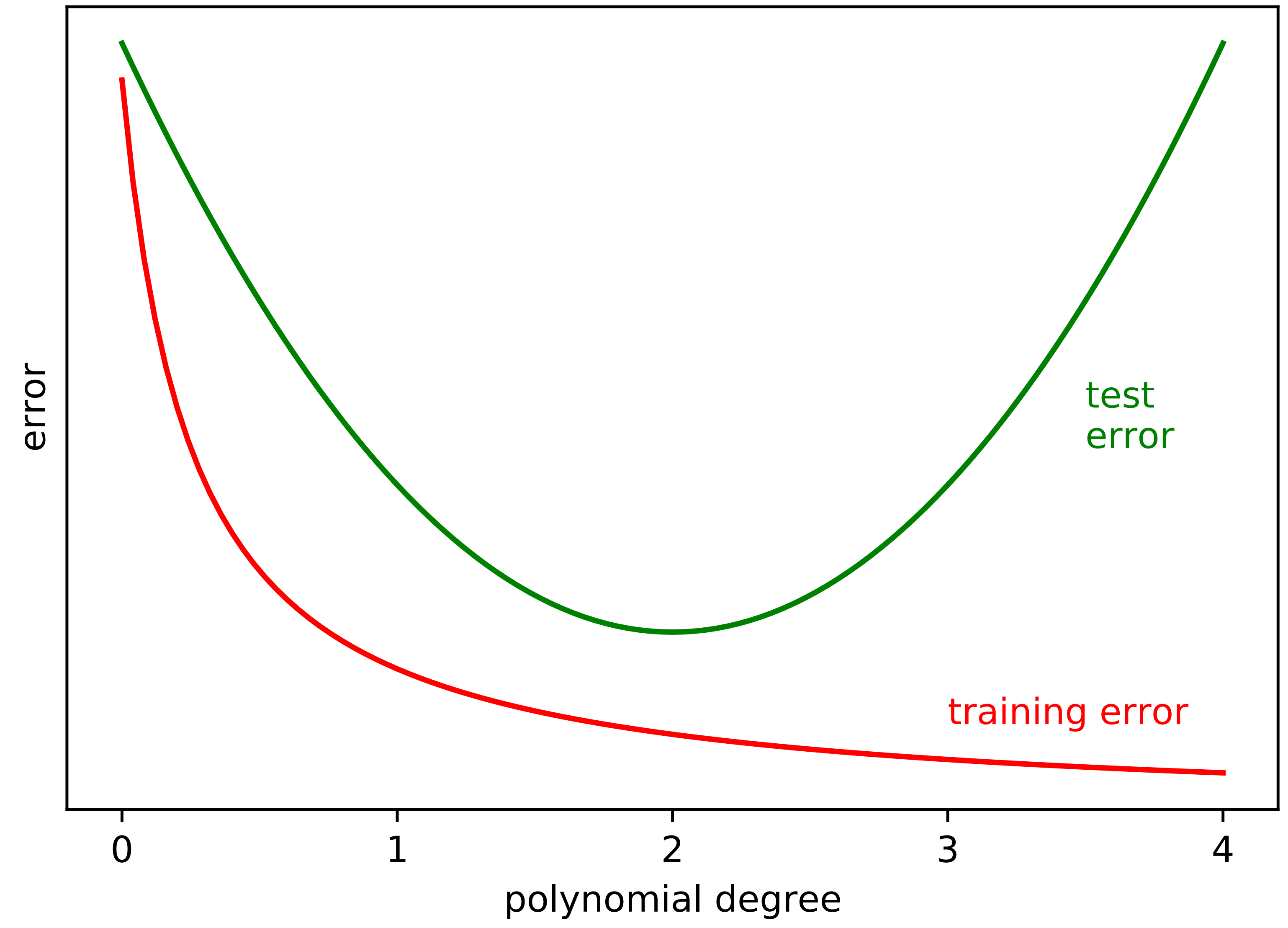
$$c(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2$$

