COMPUTER SCIENCE
INDIANA UNIVERSITY
School of Informatics and Computing
Bloomington

Course Review

Probability review

- Quantify uncertainty using probability theory
- Discussed sigma-algebras and probability measures
- Discussed random variables as functions of event-space
- Discussed relationships between random variables, including (in)dependence and conditional independence (belief network)
- Discussed operations, like expected value, marginalization, Bayes rule, chain rule

Exercise understanding MAP

- For MAP, our goal is to maximize the posterior: p(M I D)
  - M is the model
  - D is the data
- Is p(M I D) a PMF or a PDF?
- Example: Let p(x I M) be a Gaussian distribution
  - case 1: assume picking mean M, from R
  - case 2: assume picking mean M in set {-1, 0, 1}
  - case 3: assume picking mean M from set [-5, 5]

Exercise: probability

- Suppose that we have created a machine learning algorithm that predicts whether a link will be clicked with 99% sensitivity (TPR) and 99% specificity (FPR). The rate the link is actually clicked is 1/1000 visits to a website. If we predict the link will be clicked on a specific visit, what is the probability it will actually be clicked?

- Let C be binary RV, with C = 1 indicating predict click

- $p(C = 1 \mid y = 1) = \text{TPR}$

- $p(C = 1 \mid y = 0) = 1\text{-FPR}$

Linear regression

- Assume p(y l x) is Gaussian distributed with a fixed variance for the noise term epsilon

$$\nabla E(\mathbf{w}) = \mathbf{0} \text{ we find that}$$

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

Learning the variance

- What if we now also wanted to learn the variance? How does the problem formulation change?

- Let’s do an exercise!

Regularization

- MAP for linear regression with
- Gaussian prior on weights: l2 regularization
- Laplace prior on weights: l1 regularization
- What does it mean to put a distribution over our parameters?
    - a constraint region, w in [-5, 5], specifies must come from that set, but does not encode a preference or likelihood
    - a distribution encodes a preference for particular values

Regularization intuition

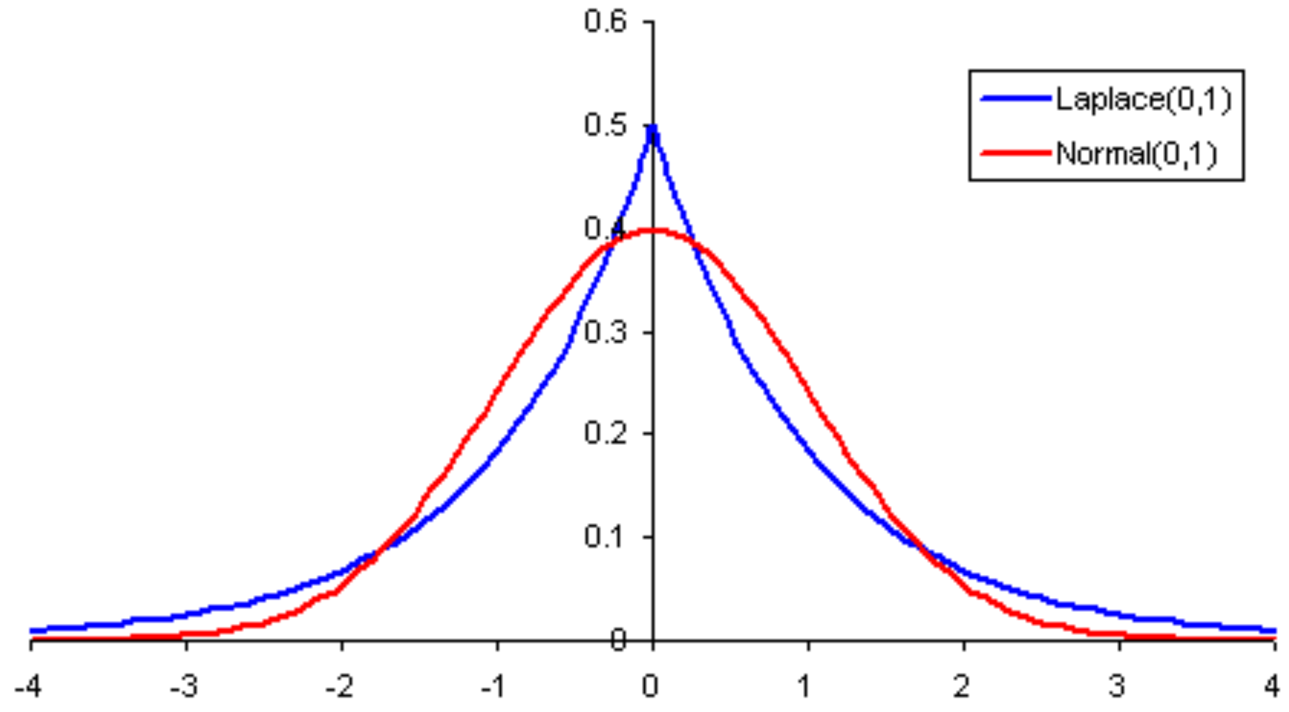
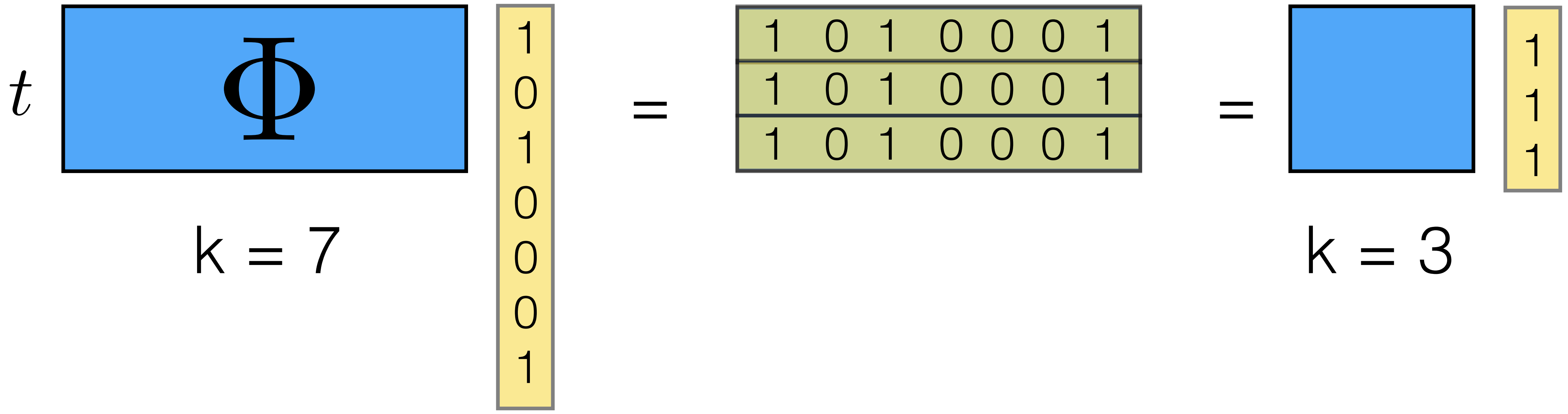


