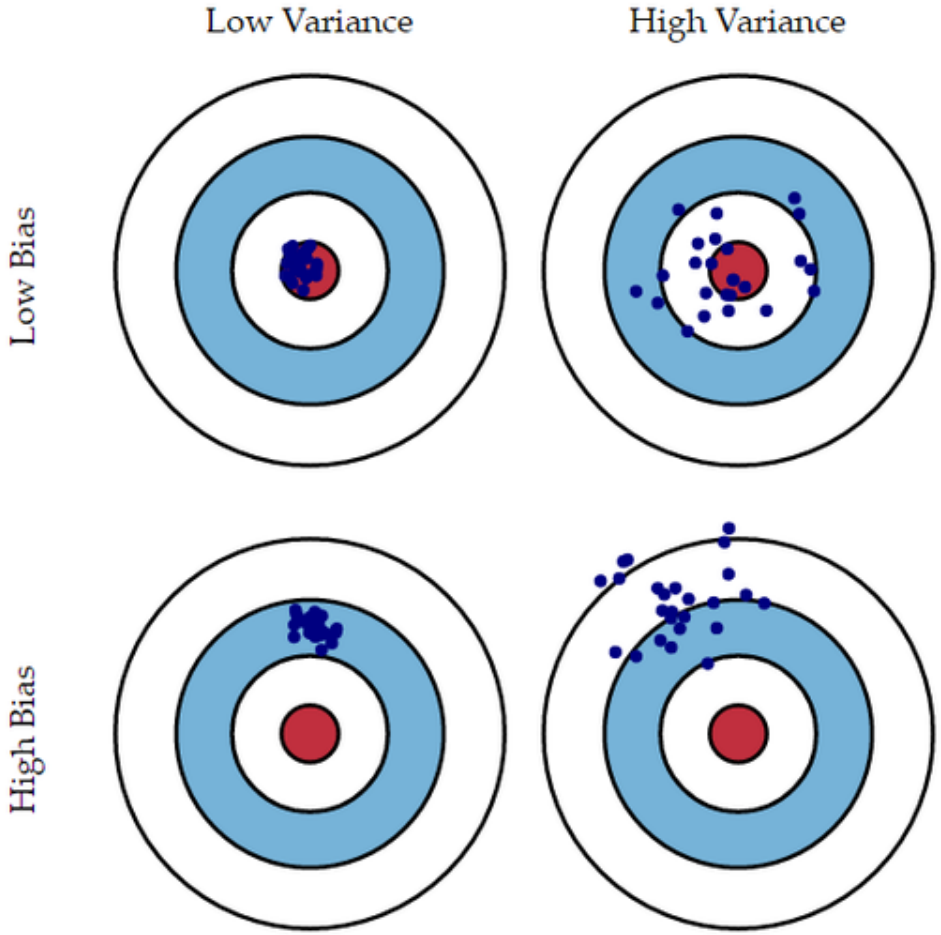
CSC 311: Introduction to Machine Learning

Lecture 4 - Bias-Variance Decomposition, Ensemble Method I: Bagging, Linear Classification

Amir-massoud Farahmand & Emad A.M. Andrews

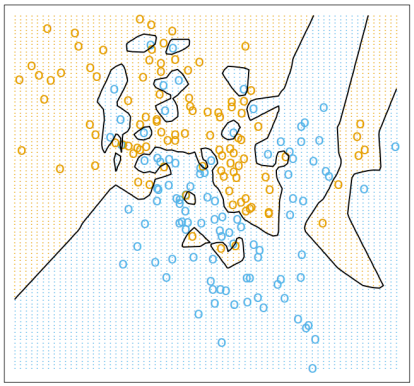
University of Toronto

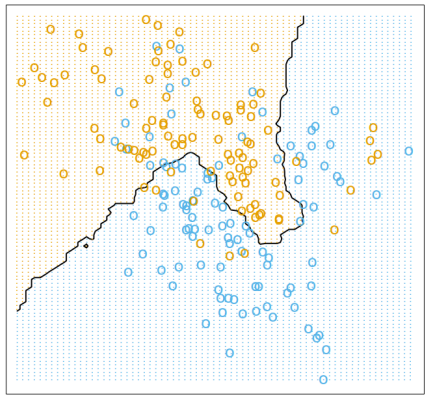
Bias-Variance Decomposition

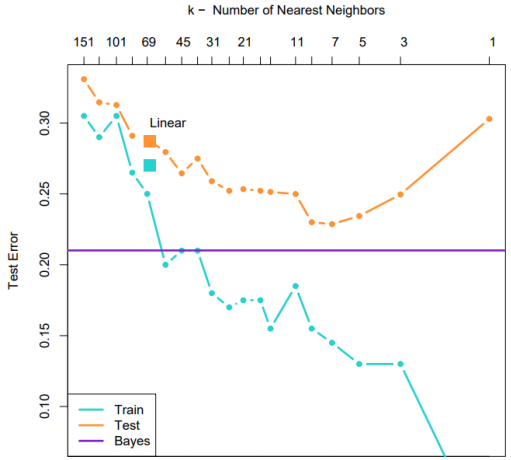


Bias-Variance Decomposition

- Recall that overly simple models underfit the data, and overly complex models overfit.







- We quantify this effect in terms of the bias-variance decomposition.

Bias-Variance Decomposition for the Mean Estimator

- For the next few slides, we consider the simple problem of estimating the mean of a random variable using data.
- Consider a r.v. Y with an unknown distribution p. This random variable has an (unknown) mean $m = \mathbb{E}[Y]$ and variance $\sigma^2 = \text{Var}[Y] = \mathbb{E}\left[(Y - m)^2\right]$.
- Given: a dataset $\mathcal{D} = \{Y_1, \ldots, Y_n\}$ with independently sampled $Y_i \sim p$.
- How can we estimate m using $\mathcal{D}$?
- Consider an algorithm that receives $\mathcal{D}$, does some processing on data, and outputs a number. The goal of this algorithm is to provide an estimate of m. Let us denote it by $h(\mathcal{D})$.
- Some good and bad examples:
  - Sample average: $h(\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n} Y_i$
  - Single-sample estimator: $h(\mathcal{D}) = Y_1$
  - Zero estimator: $h(\mathcal{D}) = 0$
- How well do they perform?

Bias-Variance Decomposition for the Mean Estimator

- How can we assess the performance of a particular $h(\mathcal{D})$?
- Ideally, we want $h(\mathcal{D})$ be exactly equal to $m = \mathbb{E}[Y]$. But this might be too much to ask. (why?)
- What we can hope for is that $h(\mathcal{D}) \approx m$. How can we quantify the accuracy of approximation?
- We use the squared error $\text{err}(\mathcal{D}) = |h(\mathcal{D}) - m|^2$ as a measure of quality. This is the familiar squared error loss function in regression.
- The error $\text{err}(\mathcal{D})$ is a r.v. itself. (why?) For a dataset $\mathcal{D} = \{Y_1, \ldots, Y_n\}$ the loss $\text{err}(D)$ might be small, but for another $\mathcal{D}' = \{Y'_1, \ldots, Y'_n\}$ (still with $Y'_i \sim p$) the loss $\text{err}(D')$ might be large. We would like to quantify the "average" error.
- We focus on the expectation of $\text{err}(\mathcal{D})$, i.e.,

$$\mathbb{E}[\text{err}(\mathcal{D})] = \mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - m|^2\right].$$

- Note that the dataset $\mathcal{D}$ is random and this expectation is w.r.t. its randomness.

Bias-Variance Decomposition for the Mean Estimator

- We would like to understand what determines $\mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - m|^2\right]$ by looking more closely at it.
- We can decompose $\mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - m|^2\right]$ by adding and subtracting $\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]$ inside $|\cdot|$:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - m|^2\right] =& \mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right] + \mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right] - m|^2\right] \\ =& \mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]|^2\right] + \mathbb{E}_{\mathcal{D}}\left[|\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right] - m|^2\right] + \\ & 2\mathbb{E}_{\mathcal{D}}\left[\left(h(\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]\right)\left(\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right] - m\right)\right].\end{aligned}$$

- Let us simplify the right hand side (RHS).
- Recall that if X is a random variable and f is a function, the quantity $f(X)$ is a random variable. But its expectation $\mathbb{E}\left[f(X)\right]$ is not. We can say that the expectation takes the randomness away. So $\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]$ is not a random variable anymore. We have

$$\mathbb{E}_{\mathcal{D}}\left[|\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right] - m|^2\right] = |\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right] - m|^2.$$

Bias-Variance Decomposition for the Mean Estimator

- Let us consider $\mathbb{E}_{\mathcal{D}}\left[\left(h(\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]\right)\left(\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right] - m\right)\right]$. To reduce the clutter, we denote $\bar{m} = \mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]$. Note that $\bar{m}$ is an expectation of a r.v., so it is not random. This means that $\mathbb{E}\left[\bar{m}h(\mathcal{D})\right] = \bar{m}\mathbb{E}\left[h(\mathcal{D})\right]$.
- We have

$$\mathbb{E}_{\mathcal{D}}\left[\left(h(\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]\right)\left(\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right] - m\right)\right] =$$
$$\mathbb{E}_{\mathcal{D}}\left[(h(\mathcal{D}) - \bar{m})(\bar{m} - m)\right] = (\bar{m} - m)\underbrace{\left(\mathbb{E}\left[h(\mathcal{D})\right] - \bar{m}\right)}_{=0} = 0$$

Bias-Variance Decomposition for the Mean Estimator

Bias-Variance Decomposition

$$\mathbb{E}_{\mathcal{D}}\left[\left|h(\mathcal{D}) - m\right|^2\right] = \underbrace{\left|\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right] - m\right|^2}_{\text{bias}} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left|h(\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]\right|^2\right]}_{\text{variance}}.$$

- Bias: The error of the expected estimator (over draws of dataset $\mathcal{D}$) compared to the mean $m = \mathbb{E}[Y]$ of the random variable Y.
- Variance: The variance of a single estimator $h(\mathcal{D})$ (whose randomness comes from $\mathcal{D}$).
- This is for an estimator of a mean of a random variable. We shall extend this decomposition to more general estimators too.