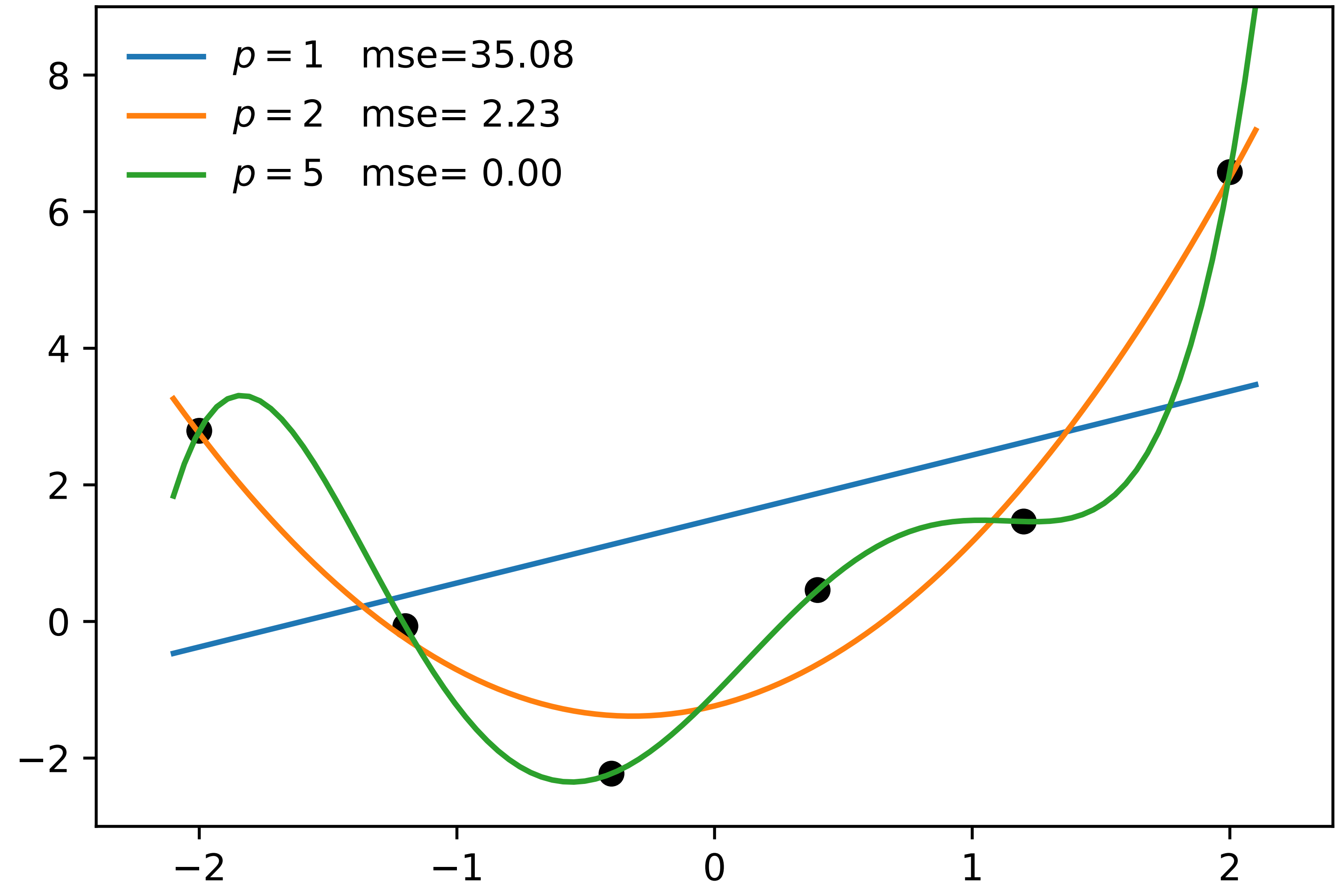
Outline

1. Recap & Logistics
2. Overfitting in Polynomial Regression
3. Regularization

Overfitting in Polynomial Regression

- **Overfitting:** Too-complicated model has **low training error** at the expense of **high generalization error**
  - e.g., polynomial regression with too-large polynomial degree
- **Question:** How can we avoid overfitting in polynomial regression?