Figure 4.5: A comparison between Gaussian and Laplace priors. The Gaussian prior prefers the values to be near zero, whereas the Laplace prior more strongly prefers the values to equal zero.

l1 regularization

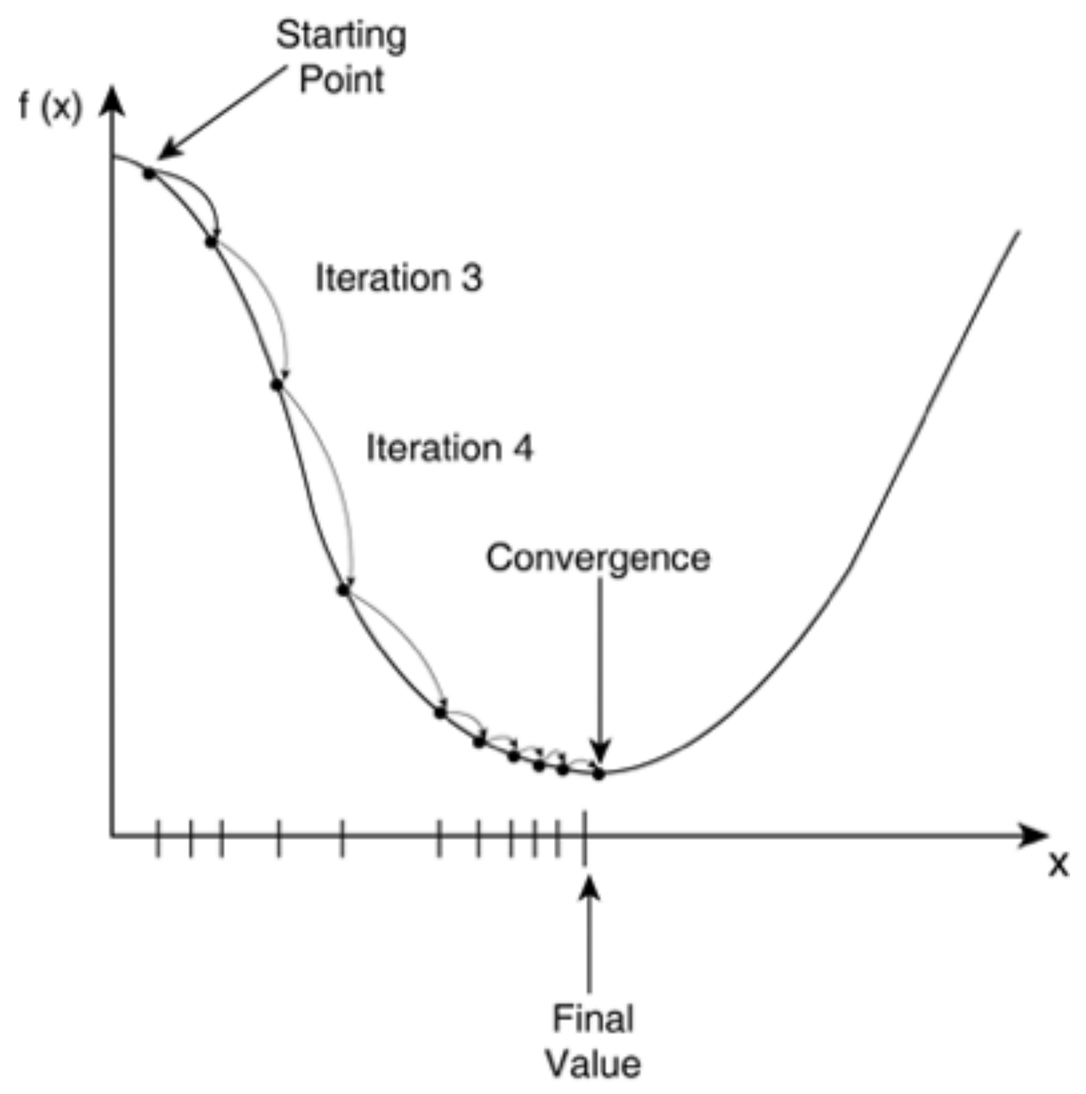
- Feature selection, as well as preventing large weight



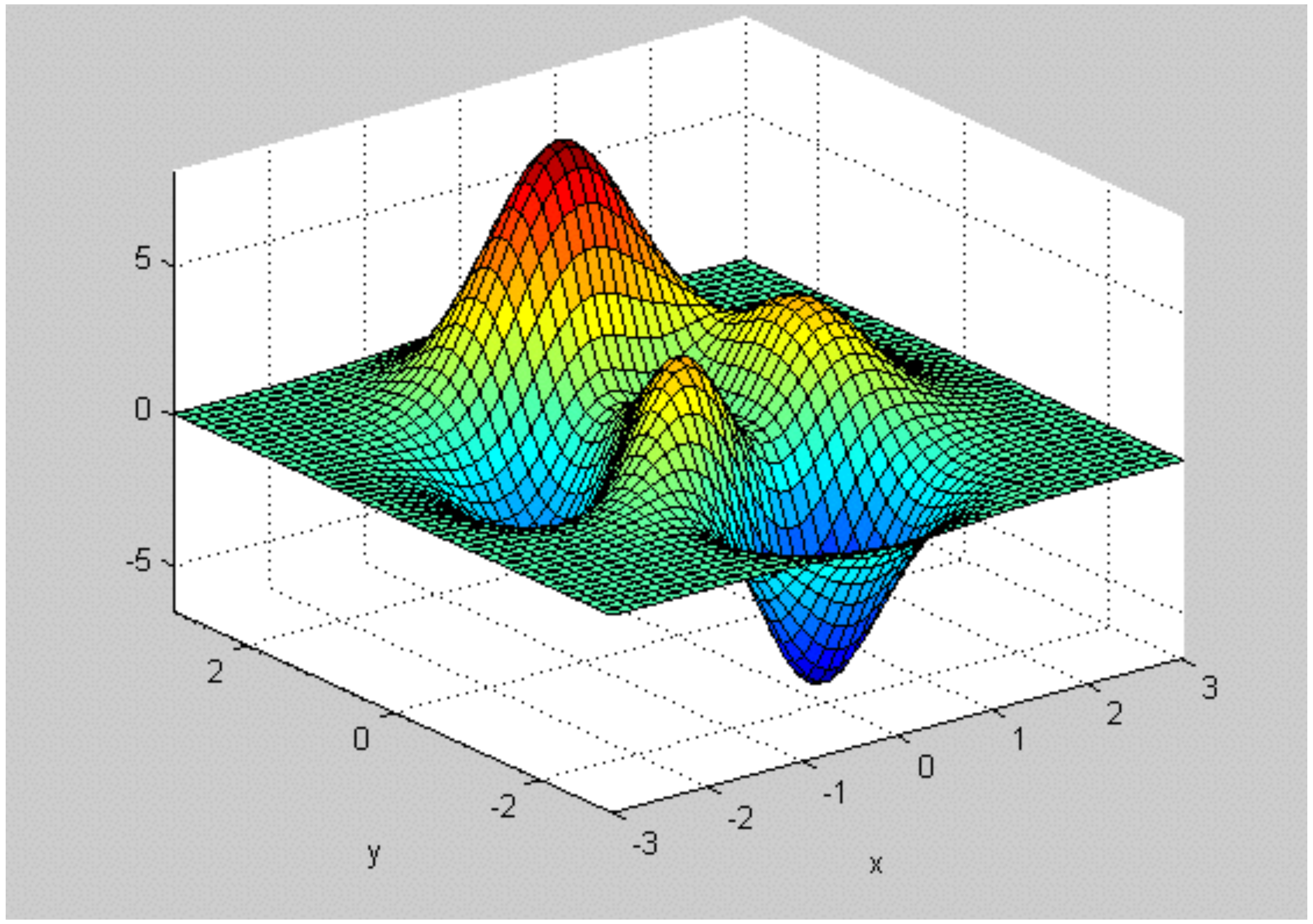
- How do we solve this optimization?