Bias-Variance Decomposition for the Mean Estimator: Examples

Bias-Variance Decomposition

$$\mathbb{E}_{\mathcal{D}}\left[\left|h(\mathcal{D})-m\right|^2\right]=\underbrace{\left|\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]-m\right|^2}_{\text{bias}}+\underbrace{\mathbb{E}_{\mathcal{D}}\left[\left|h(\mathcal{D})-\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]\right|^2\right]}_{\text{variance}}.$$

- Let us compute the bias and variance of a few estimators. Recall that $m=\mathbb{E}\left[Y\right]$ and $\sigma^2=\mathrm{Var}\{Y\}=\mathbb{E}\left[(Y-m)^2\right]$.
- Sample average: $h(\mathcal{D})=\frac{1}{n}\sum_{i=1}^n Y_i$.
  - Bias $\left|\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]-m\right|^2=\left|\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n Y_i\right]-m\right|^2=$ $\left|\frac{1}{n}\sum_{i=1}^n\mathbb{E}\left[Y_i\right]-m\right|^2=\left|\frac{1}{n}\sum_{i=1}^n m-m\right|^2=0.$
  - Variance: $\mathbb{E}\left[\left|h(\mathcal{D})-\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]\right|^2\right]=\mathbb{E}\left[\left|\frac{1}{n}\sum_{i=1}^n Y_i-\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n Y_i\right]\right|^2\right]=$ $\mathbb{E}\left[\left|\frac{1}{n}\sum_{i=1}^n(Y_i-m)\right|^2\right]=\frac{1}{n^2}\sum_{i=1}^n\mathbb{E}\left[(Y_i-m)^2\right]=\frac{1}{n^2}n\sigma^2=\frac{\sigma^2}{n}.$
  - $\mathbb{E}_{\mathcal{D}}\left[\left|h(\mathcal{D})-m\right|^2\right]=\text{bias}+\text{variance}=0+\frac{\sigma^2}{n}.$

Bias-Variance Decomposition for the Mean Estimator: Examples

Bias-Variance Decomposition

$$\mathbb{E}_{\mathcal{D}}\left[\left|h(\mathcal{D})-m\right|^2\right]=\underbrace{\left|\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]-m\right|^2}_{\text{bias}}+\underbrace{\mathbb{E}_{\mathcal{D}}\left[\left|h(\mathcal{D})-\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]\right|^2\right]}_{\text{variance}}.$$

- Single-sample estimator: $h(\mathcal{D})=Y_1$
  - The algorithm behind this estimator only looks at the first data point and ignores the rest.
  - Bias $\left|\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]-m\right|^2=\left|\mathbb{E}\left[Y_1\right]-m\right|^2=\left|m-m\right|^2=0$.
  - Variance: $\mathbb{E}\left[\left|h(\mathcal{D})-\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]\right|^2\right]=\mathbb{E}\left[\left|Y_1-\mathbb{E}\left[Y_1\right]\right|^2\right]=\sigma^2$.
  - $\mathbb{E}_{\mathcal{D}}\left[\left|h(\mathcal{D})-m\right|^2\right]=\text{bias}+\text{variance}=0+\sigma^2$.

Bias-Variance Decomposition for the Mean Estimator: Examples

Bias-Variance Decomposition

$$\mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - m|^2\right] = \underbrace{|\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right] - m|^2}_{\text{bias}} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]|^2\right]}_{\text{variance}}.$$

- Zero estimator: $h(\mathcal{D}) = 0$
  - The algorithm behind this estimator does not look at data and always outputs zero. (We do not really want to use it in practice.)
  - Bias $|\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right] - m|^2 = |0 - m|^2 = m^2$.
  - Variance: $\mathbb{E}\left[|h(\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]|^2\right] = \mathbb{E}\left[|0 - \mathbb{E}\left[0\right]|^2\right] = 0$.
  - $\mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - m|^2\right] = \text{bias} + \text{variance} = m^2 + 0$.

Bias-Variance Decomposition for the Mean Estimator: Examples

- Summary:
  - Sample average: $\mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - m|^2\right] = \text{bias} + \text{variance} = 0 + \frac{\sigma^2}{n}$
  - Single-sample estimator:
    $\mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - m|^2\right] = \text{bias} + \text{variance} = 0 + \sigma^2.$
  - Zero estimator: $\mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - m|^2\right] = \text{bias} + \text{variance} = m^2 + 0.$
- These estimators show different behaviour of bias and variance. The zero estimator has no variance (surprising?), but potentially a lot of bias (unless we are "lucky" and the m is in fact very close to 0). On the other hand, the sample average has zero bias, but in general it has a non-zero variance.(Q: When does it have a zero variance?)

Bias-Variance Decomposition for the Mean Estimator

- We could also define error as

$$\mathbb{E}_{\mathcal{D},Y}\left[\left|h(\mathcal{D})-Y\right|^2\right]$$

  instead of $\mathbb{E}_{\mathcal{D}}\left[\left|h(\mathcal{D})-m\right|^2\right]$. This measure the expected squared error of $h(\mathcal{D})$ compared to Y instead of the mean $m=\mathbb{E}[Y]$.

- We have a similar decomposition:

$$\begin{aligned}\mathbb{E}\left[\left|h(\mathcal{D})-Y\right|^2\right] =& \mathbb{E}\left[\left|h(\mathcal{D})-m+m-Y\right|^2\right]\\ =& \mathbb{E}\left[\left|h(\mathcal{D})-m\right|^2\right]+\mathbb{E}\left[\left|m-Y\right|^2\right]+\\ & 2\mathbb{E}\left[\left(h(\mathcal{D})-m\right)\left(m-Y\right)\right].\end{aligned}$$

- The last term is zero because

$$\begin{aligned}\mathbb{E}\left[\left(h(\mathcal{D})-m\right)\left(m-Y\right)\right] &= \mathbb{E}\left[\mathbb{E}\left[\left(h(\mathcal{D})-m\right)\left(m-Y\right)\mid\mathcal{D}\right]\right]\\ &= \mathbb{E}\left[\left(h(\mathcal{D})-m\right)\mathbb{E}\left[m-Y\mid\mathcal{D}\right]\right]=0.\end{aligned}$$

Bias-Variance Decomposition for the Mean Estimator

Bias-Variance Decomposition

$$\mathbb{E}\left[|h(\mathcal{D}) - Y|^2\right] = \underbrace{|\mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right] - m|^2}_{\text{bias}} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathcal{D})\right]|^2\right]}_{\text{variance}} + \underbrace{\mathbb{E}\left[|Y - m|^2\right]}_{\text{Bayes error}}.$$

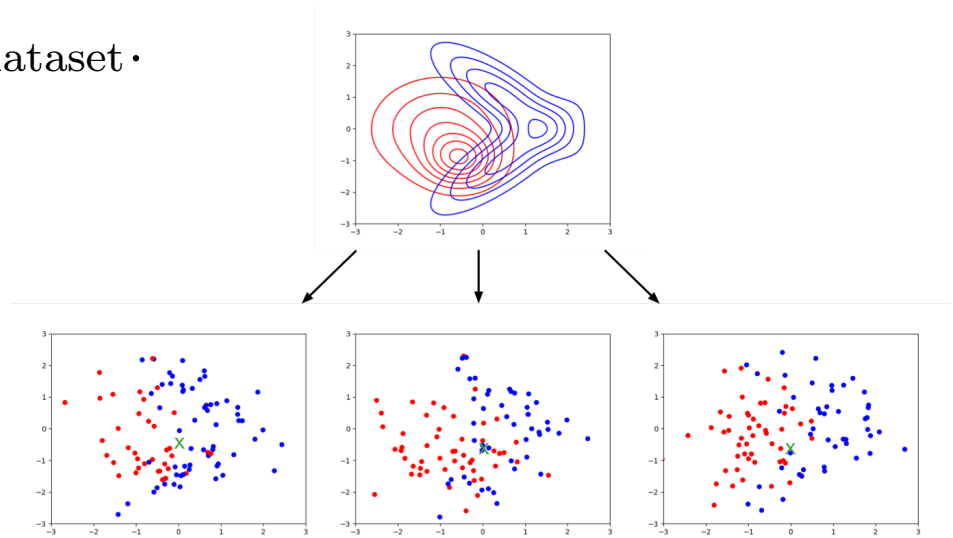
- We have an additional term of $\mathbb{E}\left[|m - Y|^2\right] = \sigma^2$. This is the variance of Y. This comes from the randomness of the r.v. Y and cannot be avoided. This is called the Bayes error.

Bias-Variance Decomposition: General Case

- What about the bias-variance decomposition for a machine learning algorithm such as a regression estimator or a classifier?
- Two importance issues to be addressed:
  - We are not trying to estimate a single real-valued number ($h(\mathcal{D}) \in \mathbb{R}$) anymore, but a function over input $\mathbf{x}$. How can we measure the error in this case?
  - When we only wanted to estimate the mean, the "best" solution was $m = \mathbb{E}[Y]$. What is the best solution here?

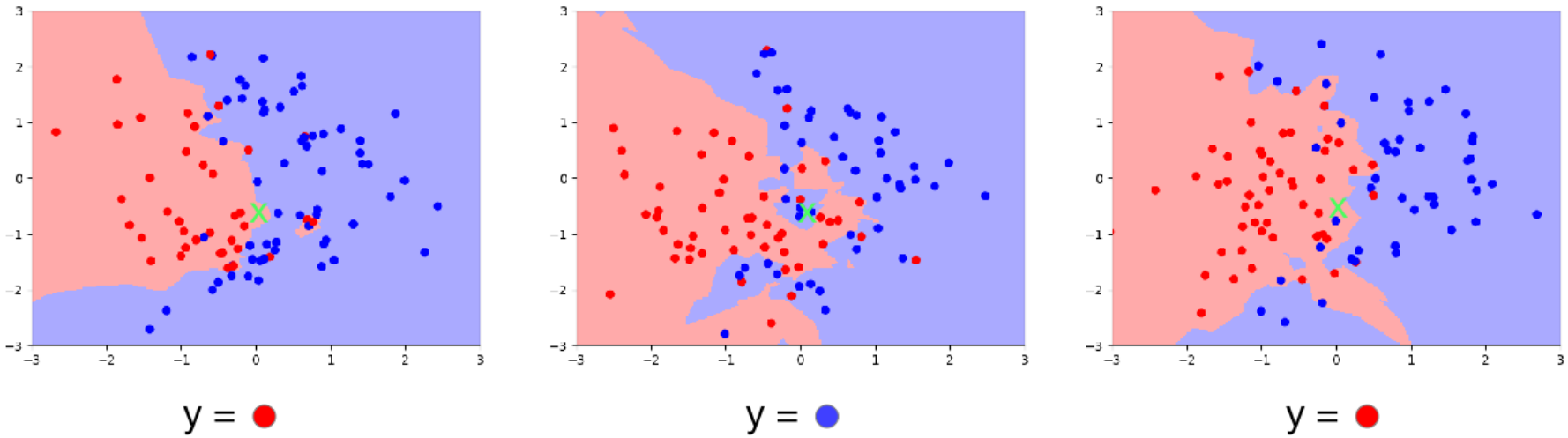
Bias-Variance Decomposition: General Case

- Suppose that the training set $\mathcal{D}$ consists of N pairs $(\mathbf{x}^{(i)}, t^{(i)})$ sampled **independent and identically distributed (i.i.d.)** from a **sample generating distribution** p_{sample}, i.e., $(\mathbf{x}^{(i)}, t^{(i)}) \sim p_{\text{sample}}$.
- Let us denote its marginal distribution on $\mathbf{x}$ by $p_{\mathbf{x}}$.
- Let p_{dataset} denote the induced distribution over training sets, i.e. $\mathcal{D} \sim p_{\text{dataset}}$.
- Pick a fixed query point $\mathbf{x}$ (denoted with a green x).
- Consider an experiment where we sample lots of training datasets i.i.d. from p_{dataset}.