Avoiding Overfitting via Cross-Validation

One possible approach for avoiding overfitting in polynomial regression:

1. Perform k**-fold cross-validation** for p-degree polynomial regression for all $1 \leq p \leq P$
2. Let p^* be the p that minimizes the estimated generalization error
3. Fit a p^*-degree polynomial on the full training dataset

Question: What are the possible problems with this approach?

Hyperparameter Selection as Feature Selection

- Choosing p in polynomial regression is equivalent to choosing which possible **features to include**
- Polynomial features have a natural grouping
  - Usually doesn't make sense to include x^5 if you aren't also including x^4, x^3, x^2, x
- **Question:** What if we have **arbitrary features**? How can we choose which subset to include?
- What is the complexity of trying **every subset**?

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ \cancel{x^2} \\ \cancel{x^3} \\ \cancel{x^4} \\ \cancel{x^5} \\ \cancel{x^6} \\ \cancel{x^7} \end{bmatrix} \qquad \phi(x) = \begin{bmatrix} \phi_0(x) \\ \cancel{\phi_1(x)} \\ \phi_2(x) \\ \cancel{\phi_3(x)} \\ \cancel{\phi_4(x)} \\ \cancel{\phi_5(x)} \\ \phi_6(x) \\ \cancel{\phi_7(x)} \end{bmatrix}$$

$p = 5$

Detecting Overfitting Revisited: Weights

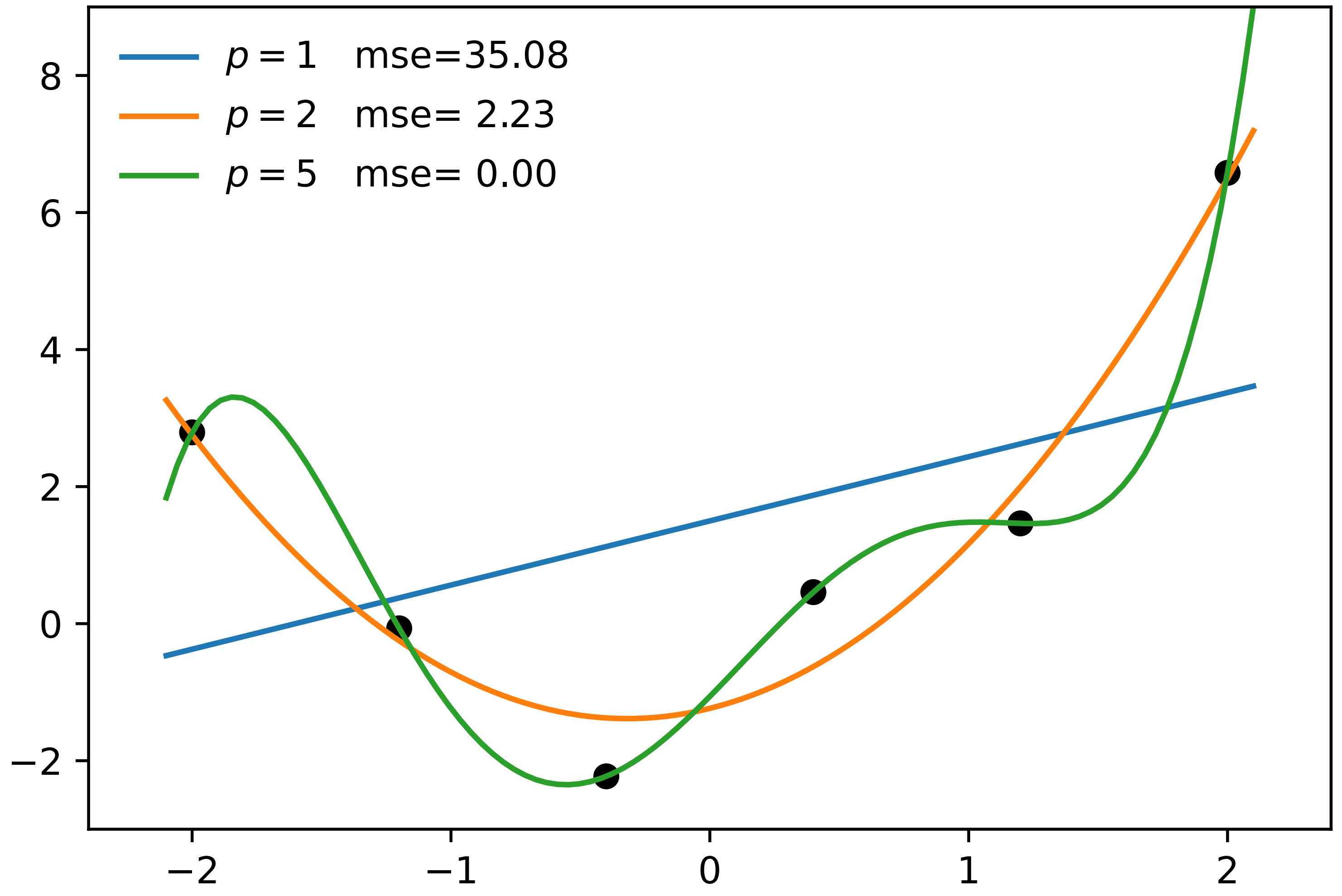
- Do you notice anything about the fitted weights for our polynomial example?