$$\min_{\mathbf{w}\in\mathbb{R}^d} \|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_1$$

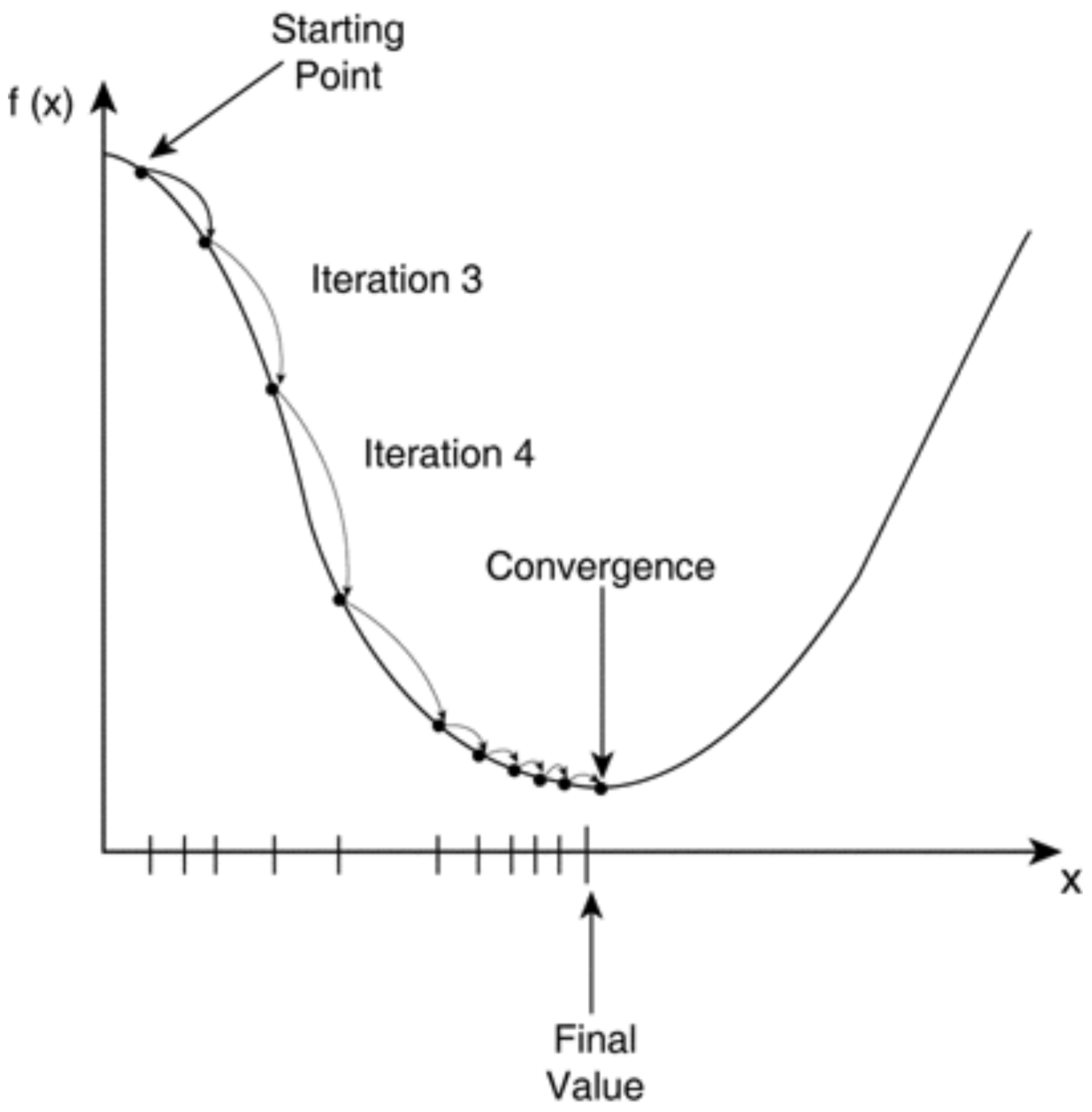
Gradient descent



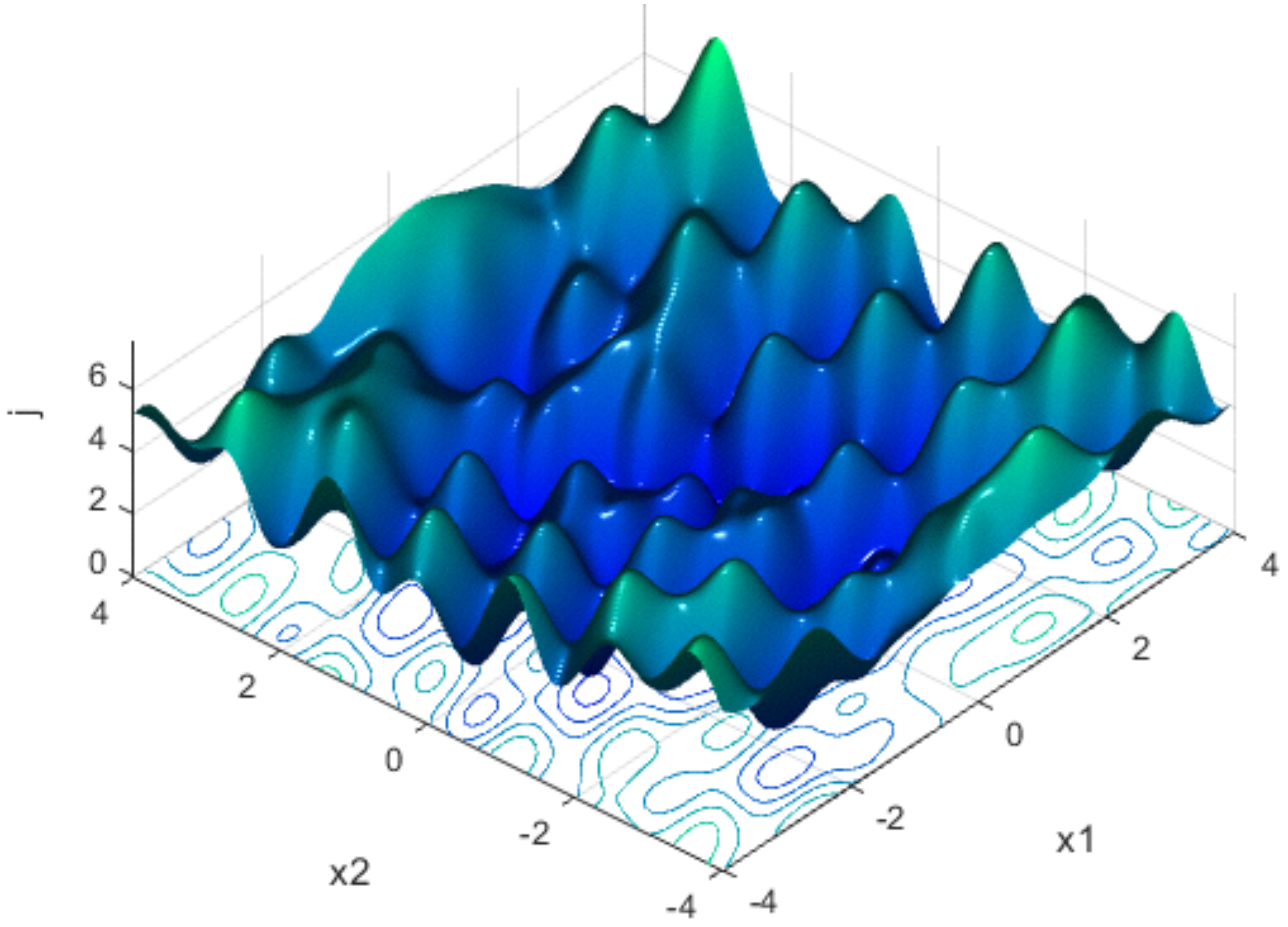
Multivariate optimization



Convex versus nonconvex



Convex function



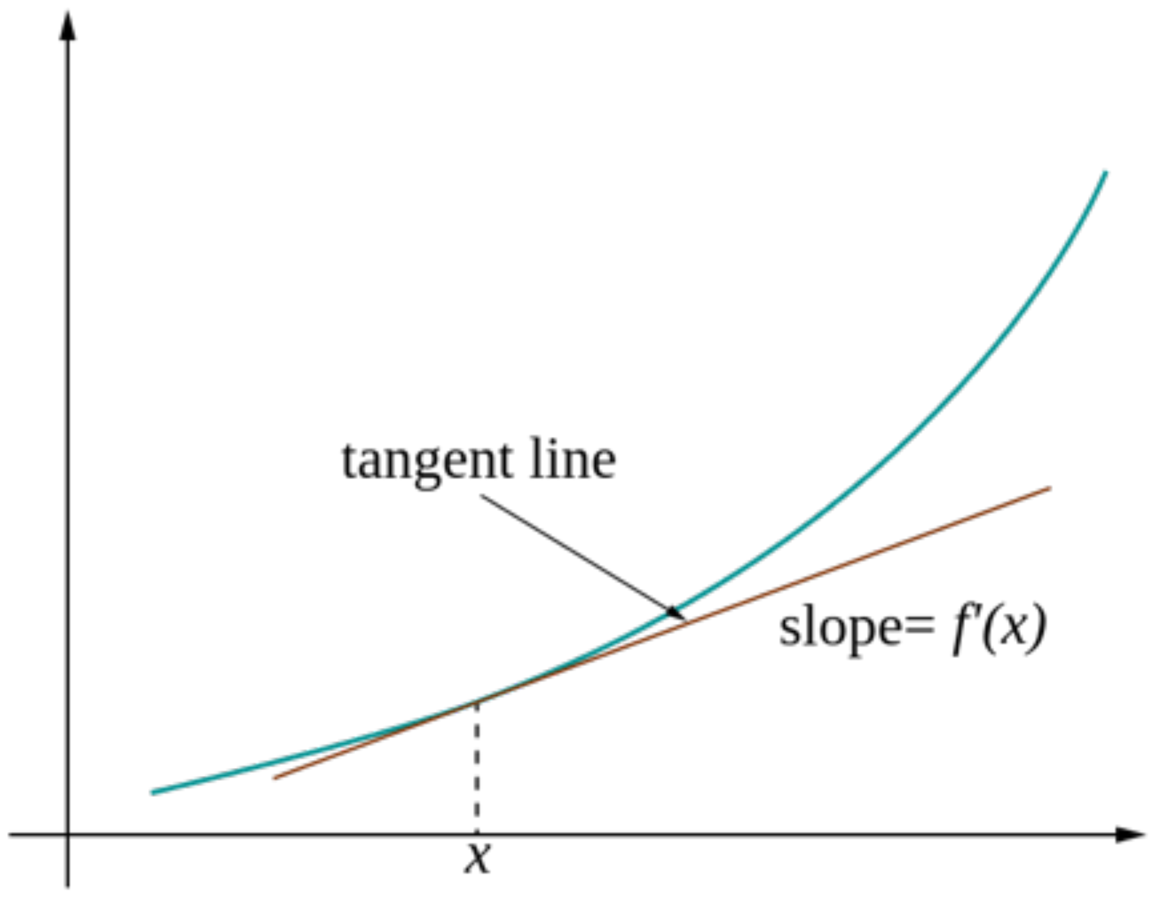
Non-convex function

First-order conditions

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \ \ldots, \ \frac{\partial f}{\partial x_d} \right]$$

- Gradient = 0 provides a minimum, maximum or saddle point
- Gradient gives direction of steepest ascent
    - gives slope of tangent line

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$



First-order and second-order

- We took the second-order Taylor series expansion

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2}(x - x_0)^2 f''(x_0).$$

$$f'(x) \approx f'(x_0) + (x - x_0)f''(x_0) = 0.$$

- For second-order gradient descent, we find a stationary point of this approximation to get our new point

$$x = x_0 - \frac{f'(x_0)}{f''(x_0)}.$$

- For first-order, we replace the second-derivative with an approximation (which corresponds to the step-size)

$$x = x_0 - \eta f'(x_0)$$

Multivariate case

- Similar approach, but second-order approximation includes
  - gradient (generalization of first derivative) and
  - Hessian (generalization of second derivative)