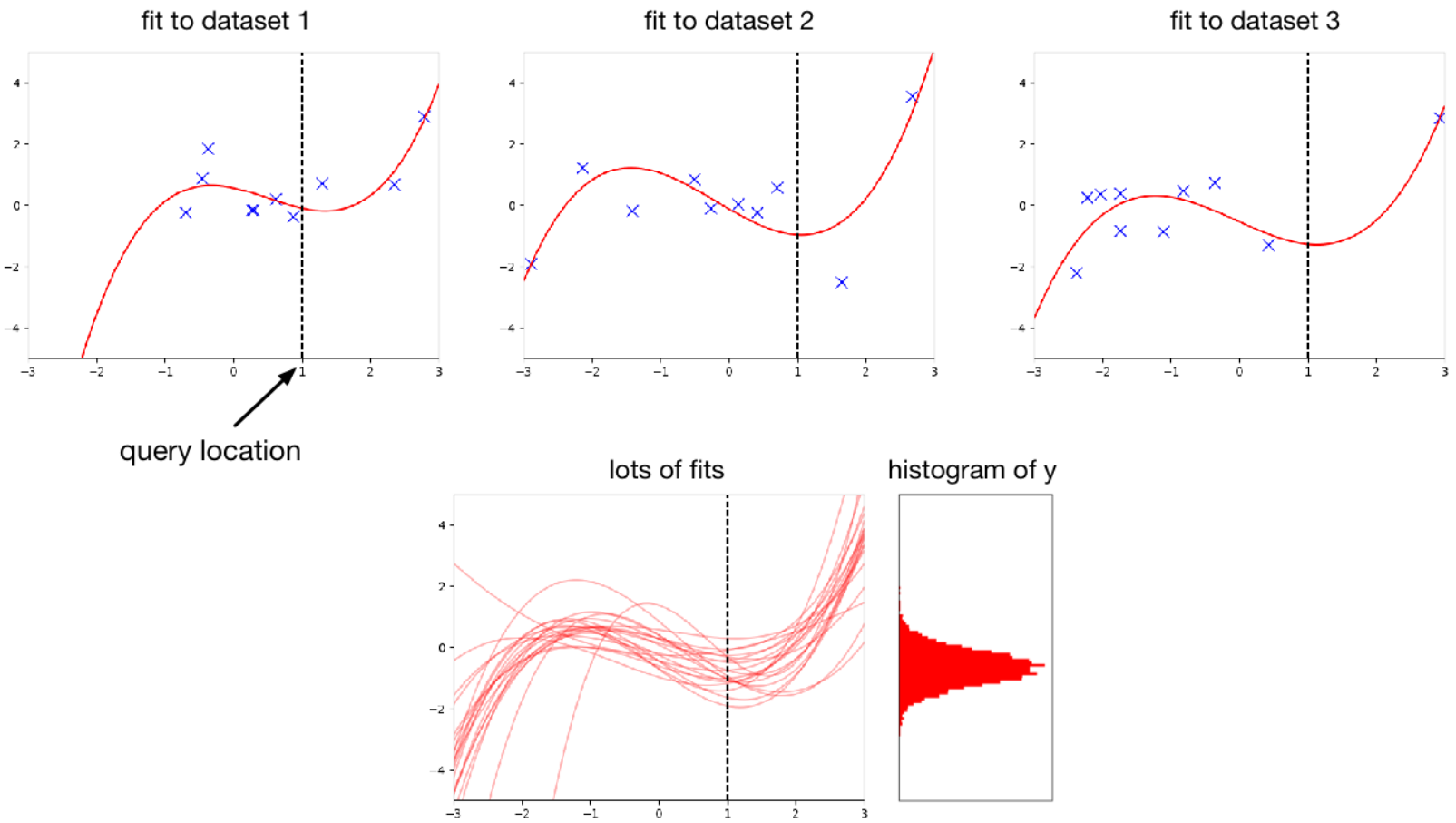
Bias-Variance Decomposition: General Case

- Let us run our learning algorithm on each training set $\mathcal{D}$, producing a regressor or classifier $h(\mathcal{D}) : \mathcal{X} \to \mathcal{T}$.
- Note that $h(\mathcal{D})$ is a random function.
- Fix a query point $\mathbf{x}$. We use $h(\mathcal{D})$ to predict the output at $\mathbf{x}$, i.e., $y = h(\mathbf{x}; \mathcal{D})$.
- y is a random variable, where the **randomness comes from the choice of training set**
  - $\mathcal{D}$ is random $\implies$ $h(\cdot; \mathcal{D})$ is random $\implies$ $h(\mathbf{x}; \mathcal{D})$ is random



Bias-Variance Decomposition: Basic Setup

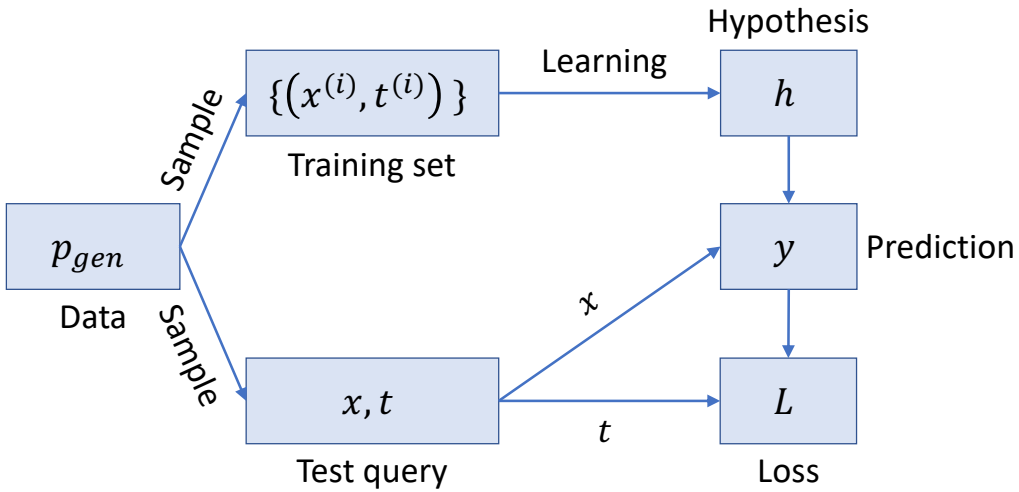
Here is the analogous setup for regression:



Since $y = h(\mathbf{x}; \mathcal{D})$ is a random variable, we can talk about its expectation, variance, etc. over the distribution of training sets p_{dataset}

Bias-Variance Decomposition: General Case

- Recap of the setup:



- When $\mathbf{x}$ is fixed, this is very similar to the mean estimator case.
- Can we have a bias-variance decomposition for a $h(\mathcal{D})$, where we measured $\mathbb{E}_{\mathcal{D}}\left[|h(\mathcal{D}) - m|^2\right]$?
- Two questions:
  - What should replace m in the error decomposition?
  - How should we evaluate the performance when $\mathbf{x}$ is random?

Bayes Optimality

Claim: For a fixed $\mathbf{x}$, the best estimator is the conditional expectation of the target value $y_*(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}]$ (Distribution of $t \sim p(t|\mathbf{x})$), i.e.,

$$y_*(\mathbf{x}) = \operatorname*{argmin}_y \mathbb{E}[(y-t)^2 \,|\, \mathbf{x}].$$

- **Proof:** Start by conditioning on (a fixed) $\mathbf{x}$.

$$\begin{aligned}
\mathbb{E}[(y-t)^2 \,|\, \mathbf{x}] &= \mathbb{E}[y^2 - 2yt + t^2 \,|\, \mathbf{x}] \\
&= y^2 - 2y\mathbb{E}[t \,|\, \mathbf{x}] + \mathbb{E}[t^2 \,|\, \mathbf{x}] \\
&= y^2 - 2y\mathbb{E}[t \,|\, \mathbf{x}] + \mathbb{E}[t \,|\, \mathbf{x}]^2 + \mathrm{Var}[t \,|\, \mathbf{x}] \\
&= y^2 - 2yy_*(\mathbf{x}) + y_*(\mathbf{x})^2 + \mathrm{Var}[t \,|\, \mathbf{x}] \\
&= (y - y_*(\mathbf{x}))^2 + \mathrm{Var}[t \,|\, \mathbf{x}]
\end{aligned}$$

- The first term is nonnegative, and can be made 0 by setting $y = y_*(\mathbf{x})$.
- The second term does not depend on y. It corresponds to the inherent unpredictability, or **noise**, of the targets, and is called the Bayes error or **irreducible error**.
  - This is the best we can ever hope to do with any learning algorithm. An algorithm that achieves it is Bayes optimal.

Bias-Variance Decomposition: General Case

- For each query point $\mathbf{x}$, the expected loss is different. We are interested in quantifying how well our estimator performs over the distribution p_{sample}. That is, the error measure is

$$\begin{aligned}\text{err}(\mathcal{D}) &= \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}}\left[|h(\mathbf{x};D) - y_*(\mathbf{x})|^2\right] \\ &= \int |h(\mathbf{x};D) - y_*(\mathbf{x})|^2 \, p_{\mathbf{x}}(\mathbf{x})\mathrm{d}\mathbf{x}.\end{aligned}$$

- This is similar to $\text{err}(\mathcal{D}) = |h(\mathcal{D}) - m|^2$ of the Mean Estimator case, except that
  - The ideal estimator is $y_*(\mathbf{x})$ and not m.
  - We take average over $\mathbf{x}$ according to the probability distribution $p_{\mathbf{x}}$.
- As before, $\text{err}(\mathcal{D})$ is random due to the randomness of $\mathcal{D} \sim p_{\text{dataset}}$.
- We focus on the expectation of $\text{err}(\mathcal{D})$, i.e.,

$$\mathbb{E}\left[\text{err}(\mathcal{D})\right] = \mathbb{E}_{\mathcal{D}\sim p_{\text{dataset}},\mathbf{x}\sim p_{\mathbf{x}}}\left[|h(\mathbf{x};D) - y_*(\mathbf{x})|^2\right].$$

Bias-Variance Decomposition: General Case

- To obtain the bias-variance decomposition of

$$\mathbb{E}\left[\text{err}(\mathcal{D})\right] = \mathbb{E}_{\mathcal{D}\sim p_{\text{dataset}},\mathbf{x}\sim p_{\mathbf{x}}}\left[\left|h(\mathbf{x};D) - y_*(\mathbf{x})\right|^2\right].$$

  we add and subtract $\mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right]$ inside $|\cdot|$ (similar to the previous case):

$$\mathbb{E}_{\mathcal{D},\mathbf{x}}\left[\left|h(\mathbf{x};\mathcal{D}) - y_*(\mathbf{x})\right|^2\right] =$$

$$\mathbb{E}_{\mathcal{D},\mathbf{x}}\left[\left|h(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right] + \mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right] - y_*(\mathbf{x})\right|^2\right] =$$

$$\mathbb{E}_{\mathcal{D},\mathbf{x}}\left[\left|h(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right]\right|^2\right] + \mathbb{E}_{\mathcal{D},\mathbf{x}}\left[\left|\mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right] - y_*(\mathbf{x})\right|^2\right] +$$

$$2\mathbb{E}_{\mathcal{D},\mathbf{x}}\left[\left(h(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right]\right)\left(\mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right] - y_*(\mathbf{x})\right)\right] =$$

$$\mathbb{E}_{\mathcal{D},\mathbf{x}}\left[\left|h(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right]\right|^2\right] + \mathbb{E}_{\mathbf{x}}\left[\left|\mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right] - y_*(\mathbf{x})\right|^2\right]$$

- Try to convince yourself that the inner product term is zero.
- This is the bias and variance decomposition for the general estimator (with the squared error loss).