$$p = 1 : \mathbf{w} = [1.50, \quad 0.94]$$

$$p = 2 : \mathbf{w} = [-1.24, \quad 0.94, \quad 1.47]$$

$$p = 5 : \mathbf{w} = [-1.06, \quad \boxed{3.84,} \quad 1.10, \boxed{-3.06,} \quad 0.084, \quad 0.59]$$

- The overfitted 5th-degree polynomial has some large-magnitude weights (**why?**)



Regularization

Regularization:
Instead of minimizing average training cost, minimize a combination of **average training cost** and **model complexity**:

$$c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{cost}(f(\mathbf{x}_i; \mathbf{w}), y_i) + \lambda \mathrm{penalty}(\mathbf{w})$$

- The training cost depends on the dataset, but the penalty must depend only on the **parameters**
  - The λ hyperparameter controls relative importance of training set cost vs. model complexity
- For linear regression, common regularizations:

L2 regularizer ("ridge"): $\mathrm{penalty}(\mathbf{w}) = \sum_{j=1}^{d} w_j^2$

L1 regularizer ("lasso"): $\mathrm{penalty}(\mathbf{w}) = \sum_{j=1}^{d} |w_j|$

L2-Regularized Linear Regression

$$c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^T\mathbf{x}_i - y_i)^2 + \lambda \sum_{j=1}^{d} w_j^2$$

- The more features get used, the more complicated the model
- The more features get used, the higher the sum of (squared) weights
  - Features with smaller-magnitude coefficients have less effect on output
- So, only allow more complex models if the improvement in training cost is "worth it"
- **Question:** What are the advantages of this approach over cross-validation?

L2-Regularization as MAP

- Up until now, we have considered the **maximum likelihood** solution to the linear regression problem
- **L2-Regularized** linear regression can be understood as the **MAP** solution to linear regression, with an independent **Gaussian prior** on the weights
  - Each element assumed to have independent prior $\mathcal{N}(0, \boxed{\sigma^2/\lambda})$:

?

$$p(\mathbf{w}) = \prod_{j=1}^{d} p(w_j) = \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\sigma^2/\lambda}} \exp\left(\frac{w_j^2}{2\sigma^2/\lambda}\right)$$