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T \cdot (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \cdot H_{f(\mathbf{x}_0)} \cdot (\mathbf{x} - \mathbf{x}_0)\,,$$

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \left(H_{f(\mathbf{x}^{(i)})}\right)^{-1} \cdot \nabla f(\mathbf{x}^{(i)}),$$

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_k}\right) \qquad H_{f(\mathbf{x})} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_k} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & & \\ \vdots & & \ddots & \\ \frac{\partial^2 f}{\partial x_k \partial x_1} & & & \frac{\partial^2 f}{\partial x_k^2} \end{bmatrix}$$

Pros and Cons

- First-order uses a much more significant approximation, and selecting step-size can be difficult
- Second-order is much more expensive, but Hessian provides curvature information a so a very good descent direction
- Quasi-second order methods: try to balance between the two, by approximating some of this curvature information
  - e.g., approximate only the diagonal of the Hessian matrix
  - e.g., Adadelta

First-order conditions

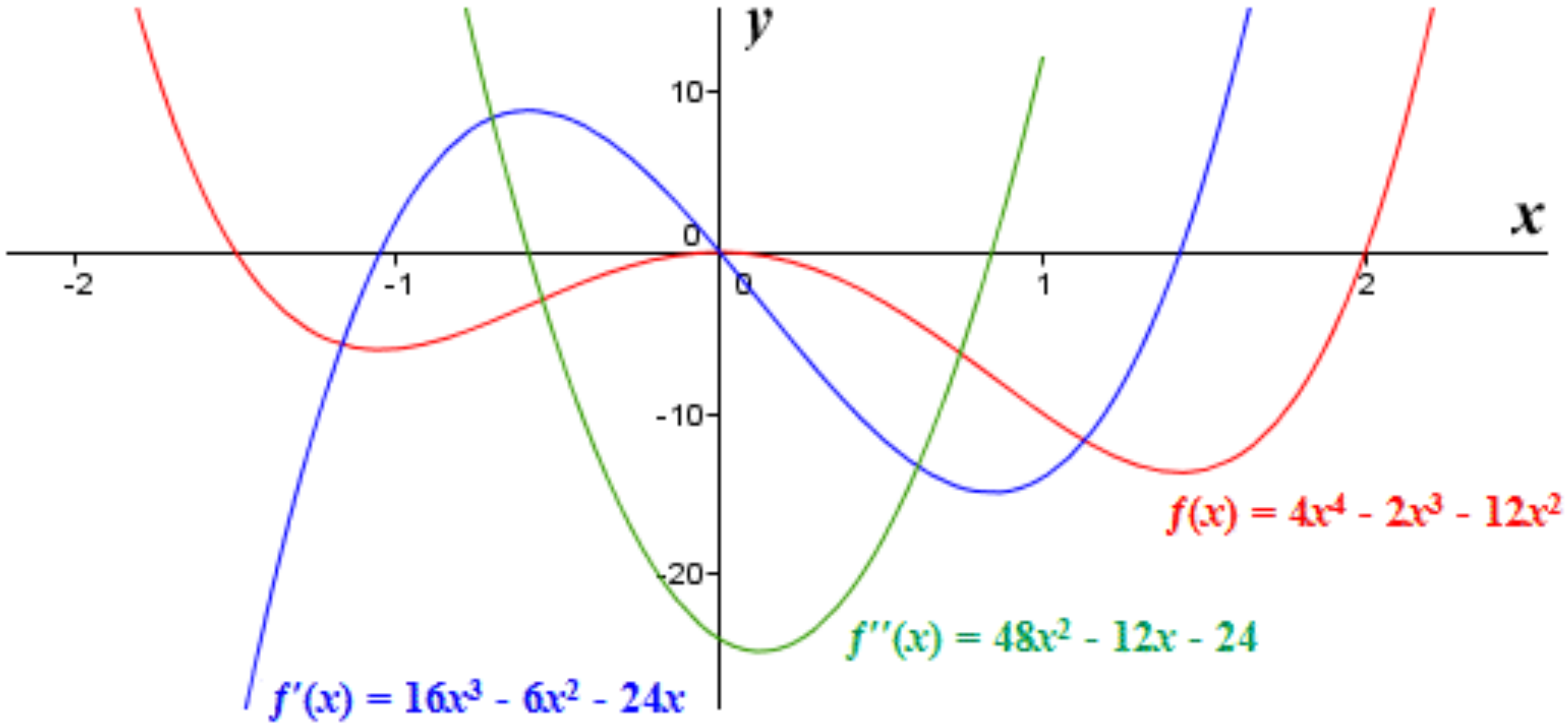
$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \ \ldots, \ \frac{\partial f}{\partial x_d}\right]$$

- Gradient = 0 provides a minimum, maximum or saddle point
- Gradient gives direction of steepest ascent

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

Second derivative test

- If $f''(x) < 0$ then f has a local maximum at x.
- If $f''(x) > 0$ then f has a local minimum at x.
- If $f''(x) = 0$, the test is inconclusive.