Bias-Variance Decomposition for the General Estimator

Bias-Variance Decomposition

$$\mathbb{E}_{\mathcal{D},\mathbf{x}}\left[\left|h(\mathbf{x};\mathcal{D}) - y_*(\mathbf{x})\right|^2\right] = \underbrace{\mathbb{E}_{\mathbf{x}}\left[\left|\mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right] - y_*(\mathbf{x})\right|^2\right]}_{\text{bias}} + \underbrace{\mathbb{E}_{\mathcal{D},\mathbf{x}}\left[\left|h(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right]\right|^2\right]}_{\text{variance}}.$$

- Bias: The squared error between the average estimator (averaged over dataset $\mathcal{D}$) and the best predictor $y_*(\mathbf{x}) = \mathbb{E}\left[t|\mathbf{x}\right]$, averaged over $\mathbf{x} \sim p_{\mathbf{x}}$.
- Variance: The variance of a single estimator $h(\mathbf{x};\mathcal{D})$ (whose randomness comes from $\mathcal{D}$).

Bias-Variance Decomposition: General Case

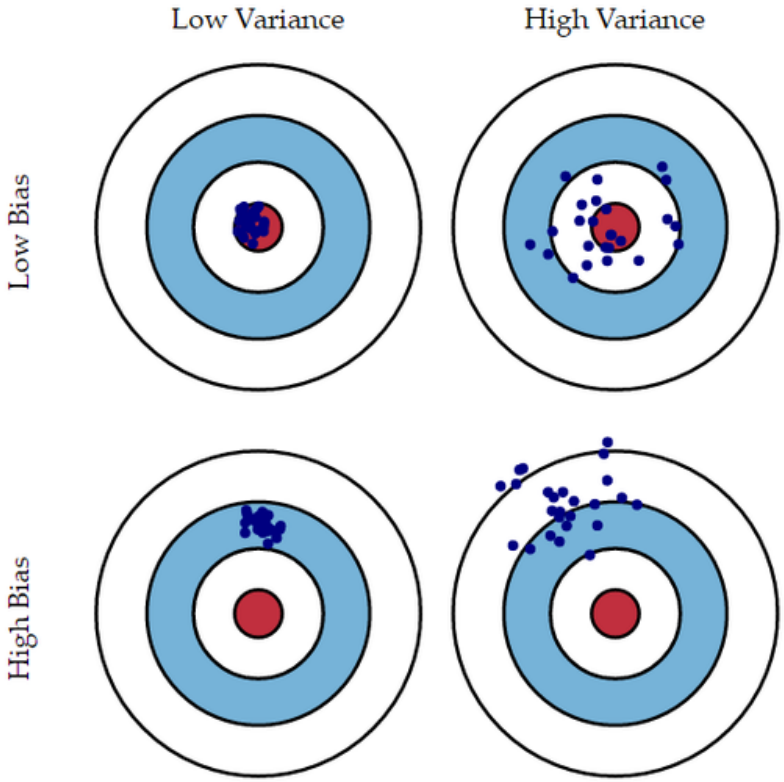
Bias-Variance Decomposition

$$\mathbb{E}_{\mathcal{D},\mathbf{x}}\left[|h(\mathbf{x};\mathcal{D})-t|^2\right] = \underbrace{\mathbb{E}_{\mathbf{x}}\left[|\mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right]-y_*(\mathbf{x})|^2\right]}_{\text{bias}} +$$
$$\underbrace{\mathbb{E}_{\mathcal{D},\mathbf{x}}\left[|h(\mathbf{x};\mathcal{D})-\mathbb{E}_{\mathcal{D}}\left[h(\mathbf{x};\mathcal{D}) \mid \mathbf{x}\right]|^2\right]}_{\text{variance}} + \underbrace{\mathbb{E}\left[|y_*(\mathbf{x})-t|^2\right]}_{\text{Bayes error}}.$$

- We have an additional term of $\mathbb{E}\left[|y_*(\mathbf{x})-t|^2\right] = \mathbb{E}_{\mathbf{x}}\left[\text{Var}[t \mid \mathbf{x}]\right]$. This is due to the the variance of t at each fixed $\mathbf{x}$, averaged over $\mathbf{x} \sim p_{\mathbf{x}}$. As before, this comes from the randomness of the r.v. t and cannot be avoided. This is the Bayes error.

Bias-Variance Decomposition: A Visualization

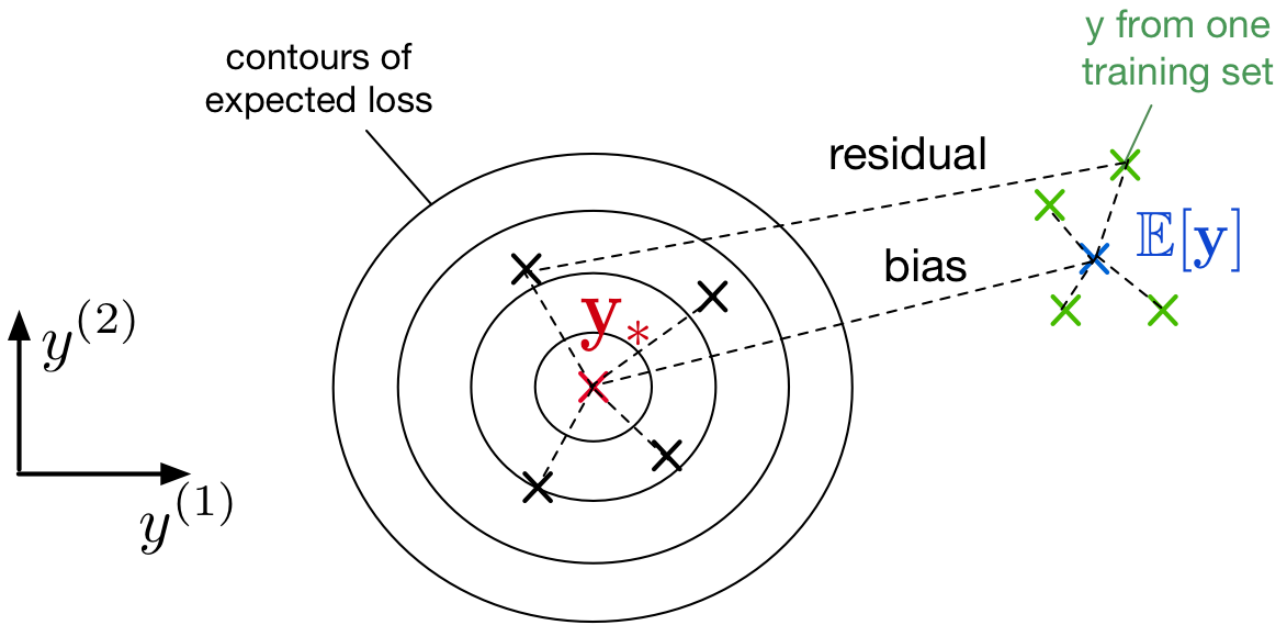
- Throwing darts = predictions for each draw of a dataset



- What doesn't this capture?
- We average over points $\mathbf{x}$ from the data distribution

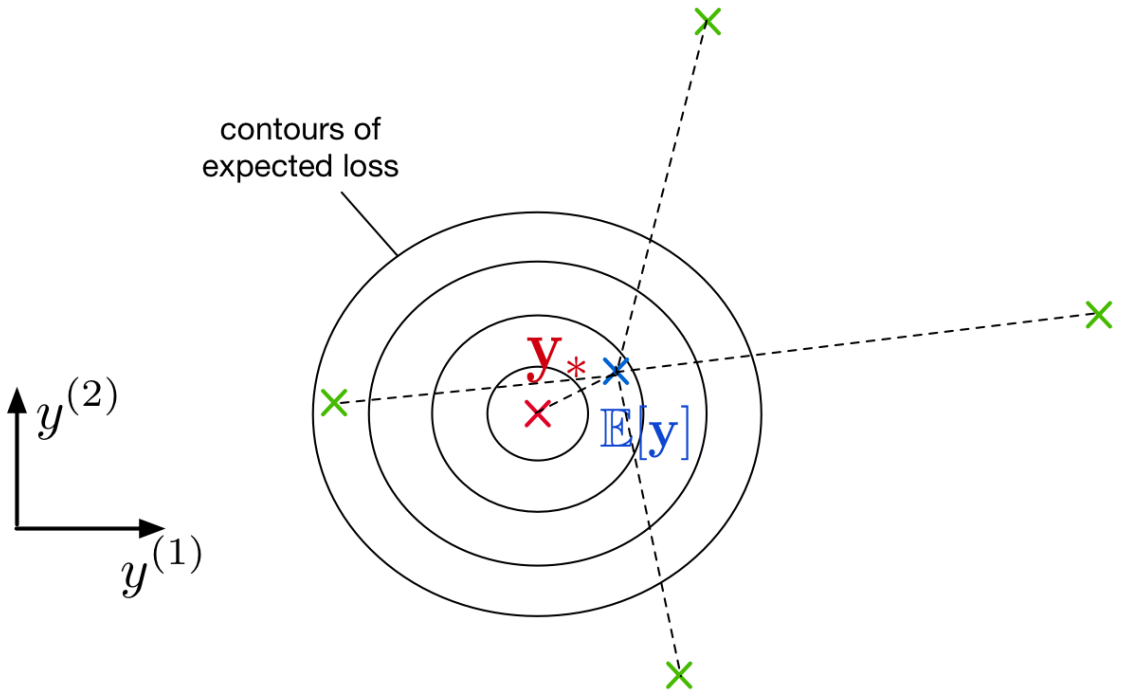
Bias-Variance Decomposition: Another Visualization

- We can visualize this decomposition in **output space**, where the axes correspond to predictions on the test examples.
- If we have an overly simple model (e.g. k-NN with large k), it might have
  - high bias (because it is too simplistic to capture the structure in the data)
  - low variance (because there is enough data to get a stable estimate of the decision boundary)



Bias-Variance Decomposition: Another Visualization

- If you have an overly complex model (e.g. k-NN with $k = 1$), it might have
  - low bias (since it learns all the relevant structure)
  - high variance (it fits the quirks of the data you happened to sample)