MAP Objective for Linear Regression

For MAP linear regression, we minimize the negative log posterior:

$$
\begin{aligned}
\mathbf{w}_{\text{MAP}} &= \arg\max_{\mathbf{w} \in \mathbb{R}^{d+1}} p(\mathbf{w} \mid \mathscr{D}) \\
&= \arg\max_{\mathbf{w} \in \mathbb{R}^{d+1}} \log p(\mathbf{w} \mid \mathscr{D}) \\
&= \arg\min_{\mathbf{w} \in \mathbb{R}^{d+1}} -\log p(\mathbf{w} \mid \mathscr{D}) \\
&= \arg\min_{\mathbf{w} \in \mathbb{R}^{d+1}} -\log \left(\frac{p(\mathscr{D} \mid \mathbf{w}) p(\mathbf{w})}{p(\mathscr{D})} \right) \\
&= \arg\min_{\mathbf{w} \in \mathbb{R}^{d+1}} -\log \big(p(\mathscr{D} \mid \mathbf{w}) p(\mathbf{w}) \big)
\end{aligned}
$$

Log-Prior for Gaussian Prior

$$p(\mathbf{w}) = \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\sigma^2/\lambda}} \exp\left(-\frac{w_j^2}{2\sigma^2/\lambda}\right)$$

$$\log p(\mathbf{w}) = \sum_{j=1}^{d} \log \left[\frac{1}{\sqrt{2\pi\sigma^2/\lambda}} \exp\left(-\frac{w_j^2}{2\sigma^2/\lambda}\right)\right]$$

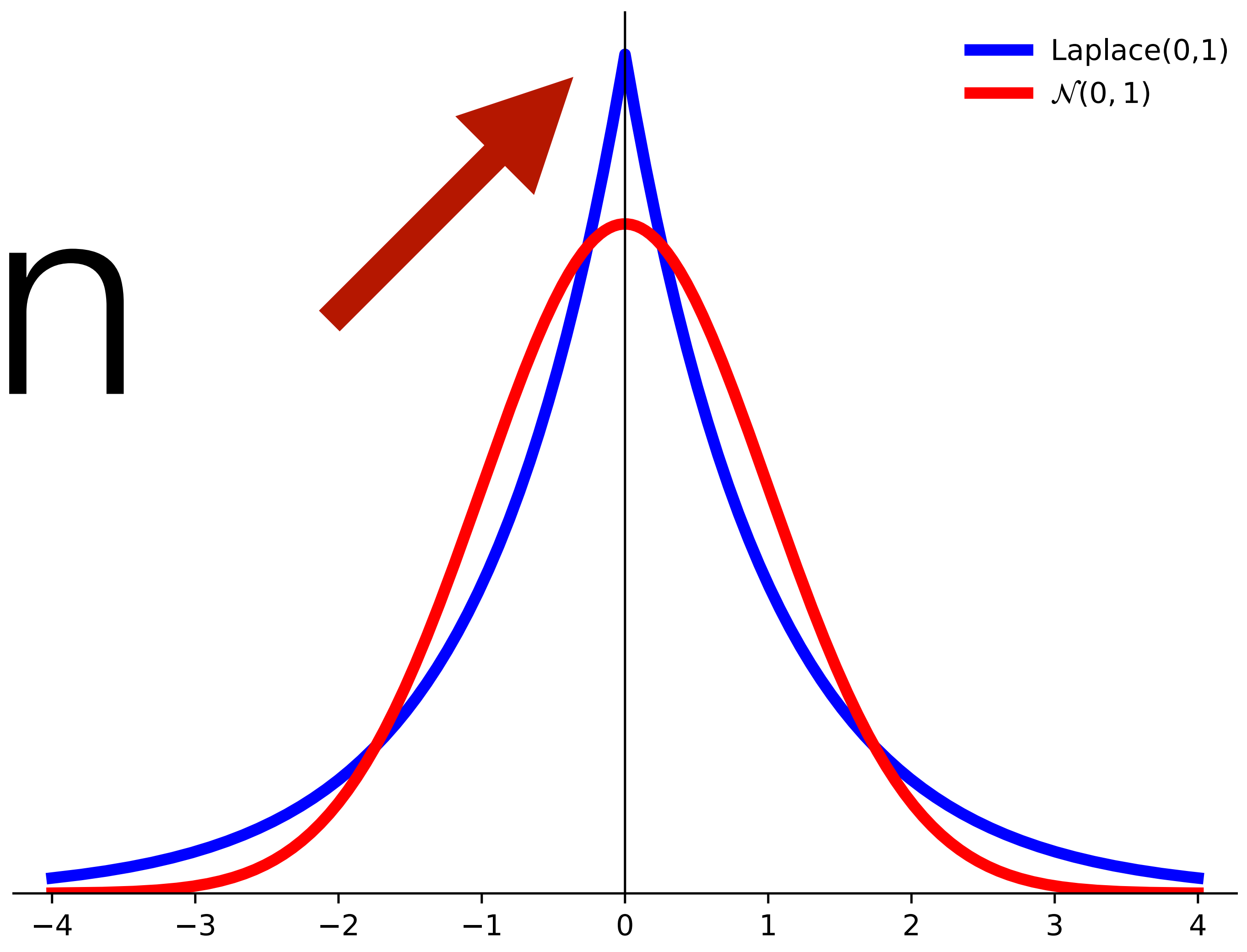
$$= \sum_{j=1}^{d} -\frac{1}{2}\log(2\pi\sigma^2/\lambda) - \frac{w_j^2}{2\sigma^2/\lambda}$$