Directional second derivative

$$\text{At stationary point } \mathbf{w}^*, \nabla f(\mathbf{w}) = \mathbf{0}$$

$$\mathbf{w}(t) = \mathbf{w}^* + t\mathbf{w}$$

$$g(t) = f(\mathbf{w}(t))$$

$$g'(0) = \nabla f(\mathbf{w}(t))^\top \mathbf{w} = 0$$

$$g''(0) = \mathbf{w}^\top \nabla^2 f(\mathbf{w}(t))^\top \mathbf{w}$$

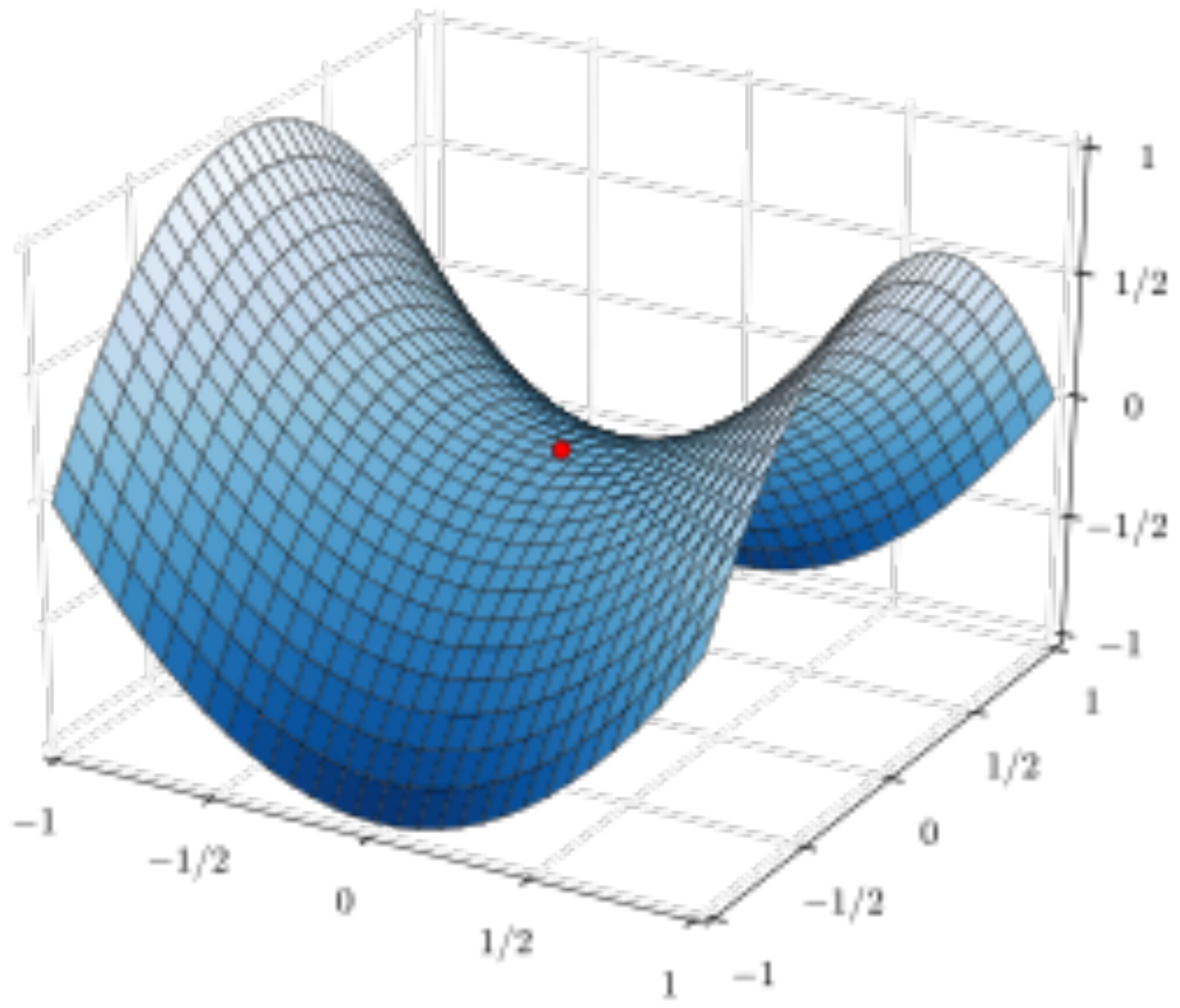
Intuition for second derivative test in univariate setting

$$0 < f''(x) = \lim_{h\to 0} \frac{f'(x+h) - f'(x)}{h} = \lim_{h\to 0} \frac{f'(x+h) - 0}{h} = \lim_{h\to 0} \frac{f'(x+h)}{h}.$$

Thus, for h sufficiently small we get

$$\frac{f'(x+h)}{h} > 0$$

Hessian intuition



Hessian

- If $f''(x) < 0$ then f has a local maximum at x.
- If $f''(x) > 0$ then f has a local minimum at x.
- If $f''(x) = 0$, the test is inconclusive.

$$g''(0) = \mathbf{w}^\top \nabla^2 f(\mathbf{w}(t))^\top \mathbf{w}$$

$$\nabla^2 f = \mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Hessian

- If $f''(x) < 0$ then f has a local maximum at x.
- If $f''(x) > 0$ then f has a local minimum at x.
- If $f''(x) = 0$, the test is inconclusive.

$$g''(0) = \mathbf{w}^\top \nabla^2 f(\mathbf{w}(t))^\top \mathbf{w}$$

$$\text{Positive definite: } \mathbf{w}^\top \mathbf{H}\mathbf{w} > 0 \text{ for all } \mathbf{w} \neq \mathbf{0}$$

$$\text{Negative definite: } \mathbf{w}^\top \mathbf{H}\mathbf{w} < 0 \text{ for all } \mathbf{w} \neq \mathbf{0}$$

- If H is positive definite at x, then local minimum at x
- If H is negative definite at x, then local maximum at x
- If H has both positive and negative eigenvalues at x, then a saddle point at x

Stochastic gradient descent

- Batch gradient descent uses a batch of samples to compute a sample average, i.e., an estimate of the true expected error
- Stochastic gradient descent only uses 1 instance to obtain an (unbiased) estimate of the true expected error
- For mini-batch, we use a small random subset of points, e.g. 10 points out of 10k. Why is this ok?
- Can you have first & second-order stochastic gradient descent?

l1 regularization

- How do we solve this optimization?

$$\min_{\mathbf{w}\in\mathbb{R}^d} \|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2+\lambda\|\mathbf{w}\|_1$$

- What is the issue?

Constrained optimization

- Do not expect you to know how to do this for the final
- Idea: introduce Lagrange multipliers to bring up constraints into the objective function (like regularizers)
- Solve for Lagrange multipliers as well

Classification

- Logistic regression
- Multinomial logistic regression
- Support vector machine (SVMs)
- Naive Bayes

Generalized linear models

- Can pick any exponential family distribution for p(y l x)
- If p(y l x) is Gaussian, then we get linear regression with <x, w> approximating E[y l x]
- If p(y l x) is Bernoulli, then we get logistic regression with sigmoid(<x, w>) approximating E[y l x]
- If p(y l x) is Poisson, then we get Poisson regression with exp(<x, w>) approximating E[y l x]
- If p(y l x) is a Multinomial (multiclass), then we get multinomial logistic regression with softmax(<x, w>) approximating E[y l x]
- For all of these, just estimating w to get this dot product

Exercise

- What model might you use if
  - we have binary features and targets?
  - binary targets and continuous features?
  - positive targets?
  - categorical features with a large number of categories?
  - multi-class targets, with continuous features?
- When might logistic regression do better than linear regression?
- When might Poisson regression do better than linear regression?