Ensemble Methods – Part I: Bagging

Ensemble Methods: Brief Overview

- An ensemble of predictors is a set of predictors whose individual decisions are combined in some way to predict new examples, for example by (weighted) majority vote.
- For the result to be nontrivial, the learned hypotheses must differ somehow, for example because of
  - Different algorithms
  - Different choices of hyperparameters
  - Trained on different data sets
  - Trained with different weighting of the training examples
- Ensembles are usually easy to implement. The hard part is deciding what kind of ensemble you want, based on your goals.
- Two major types of ensembles methods:
  - Bagging
  - Boosting

Bagging: Motivation

- Suppose we could somehow sample m independent training sets $\{\mathcal{D}_i\}_{i=1}^m$ from p_{dataset}.
- We could then learn a predictor $h_i \triangleq h(\cdot; \mathcal{D}_i)$ based on each one, and take the average $h(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^m h_i(\mathbf{x})$.
- How does this affect the terms of the expected loss?
  - **Bias: Unchanged**, since the averaged prediction has the same expectation

$$\begin{aligned}\mathbb{E}_{\mathcal{D}_i,\ldots,\mathcal{D}_m \overset{\text{i.i.d.}}{\sim} p_{\text{dataset}}}\left[h(\mathbf{x})\right] &= \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i \sim p_{\text{dataset}}}\left[h_i(\mathbf{x})\right] \\ &= \mathbb{E}_{\mathcal{D}\sim p_{\text{dataset}}}\left[h(\mathbf{x};\mathcal{D})\right].\end{aligned}$$

  - **Variance: Reduced**, since we are averaging over independent samples

$$\operatorname*{Var}_{\mathcal{D}_1,\ldots,\mathcal{D}_m}\left[h(\mathbf{x})\right] = \frac{1}{m^2}\sum_{i=1}^m \operatorname*{Var}_{\mathcal{D}_i}[h_i(\mathbf{x})] = \frac{1}{m}\operatorname*{Var}_{\mathcal{D}}[h_{\mathcal{D}}(\mathbf{x})].$$

What if $m \to \infty$?

Bagging

- In practice, we don't have access to the underlying data generating distribution p_{sample}.
- It is expensive to collect many i.i.d. datasets from p_{dataset}.
- Solution: bootstrap aggregation, or bagging.
  - Take a single dataset $\mathcal{D}$ with n examples.
  - Generate m new datasets, each by sampling n training examples from $\mathcal{D}$, with replacement.
  - Average the predictions of models trained on each of these datasets.

Bagging

- Problem: the datasets are not independent, so we don't get the $\frac{1}{m}$ variance reduction.
  - Possible to show that if the sampled predictions have variance σ^2 and correlation ρ, then

$$\text{Var}\left(\frac{1}{m}\sum_{i=1}^{m} h_i(\mathbf{x})\right) = \frac{1}{m}(1-\rho)\sigma^2 + \rho\sigma^2.$$

- Ironically, it can be advantageous to introduce *additional* variability into your algorithm, as long as it reduces the correlation between samples.
  - Intuition: you want to invest in a diversified portfolio, not just one stock.
  - Can help to use average over multiple algorithms, or multiple configurations of the same algorithm.

Random Forests

- Random forests: bagged decision trees, with one extra trick to decorrelate the predictions
- When choosing each node of the decision tree, choose a random set of d input features, and only consider splits on those features
- The main idea in random forests is to improve the variance reduction of bagging by reducing the correlation between the trees ($\sim \rho$).
- Random forests are probably the best black-box machine learning algorithm. They often work well with no tuning whatsoever.
  - one of the most widely used algorithms in Kaggle competitions

Classification with Linear Models

Overview

- Classification: predicting a discrete-valued target
  - Binary classification: predicting a binary-valued target

- Examples
  - predict whether a patient has a disease, given the presence or absence of various symptoms
  - classify e-mails as spam or non-spam
  - predict whether a financial transaction is fraudulent

Binary linear classification

- **classification:** predict a discrete-valued target
- **binary:** predict a binary target $t \in \{0, 1\}$
  - Training examples with $t = 1$ are called positive examples, and training examples with $t = 0$ are called negative examples.
  - $t \in \{0, 1\}$ or $t \in \{-1, +1\}$ is for computational convenience.
- **linear:** model is a linear function of $\mathbf{x}$, followed by a threshold r:

$$z = \mathbf{w}^T \mathbf{x} + b$$

$$y = \begin{cases} 1 & \text{if } z \geq r \\ 0 & \text{if } z < r \end{cases}$$

Some Simplifications

Eliminating the threshold

- We can assume without loss of generality (w.l.o.g.) that the threshold is $r = 0$:

$$\mathbf{w}^T\mathbf{x} + b \geq r \quad \iff \quad \mathbf{w}^T\mathbf{x} + \underbrace{b - r}_{\triangleq w_0} \geq 0.$$

Eliminating the bias

- Add a dummy feature x_0 which always takes the value 1. The weight $w_0 = b$ is equivalent to a bias (same as linear regression)

Simplified model

$$z = \mathbf{w}^T\mathbf{x}$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Examples

- Let us consider some simple examples to examine the properties of our model
- Forget about generalization and suppose we just want to learn Boolean functions

Examples

NOT

x_0	x_1	t
1	0	1
1	1	0

- This is our "training set"

- What conditions are needed on w_0, w_1 to classify all examples?
  - When $x_1 = 0$, need: $z = w_0 x_0 + w_1 x_1 > 0 \iff w_0 > 0$
  - When $x_1 = 1$, need: $z = w_0 x_0 + w_1 x_1 < 0 \iff w_0 + w_1 < 0$
- Example solution: $w_0 = 1, w_1 = -2$
- Is this the only solution?

Examples

AND

x_0	x_1	x_2	t
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$$z = w_0 x_0 + w_1 x_1 + w_2 x_2$$

$$\text{need: } w_0 < 0$$

$$\text{need: } w_0 + w_2 < 0$$

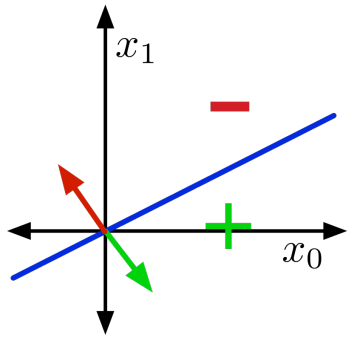
$$\text{need: } w_0 + w_1 < 0$$

$$\text{need: } w_0 + w_1 + w_2 > 0$$

Example solution: $w_0 = -1.5,\ w_1 = 1,\ w_2 = 1$

The Geometric Picture

Input Space, or Data Space for NOT example

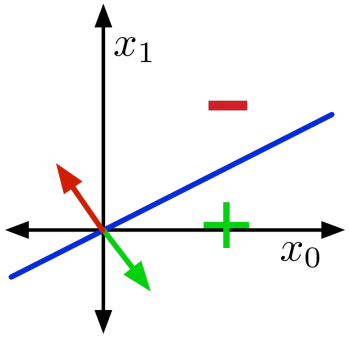


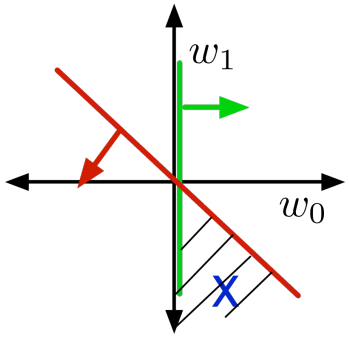
x_0	x_1	t
1	0	1
1	1	0

- Training examples are points
- Weights (hypotheses) $\mathbf{w}$ can be represented by half-spaces $H_+ = \{\mathbf{x} : \mathbf{w}^T\mathbf{x} \geq 0\}$, $H_- = \{\mathbf{x} : \mathbf{w}^T\mathbf{x} < 0\}$
  - The boundaries of these half-spaces pass through the origin (why?)
- The boundary is the decision boundary: $\{\mathbf{x} : \mathbf{w}^T\mathbf{x} = 0\}$
  - In 2-D, it is a line, but think of it as a hyperplane
- If the training examples can be perfectly separated by a linear decision rule, we say data is linearly separable.

The Geometric Picture

Weight Space





$w_0 > 0$

$w_0 + w_1 < 0$

- Weights (hypotheses) $\mathbf{w}$ are points
- Each training example $\mathbf{x}$ specifies a half-space $\mathbf{w}$ must lie in to be correctly classified: $\mathbf{w}^T\mathbf{x} > 0$ if $t = 1$.
- For NOT example:
  - $x_0 = 1, x_1 = 0, t = 1 \implies (w_0, w_1) \in \{\mathbf{w} : w_0 > 0\}$
  - $x_0 = 1, x_1 = 1, t = 0 \implies (w_0, w_1) \in \{\mathbf{w} : w_0 + w_1 < 0\}$
- The region satisfying all the constraints is the feasible region; if this region is nonempty, the problem is feasible, otw it is infeasible.

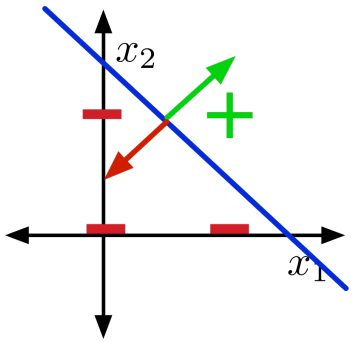
The Geometric Picture

- The **AND** example requires three dimensions, including the dummy one.
- To visualize data space and weight space for a 3-D example, we can look at a 2-D slice.
- The visualizations are similar.
  - Feasible set will always have a corner at the origin.

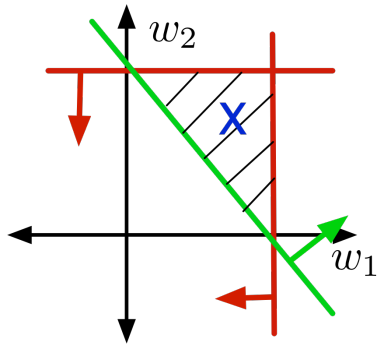
The Geometric Picture

Visualizations of the **AND** example

Data Space



Weight Space



- Slice for $x_0 = 1$ and
- example sol: $w_0 = -1.5$, $w_1 = 1$, $w_2 = 1$
- decision boundary:

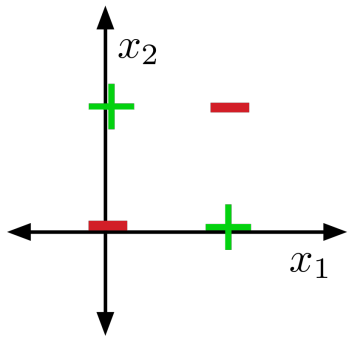
$w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$

$\implies -1.5 + x_1 + x_2 = 0$

- Slice for $w_0 = -1.5$ for the constraints
- $w_0 < 0$
- $w_0 + w_2 < 0$
- $w_0 + w_1 < 0$
- $w_0 + w_1 + w_2 > 0$

The Geometric Picture

Some datasets are not linearly separable, e.g. **XOR**



Overview

- **Recall: binary linear classifiers.** Targets $t \in \{0, 1\}$

$$z = \mathbf{w}^T \mathbf{x} + b$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

- How can we find good values for $\mathbf{w}, b$?
- If training set is separable, we can solve for $\mathbf{w}, b$ using linear programming
- If it's not separable, the problem is harder
  - data is almost never separable in real life.