$$= -\frac{d}{2}\log(2\pi\sigma^2/\lambda) - \frac{\lambda}{2\sigma^2}\sum_{j=1}^{d} w_j^2$$

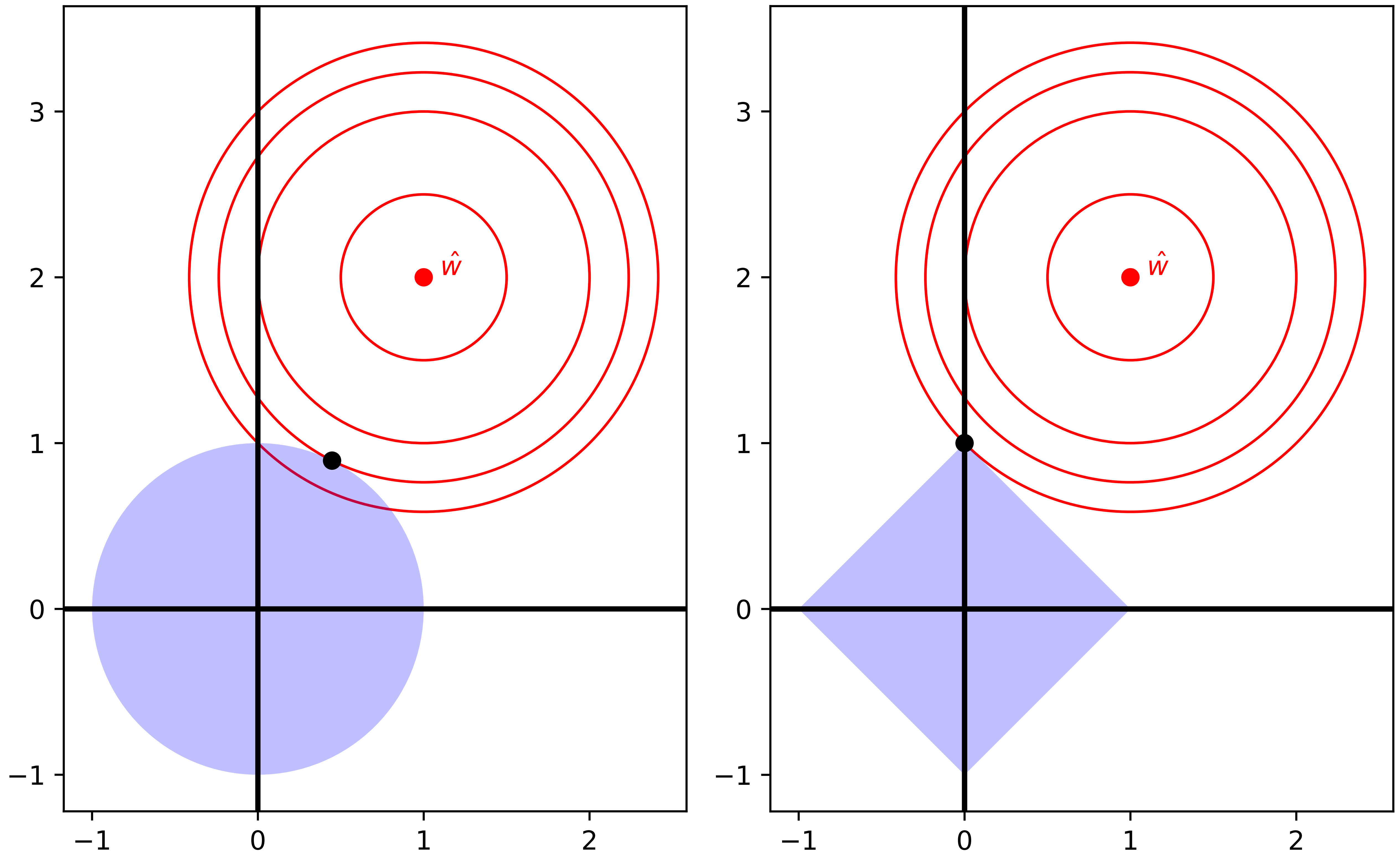
MAP Objective for Linear Regression

$$
\begin{aligned}
\mathbf{w}_{\text{MAP}} &= \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} -\log p(\mathscr{D} \mid \mathbf{w}) - \log p(\mathbf{w}) \\
&= \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} -\log p(\mathscr{D} \mid \mathbf{w}) + \frac{d}{2}\log(2\pi\sigma^2/\lambda) + \frac{\lambda}{2\sigma^2}\sum_{j=1}^{d} w_j^2 \\
&= \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(\mathbf{w}^T\mathbf{x}_i - y_i\right)^2 + \frac{d}{2}\log(2\pi\sigma^2/\lambda) + \frac{\lambda}{2\sigma^2}\sum_{j=1}^{d} w_j^2 \\
&= \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(\mathbf{w}^T\mathbf{x}_i - y_i\right)^2 + \frac{\lambda}{2\sigma^2}\sum_{j=1}^{d} w_j^2 \\
&= \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{2}\sum_{i=1}^{n}\left(\mathbf{w}^T\mathbf{x}_i - y_i\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d} w_j^2
\end{aligned}
$$

L1-Regularization



- It turns out that L1-regularization is equivalent to a Laplace prior on the weights (instead of a Gaussian)
- Note that Laplace puts **higher density at 0** than Gaussian
- L1-regularization prefers **sparse** solutions; those that set more weights to **exactly 0**
  - Not just due to higher density at 0
  - Also interaction between the shape of the loss and the shape of the regularization penalty



Summary

- **Regularization:** minimize the training cost plus a complexity penalty
  - $c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \text{cost}(f(\mathbf{x}_i; \mathbf{w}), y_i) + \lambda\, \text{penalty}(\mathbf{w})$
  - Only make a model more complex if it improves loss "enough"
  - The **hyperparameter** λ controls our notion of "enough"
- **L2 Regularization:** penalty is sum of squared weights: $\text{penalty}(\mathbf{w}) = \sum_{j=1}^{d} w_j^2$
  - L2 regularized linear regression corresponds to **MAP inference** with independent zero-mean **Gaussian priors** on each weight (except w_0)
- **L1 Regularization:** Penalty is sum of absolute values: $\text{penalty}(\mathbf{w}) = \sum_{j=1}^{d} |w_j|$
  - Corresponds to MAP inference with independent **Laplacian prior** on weights
  - Produces **sparse** solutions (many entries of $\mathbf{w}$ are set to **exactly 0**)