Representation learning

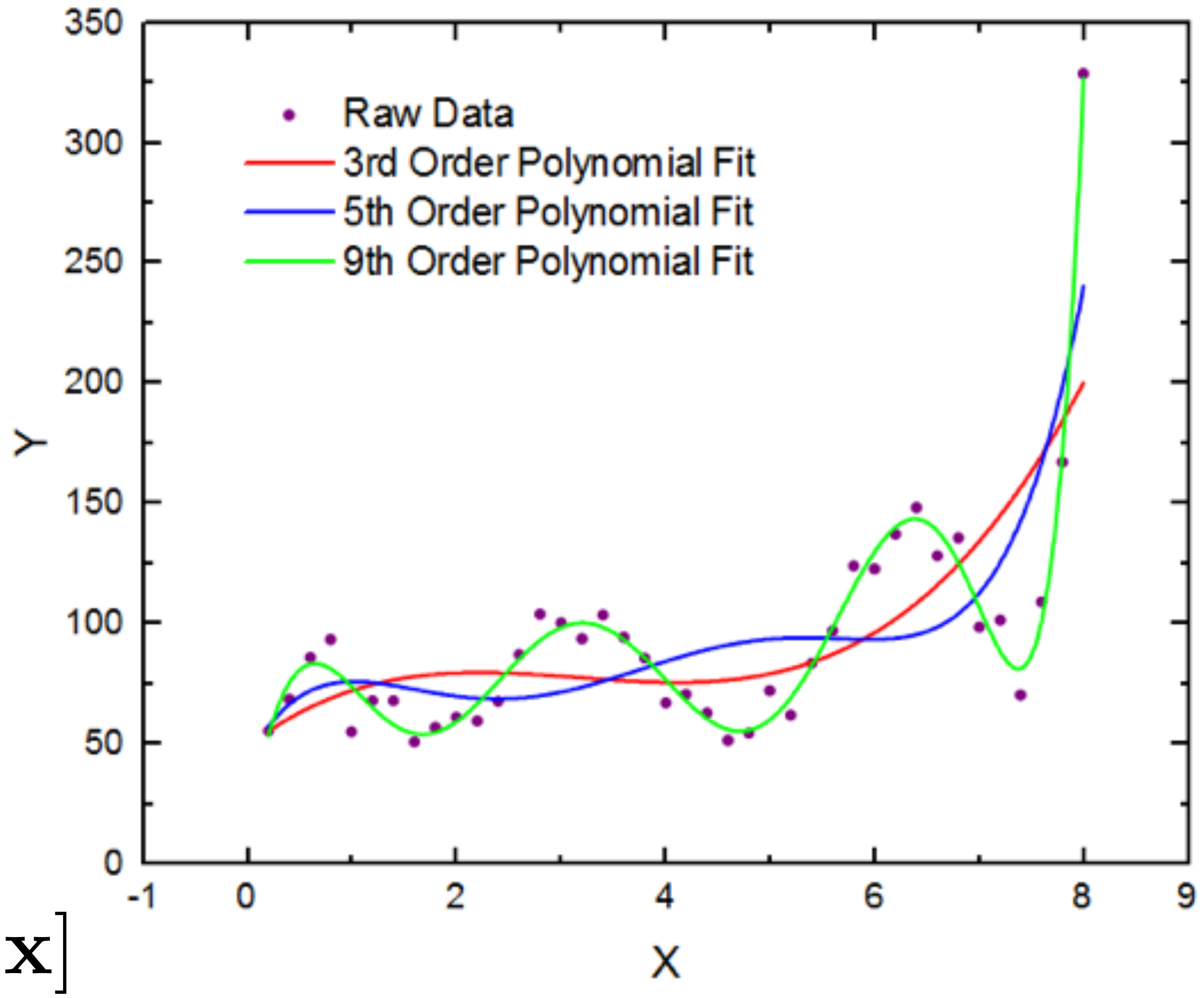
- Convert linear predictors into non-linear predictors, e.g.

$$\mathbf{x} \rightarrow$$

$$\text{2nd-order polynomial}(\mathbf{x}) =$$

$$w_6 x_1^2 + w_5 x_2^2 + w_4 x_1 x_2$$

$$+ w_2 x_2 + w_1 x_1 + w_0$$



Probabilistic assumption after transformation:

$$f(\text{polynomial}(\mathbf{x})^\top \mathbf{w}) = \mathbb{E}[Y|\mathbf{x}]$$

Radial basis function network

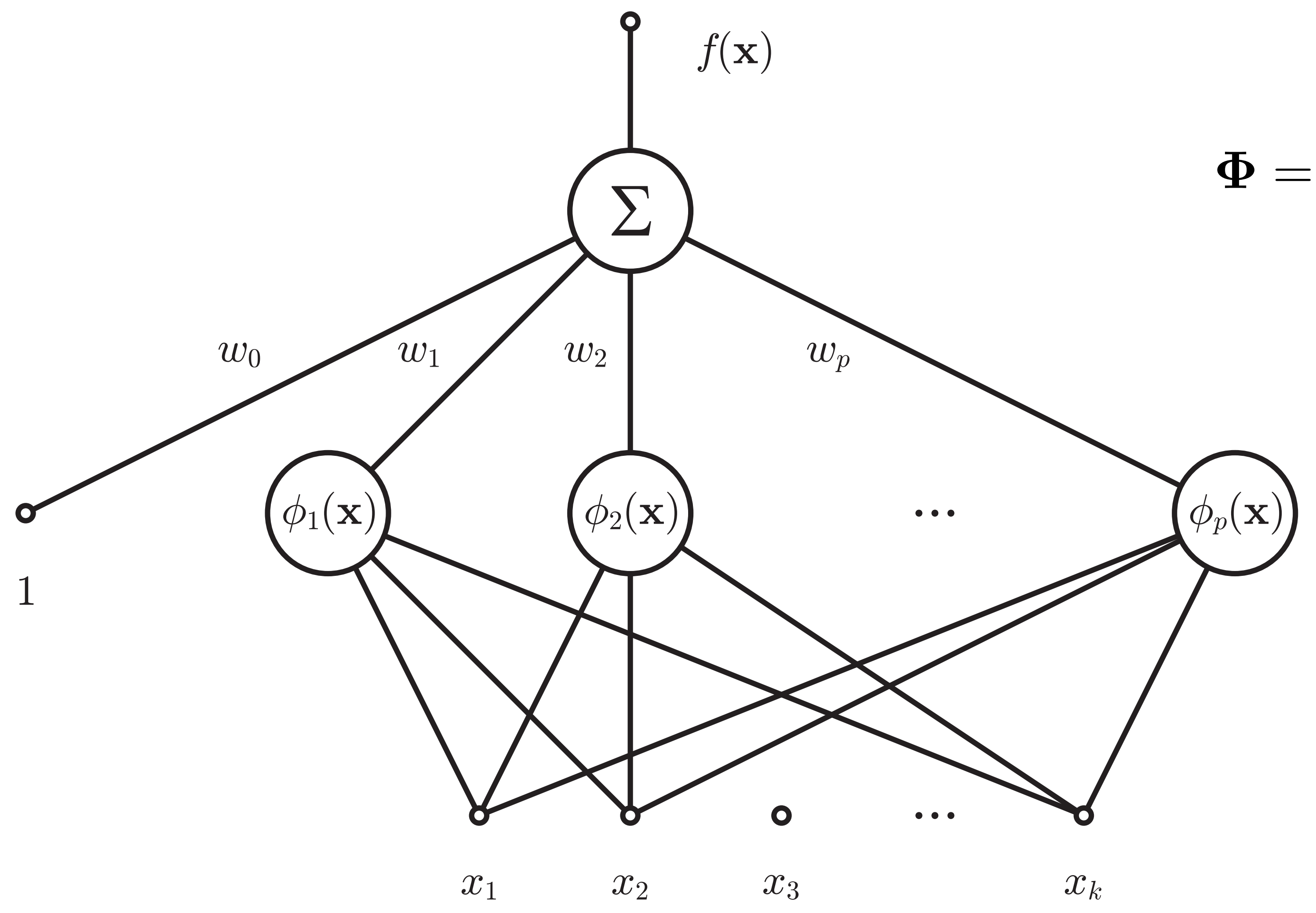


Figure 7.1: Radial basis function network.