Loss Functions

- Instead: define loss function then try to minimize the resulting cost function
  - Recall: cost is loss averaged (or summed) over the training set
- Seemingly obvious loss function: 0-1 loss

$$\begin{aligned}\mathcal{L}_{0-1}(y,t) &= \begin{cases} 0 & \text{if } y = t \\ 1 & \text{if } y \neq t \end{cases} \\ &= \mathbb{I}[y \neq t]\end{aligned}$$

Attempt 1: 0-1 Loss

- Usually, the cost $\mathcal{J}$ is the averaged loss over training examples; for 0-1 loss, this is the misclassification rate/error:

$$\mathcal{J} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[y^{(i)} \neq t^{(i)}]$$

Attempt 1: 0-1 Loss

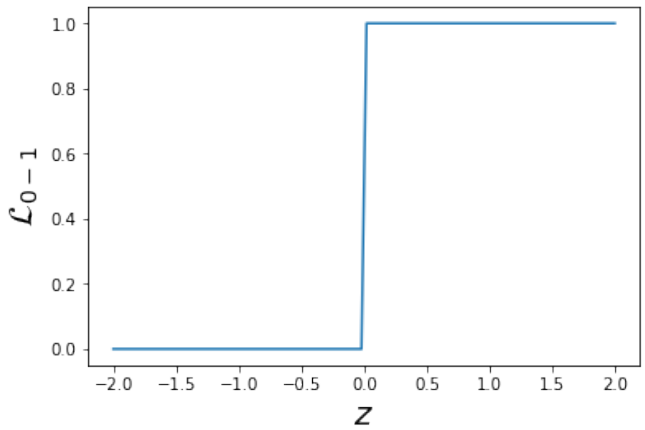
- Problem: how to optimize? In general, a hard problem (can be NP-hard)
- This is due to the step function (0-1 loss) not being nice (continuous/smooth/convex etc)

Attempt 1: 0-1 Loss

- Minimum of a function will be at its critical points.
- Let's try to find the critical point of 0-1 loss
- Chain rule:

$$\frac{\partial \mathcal{L}_{0-1}}{\partial w_j} = \frac{\partial \mathcal{L}_{0-1}}{\partial z} \frac{\partial z}{\partial w_j}$$

- But $\partial \mathcal{L}_{0-1}/\partial z$ is zero everywhere it is defined!



  - $\partial \mathcal{L}_{0-1}/\partial w_j = 0$ means that changing the weights by a very small amount has no effect on the loss (whenever the gradient of the loss is defined)
  - Almost any point has 0 gradient!

Attempt 2: Linear Regression

- Sometimes we can replace the loss function we care about with one that is easier to optimize. This is known as relaxation with a smooth surrogate loss function.
- One problem with $\mathcal{L}_{0-1}$ is that it is defined in terms of final prediction, which inherently involves a discontinuity
- Instead, define loss in terms of $\mathbf{w}^T\mathbf{x} + b$ directly
  - Redo notation for convenience: $z = \mathbf{w}^T\mathbf{x} + b$

Attempt 2: Linear Regression

- We already know how to fit a linear regression model using the squared error loss. Can we use the same squared error loss instead?

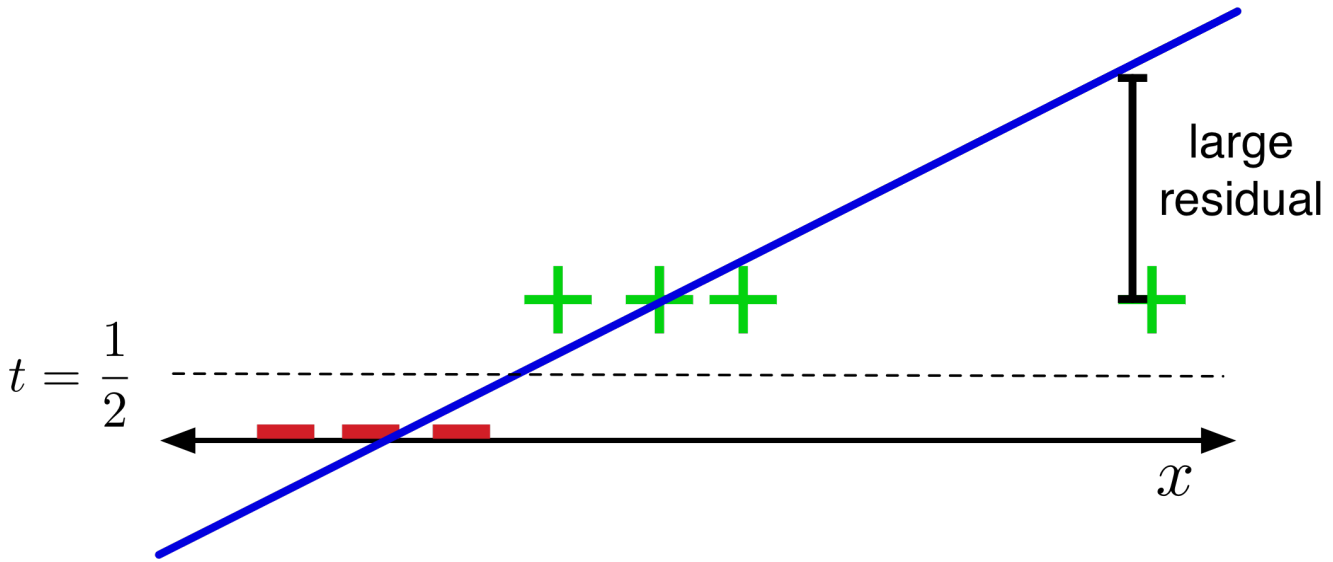
$$z = \mathbf{w}^\top \mathbf{x} + b$$

$$\mathcal{L}_{\text{SE}}(z, t) = \frac{1}{2}(z - t)^2$$

- Doesn't matter that the targets are actually binary. Treat them as continuous values.
- For this loss function, it makes sense to make final predictions by thresholding z at $\frac{1}{2}$ (why?)

Attempt 2: Linear Regression

The problem:



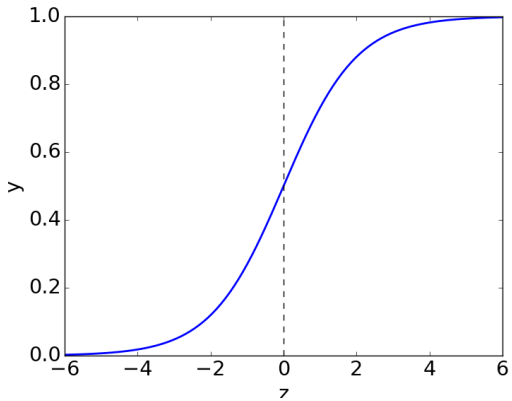
- The loss function penalizes you when you make correct predictions with high confidence!
- If $t = 1$, the loss is larger when $z = 10$ than when $z = 0$.

Attempt 3: Logistic Activation Function

- There's obviously no reason to predict values outside [0, 1]. Let's squash y into this interval.

- The logistic function is a kind of sigmoid, or S-shaped function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



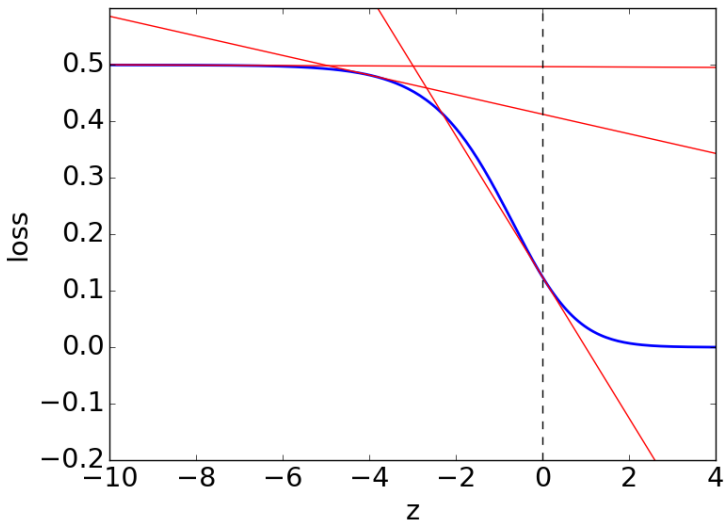
- $\sigma^{-1}(y) = \log(y/(1-y))$ is called the logit.
- A linear model with a logistic nonlinearity is known as log-linear:

$$z = \mathbf{w}^\top \mathbf{x} + b$$
$$y = \sigma(z)$$
$$\mathcal{L}_{\text{SE}}(y, t) = \frac{1}{2}(y - t)^2.$$

- Used in this way, σ is called an activation function.

Attempt 3: Logistic Activation Function

The problem:
(plot of $\mathcal{L}_{\text{SE}}$ as a function of z, assuming $t = 1$)



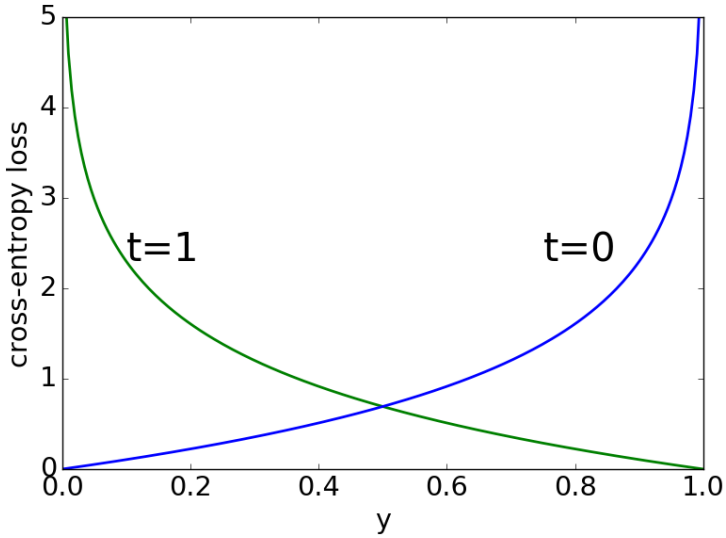
$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial w_j}$$

- For $z \ll 0$, we have $\sigma(z) \approx 0$.
- $\frac{\partial \mathcal{L}}{\partial z} \approx 0$ (check!) $\implies$ $\frac{\partial \mathcal{L}}{\partial w_j} \approx 0$ $\implies$ derivative w.r.t. w_j is small $\implies$ w_j is like a critical point
- If the prediction is really wrong, you should be far from a critical point (which is your candidate solution).

Logistic Regression

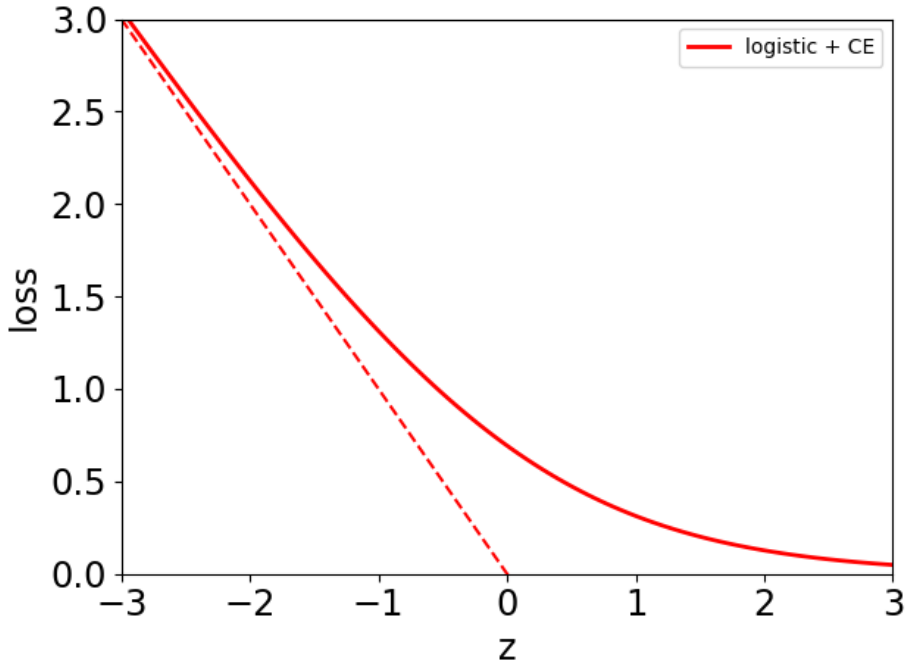
- Because $y \in [0, 1]$, we can interpret it as the estimated probability that $t = 1$.
- The pundits who were 99% confident Clinton would win were much more wrong than the ones who were only 90% confident.
- Cross-entropy loss (aka log loss) captures this intuition:

$$\begin{aligned}\mathcal{L}_{\text{CE}}(y, t) &= \begin{cases} -\log y & \text{if } t = 1 \\ -\log(1 - y) & \text{if } t = 0 \end{cases} \\ &= -t \log y - (1 - t) \log(1 - y)\end{aligned}$$



Logistic Regression:

$$z = \mathbf{w}^\top \mathbf{x} + b$$
$$y = \sigma(z)$$
$$= \frac{1}{1 + e^{-z}}$$
$$\mathcal{L}_{\text{CE}} = -t \log y - (1 - t) \log(1 - y)$$



Plot is for target $t = 1$.

Logistic Regression

- Problem: what if $t = 1$ but you're really confident it's a negative example ($z \ll 0$)?
- If y is small enough, it may be numerically zero. This can cause very subtle and hard-to-find bugs.

$$\begin{aligned} y &= \sigma(z) && \Rightarrow y \approx 0 \\ \mathcal{L}_{\text{CE}} &= -t \log y - (1-t) \log(1-y) && \Rightarrow \text{ computes } \log 0 \end{aligned}$$

- Instead, we combine the activation function and the loss into a single logistic-cross-entropy function.

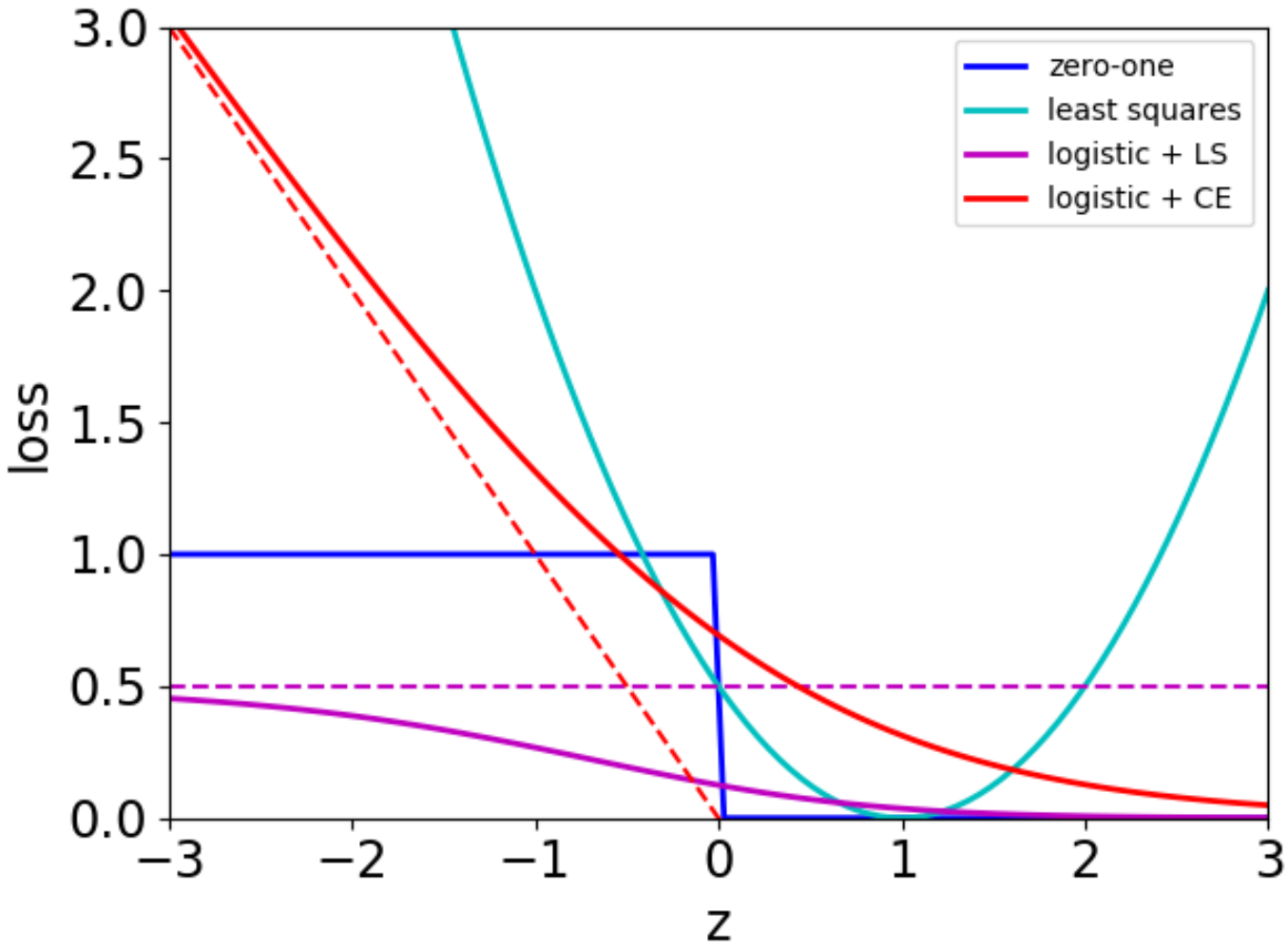
$$\mathcal{L}_{\text{LCE}}(z, t) = \mathcal{L}_{\text{CE}}(\sigma(z), t) = t \log(1 + e^{-z}) + (1-t) \log(1 + e^{z})$$

- Numerically stable computation:

```
E = t * np.logaddexp(0, -z) + (1-t) * np.logaddexp(0, z)
```

Logistic Regression

Comparison of loss functions: (for $t = 1$)



Gradient Descent

- How do we minimize the cost $\mathcal{J}$ in this case? No direct solution.
  - Taking derivatives of $\mathcal{J}$ w.r.t. $\mathbf{w}$ and setting them to 0 doesn't have an explicit solution.
- Now let's see a second way to minimize the cost function which is more broadly applicable: gradient descent.
- Gradient descent is an iterative algorithm, which means we apply an update repeatedly until some criterion is met.
- We initialize the weights to something reasonable (e.g. all zeros) and repeatedly adjust them in the direction of steepest descent.

Gradient for Logistic Regression

Back to logistic regression:

$$\mathcal{L}_{\text{CE}}(y,t) = -\,t\log(y) - (1-t)\log(1-y)$$
$$y = 1/(1+e^{-z}) \quad \text{and} \quad z = \mathbf{w}^T\mathbf{x} + b$$

Therefore

$$\frac{\partial \mathcal{L}_{\text{CE}}}{\partial w_j} = \frac{\partial \mathcal{L}_{\text{CE}}}{\partial y}\cdot\frac{\partial y}{\partial z}\cdot\frac{\partial z}{\partial w_j} = \left(-\frac{t}{y} + \frac{1-t}{1-y}\right)\cdot y(1-y)\cdot x_j$$
$$= (y-t)x_j$$

Exercise: Verify this!

Gradient descent (coordinatewise) update to find the weights of logistic regression:

$$w_j \leftarrow w_j - \alpha\frac{\partial \mathcal{J}}{\partial w_j}$$
$$= w_j - \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})\,x_j^{(i)}$$

Logistic Regression

Comparison of gradient descent updates:

- Linear regression (verify!):

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)}) \, \mathbf{x}^{(i)}$$

- Logistic regression:

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)}) \, \mathbf{x}^{(i)}$$

- Not a coincidence! These are both examples of generalized linear models. But we won't go in further detail.
- Notice $\frac{1}{N}$ in front of sums due to averaged losses. This is why you need smaller learning rate when we optimize the sum of losses ($\alpha' = \alpha/N$).

Stochastic Gradient Descent

- So far, the cost function $\mathcal{J}$ has been the average loss over the training examples:

$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}^{(i)} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y(\mathbf{x}^{(i)}, \boldsymbol{\theta}), t^{(i)}).$$

- By linearity,

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{\theta}}.$$

- Computing the gradient requires summing over *all* of the training examples. This is known as batch training.
- Batch training is impractical if you have a large dataset $N \gg 1$ (e.g. millions of training examples)!

Stochastic Gradient Descent

- Stochastic gradient descent (SGD): update the parameters based on the gradient for a single training example,
  1. Choose i uniformly at random
  2. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{\theta}}$
- Cost of each SGD update is independent of N.
- SGD can make significant progress before even seeing all the data!
- Mathematical justification: if you sample a training example uniformly at random, the stochastic gradient is an unbiased estimate of the batch gradient:

$$\mathbb{E}\left[\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{\theta}}\right] = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}}.$$

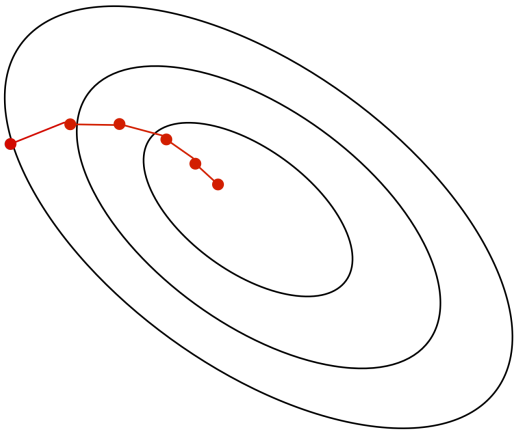
- Problems:
  - Variance in this estimate may be high
  - If we only look at one training example at a time, we can't exploit efficient vectorized operations.