$$\mathbf{\Phi} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_p(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & & \\ \vdots & & \ddots & \\ \phi_0(\mathbf{x}_n) & & & \phi_p(\mathbf{x}_n) \end{bmatrix}$$

e.g., $\phi_j(\mathbf{x}) = e^{-\frac{\left\|\mathbf{x}-\mathbf{c}_j\right\|^2}{2\sigma_j^2}}$,

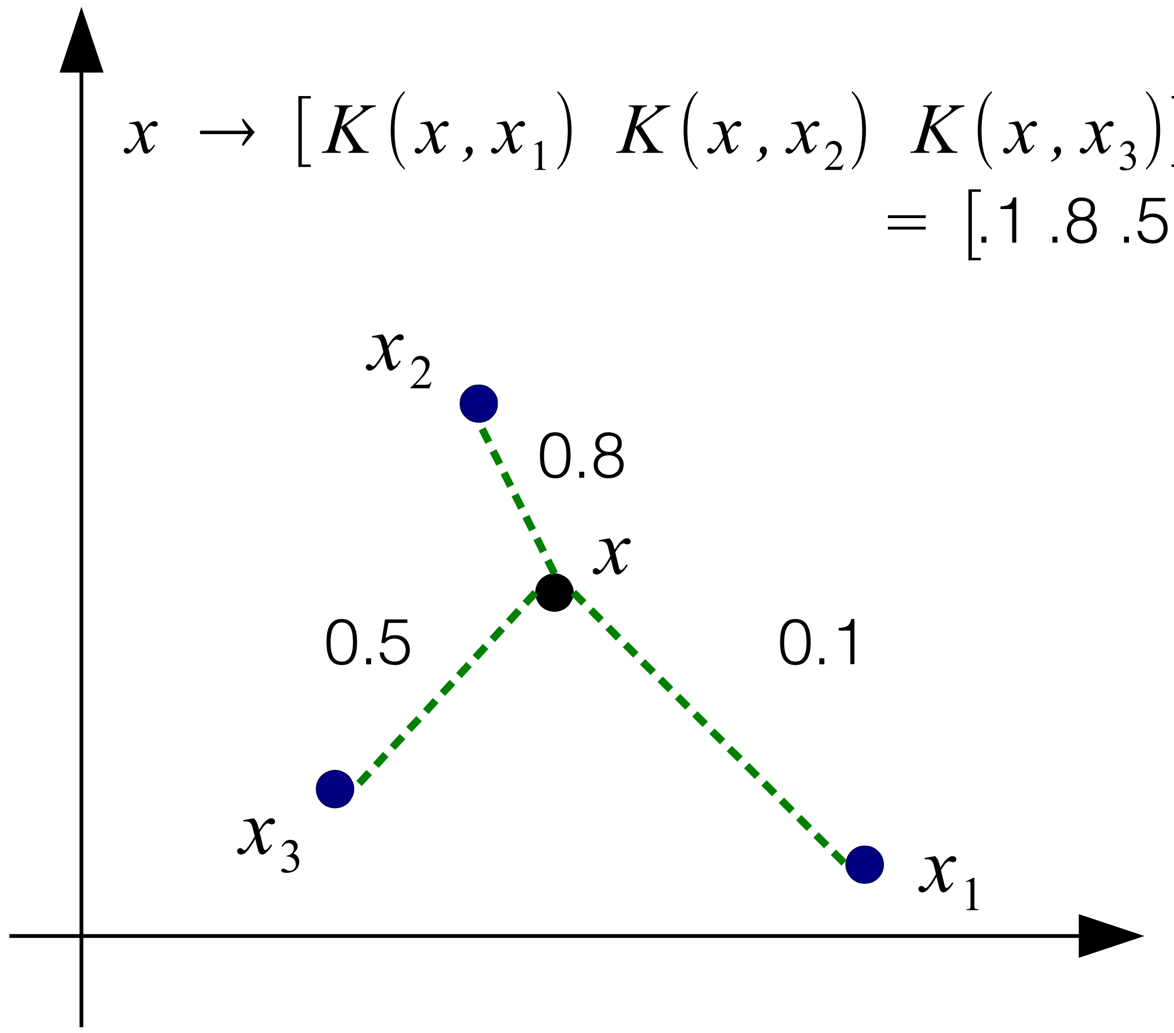
$$\begin{aligned} f(\mathbf{x}) &= w_0 + \sum_{j=1}^{p} w_j \phi_j(\mathbf{x}) \\ &= \sum_{j=0}^{p} w_j \phi_j(\mathbf{x}) \end{aligned}$$

RBF and Kernel representation

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{-\|\mathbf{x} - \mathbf{x}'\|_2^2}{\sigma^2}\right) \qquad f(\mathbf{x}) = \sum_{i=1}^{k} w_i k(\mathbf{x}, \mathbf{x}_i)$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} k(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ k(\mathbf{x}, \mathbf{x}_k) \end{bmatrix}$$

$$x \rightarrow [K(x, x_1) \;\; K(x, x_2) \;\; K(x, x_3)]' = [.1 \; .8 \; .5]'$$



Exercise

- Imagine you wanted to learn to predict if a patient has a certain genetic disease
- Your input data is a string, corresponding to a part of their DNA (e.g., ACCGGGTA)
- This data is not numeric. How might you learn on this data?

Neural networks for representation learning

- Goal is to learn this representation phi
- The hidden layer(s) produce phi
- The last layer corresponds to any generalized linear model
- What is the probabilistic assumption using
  - one hidden layer with a ReLu activation on the first layer
  - a sigmoid activation on the last layer, with cross-entropy?

$$\sigma(\mathbf{h}\mathbf{w}) = \mathbb{E}[Y|\mathbf{x}] = p(y=1|\mathbf{x})$$

$$\sigma(\mathrm{relu}(\mathbf{x}\mathbf{W}^{(2)})\mathbf{w}) = \mathbb{E}[Y|\mathbf{x}]$$

Exercise

- You want to train a neural network on a small training data set. Discuss two strategies that would prevent or reduce overfitting of such a network.

- What can you do to be more confident about if your strategies properly protected against overfitting?