Stochastic Gradient Descent

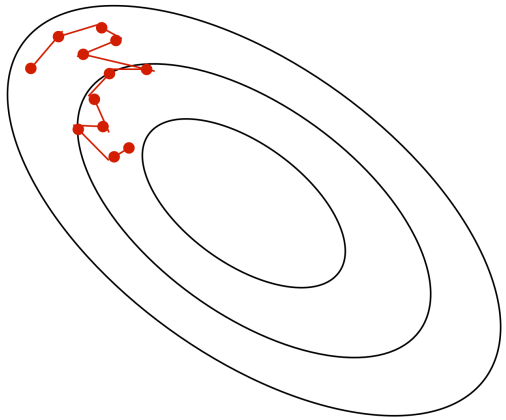
- Compromise approach: compute the gradients on a randomly chosen medium-sized set of training examples $\mathcal{M} \subset \{1, \ldots, N\}$, called a mini-batch.
- Stochastic gradients computed on larger mini-batches have smaller variance. This is similar to bagging.
- The mini-batch size $|\mathcal{M}|$ is a hyperparameter that needs to be set.
  - Too large: takes more computation, i.e. takes more memory to store the activations, and longer to compute each gradient update
  - Too small: can't exploit vectorization, has high variance
  - A reasonable value might be $|\mathcal{M}| = 100$.

Stochastic Gradient Descent

- Batch gradient descent moves directly downhill. SGD takes steps in a noisy direction, but moves downhill on average.



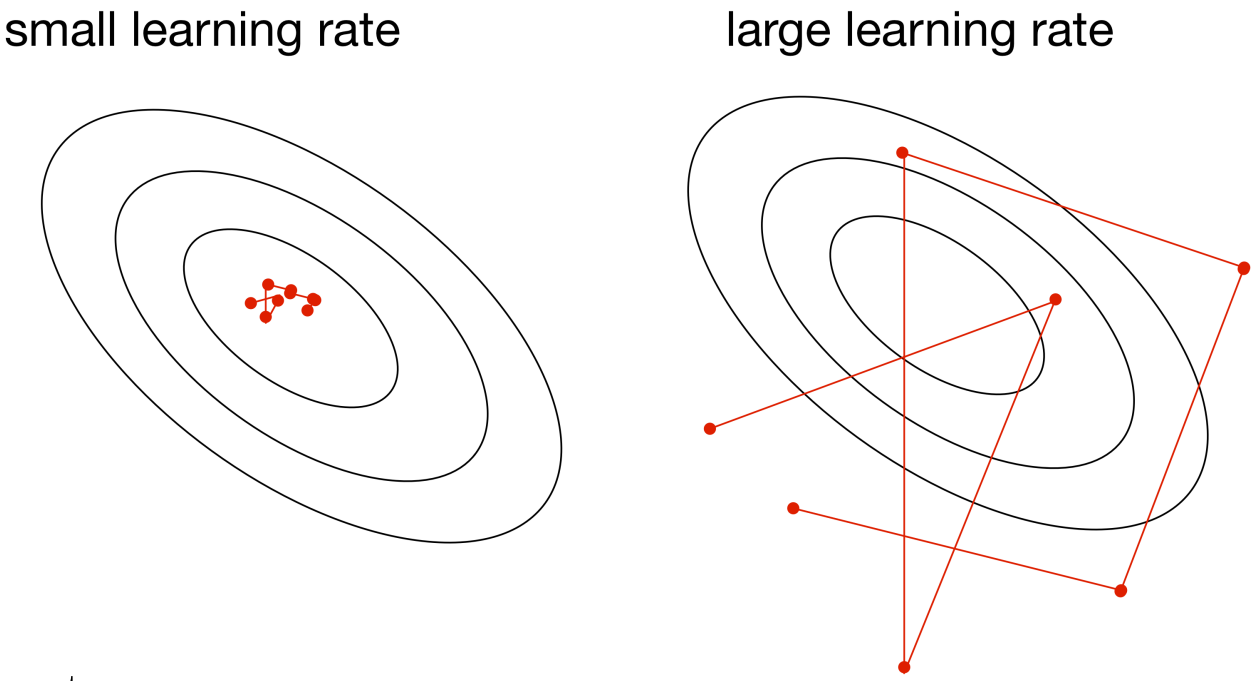
batch gradient descent



stochastic gradient descent

SGD Learning Rate

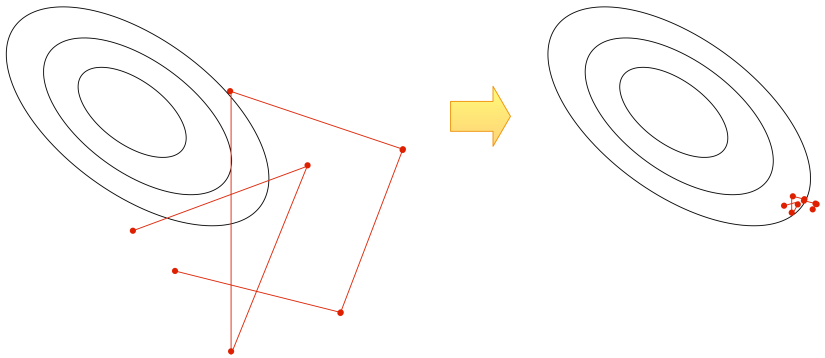
- In stochastic training, the learning rate also influences the fluctuations due to the stochasticity of the gradients.

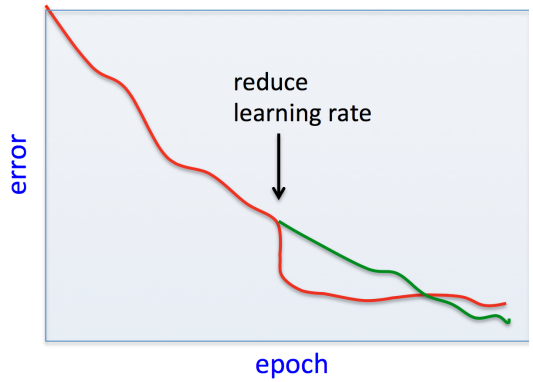


- Typical strategy:
  - Use a large learning rate early in training so you can get close to the optimum
  - Gradually decay the learning rate to reduce the fluctuations

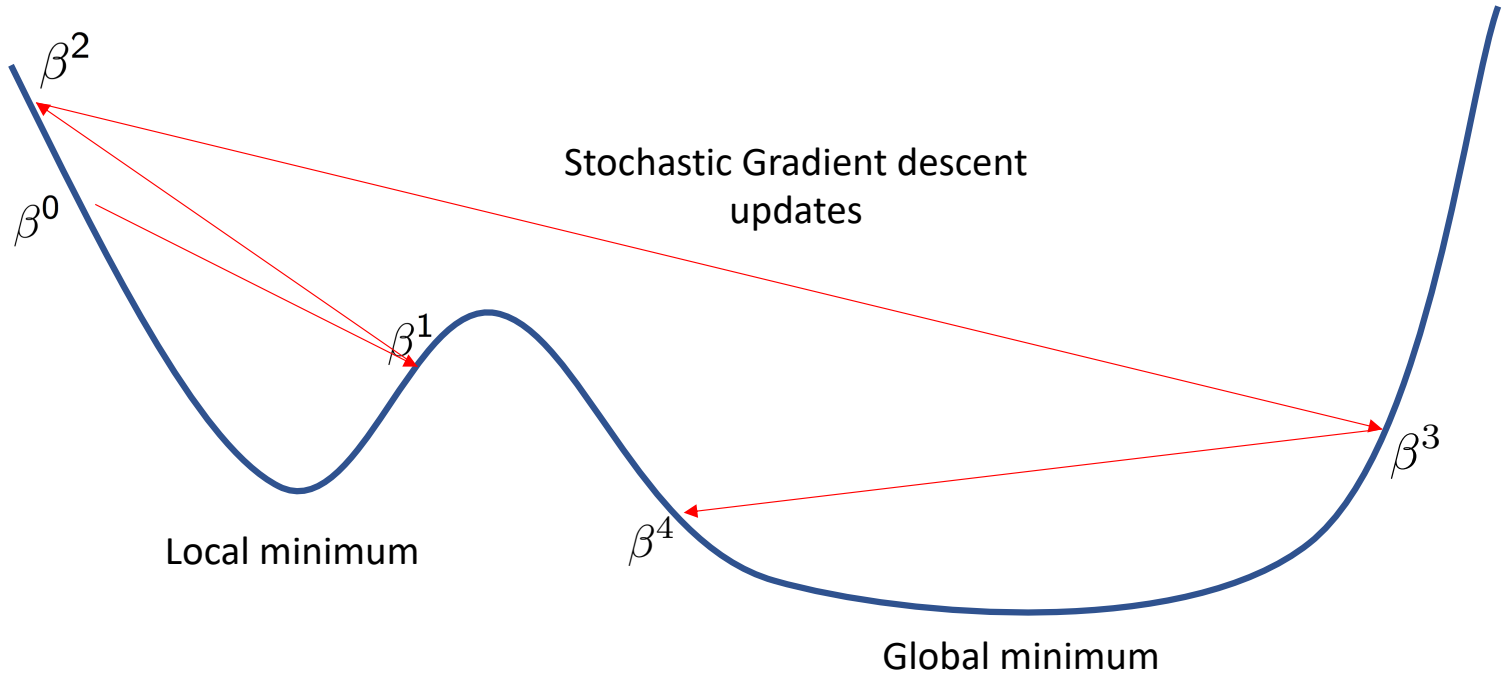
SGD Learning Rate

- Warning: by reducing the learning rate, you reduce the fluctuations, which can appear to make the loss drop suddenly. But this can come at the expense of long-run performance.





SGD and Non-convex optimization



- Stochastic methods have a chance of escaping from bad minima.
- Gradient descent with small step-size converges to first minimum it finds.

Conclusion

- Bias-Variance Decomposition
  - The error of a machine learning algorithm can be decomposed to a bias term and a variance term.
  - Hyperparameters of an algorithm might allow us to tradeoff between these two.
- Ensemble Methods
  - Bagging as a simple way to reduce the variance of an estimation method
- Binary Classification
  - $0-1$ loss is the difficult to work with
  - Use of surrogate loss functions such as the cross-entropy loss lead to computationally feasible solutions
  - Logistic regression as the result of using cross-entropy loss with a linear model going through logistic nonlinearity
  - No direct solution, but gradient descent can be used to minimize it
  - Stochastic gradient descent